Bsc. CSIT 3rd
CHAPTER 8
INPUT AND OUTPUT ORGANIZATION

CONTENTS

8.1 Input-Output Interface: Why IO Interface?, I/O Bus and Interface Modules, I/O vs Memory Bus, Isolated vs Memory Mapped I/O

8.2 Asynchronous Data Transfer: Strobe, Handshaking

8.3 Modes of Transfer: Programmed I/O, Interrupt-Initiated I/O, Direct memory Access

8.4 Priority Interrupt: Polling, Daisy-Chaining, Parallel Priority Interrupt

8.5 DMA and IOP: Direct Memory Access, Input-Output Processor, DMA vs IOP

I/O subsystem

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

Peripheral Devices

- A peripheral device is any device attached to a computer in order to expand its functionality.
- Basically input and output devices together are known as peripherals.
- Input devices are keyboard, optical input devices like bar code reader, screen input devices like touch screen and light pen.
- Output devices may be monitor, printer , speakers, etc

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 communication links for interfacing them with CPU. The purpose of communication link is to resolve the differences that exist between the computer and each peripheral. The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of CPU and memory, which are electronic devices.
2. Data transfer rate of peripherals is slower than that of CPU so some synchronization mechanism may be needed.

3. Data codes and formats in peripherals differ from that of the word format in CPU and memory.

4. Operating modes of peripherals are different from each other and each must be controlled so as not to disturb others.

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

I/O Bus and Interface Modules

A typical communication link between the processor and several peripherals is shown in the figure.

The I/O bus consists of data lines, address lines and control lines. Different peripherals are connected to it. Each peripheral has an interface module associated with it.

Functions of interface are:

- Decodes the device address.
- Decodes the I/O command in control lines.
- Provides signal for the peripheral controller.
- Synchronizes the data flow
- Supervises the transfer rate between peripheral and CPU or memory.

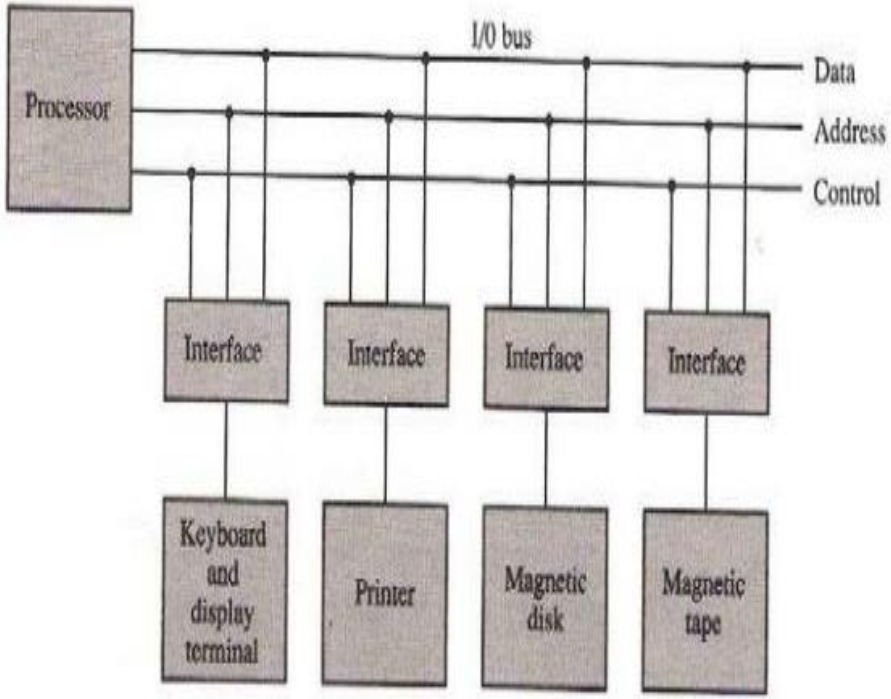


Fig: Connection of I/O bus to I/O devices

The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, 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 line. When the interface detects its own address, then it activates the path between the bus lines and the device that it controls. All the peripherals whose addresses don't correspond to the address in the address bus are disabled by their interface.

At the same time, the address is made available in the address line; the processor provides a function code in the control line which is also called I/O command. The interface selected responds to the function code and proceeds to execute it.

- There are four types of I/O command:
  - ***Control command***:

    The control command is issued to activate and inform the peripheral devices what they have to do. Eg: magnetic tape may be instructed to rewind the tape.
  - ***Status command:***

    It is used to test the various status condition of interface and peripherals. Eg: a computer may wish to check the status of the peripheral before a transfer is initiated.
  - ***Output data command***:

    It is issued to transfer data from system bus to one of storage register.
  - ***Input data command:***

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

I/O and Memory Bus

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

a) Uses two separate buses, one for memory and other for the I/O: Computer has independent sets of data, address and control buses, one for accessing memory and other for I/O. It is usually employed in a computer that has separate Input Output processor (IOP).

b) Use one common bus for both memory and I/O having separate control lines.

c) Use one common bus for memory and I/O with common control lines.

Note:

- Memory bus is for information transfers between the CPU and main memory.
- I/O bus is for information transfers between CPU and I/O devices through their I/O interface.

Isolated I/O versus Memory mapped I/O

Isolated I/O:

In isolated I/O configuration, separate read and write lines are used to distinguish the memory and I/O operations. CPU places the address information on the address bus. It then enables one of the two possible read/write lines (memory r/w or I/O r/w)to specify the operation and then specified device will be activated. Distinct input and output instructions are available. Each is associated with the address of an interface because of small number of I/O devices. The program size is reduced and the decoding speed is increased.

Extra cost for the separate sets r/w buses. Address space of the I/O modules is isolated from the memory address space.

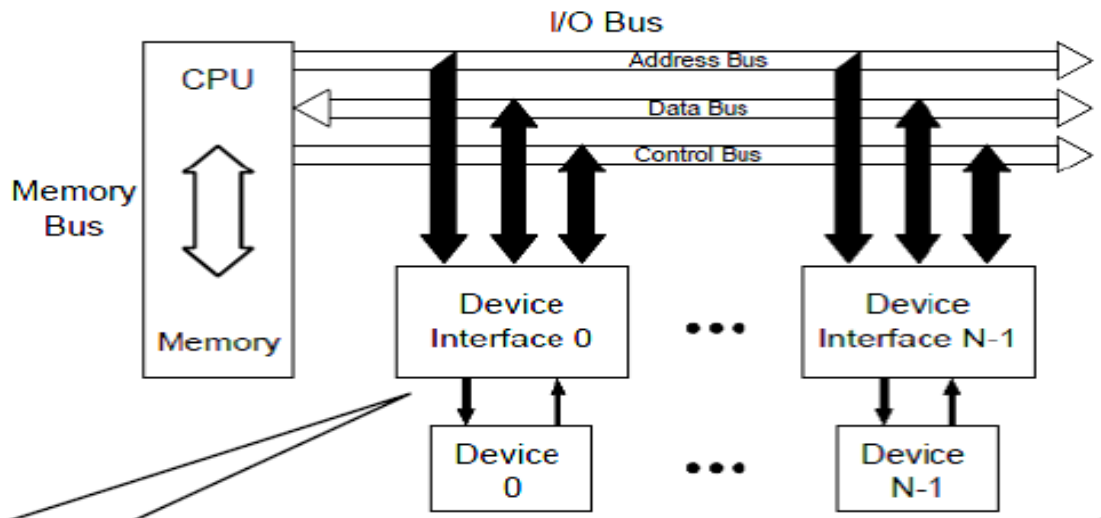
Advantage:

- Same address can be used for either memory and I/O transfer, only control line identifies whether the transfer is I/O or memory.

Disadvantage:

- Needed separate input-output read/write and memory read/write instructions for I/O and memory transfer.

- Two functionally and physically separate buses
  - Memory bus
  - I/O bus
- Each consists of the same three main groupings of wires
  - Address (example might be 6 wires, or up to 2^6 = 64 devices)
  - Data
  - Control



- Each device interface has a port which consists of a minimum of:
  - Control register
  - Status register
  - Data-in (Read) register (or buffer)
  - Data-out (Write) register (or buffer)

- CPU can execute instructions to manipulate the I/O bus separate from the memory bus
- Now only used in very high performance systems

Memory mapped I/O:

The I/O in which one common bus is used for memory and I/O bus with common control lines is called memory mapped I/O. It uses common read/write instruction for memory and I/O operation. It uses same address space for both memory and I/O and treats interface register as a part of memory. The addresses used by interface register cannot be used for memory space. So, if CPU places the register addresses and data on common bus, the memory system ignores the operation. So, I/O operation is performed.

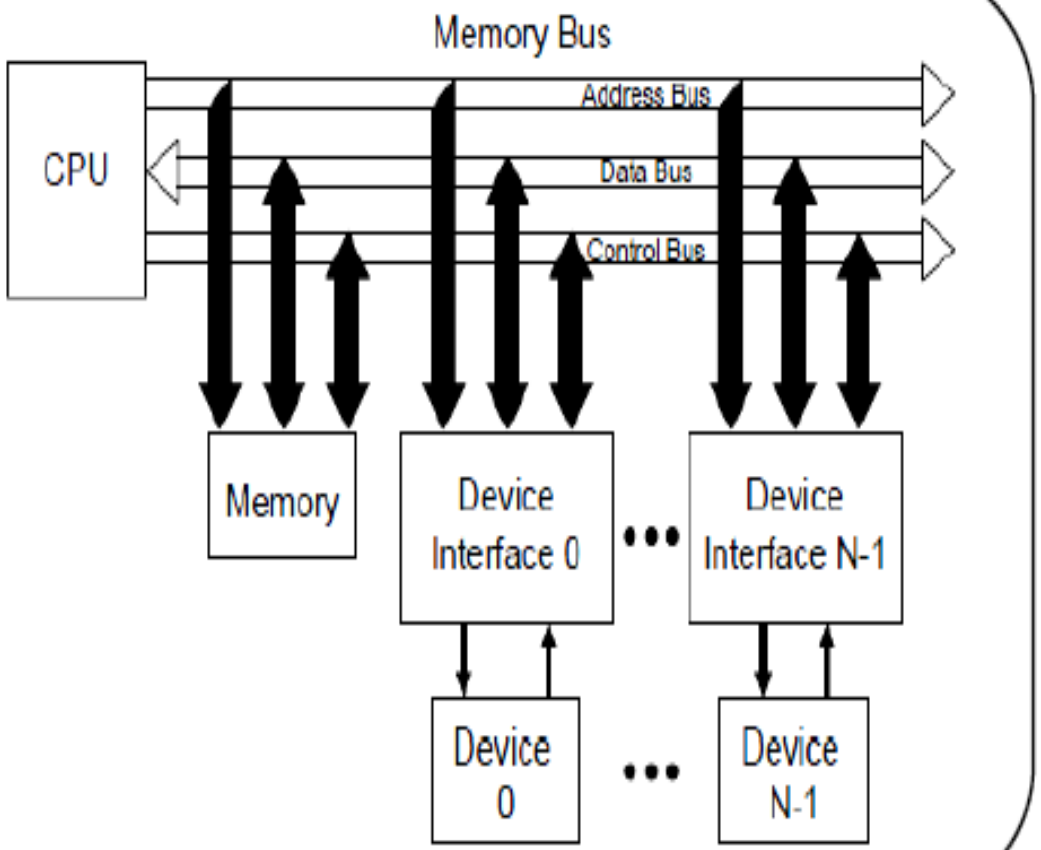
Advantage:

- Same instructions are used for memory and I/O.
- No need of separate control lines for I/O and memory operation.

Disadvantage:

- No full memory address can be used.

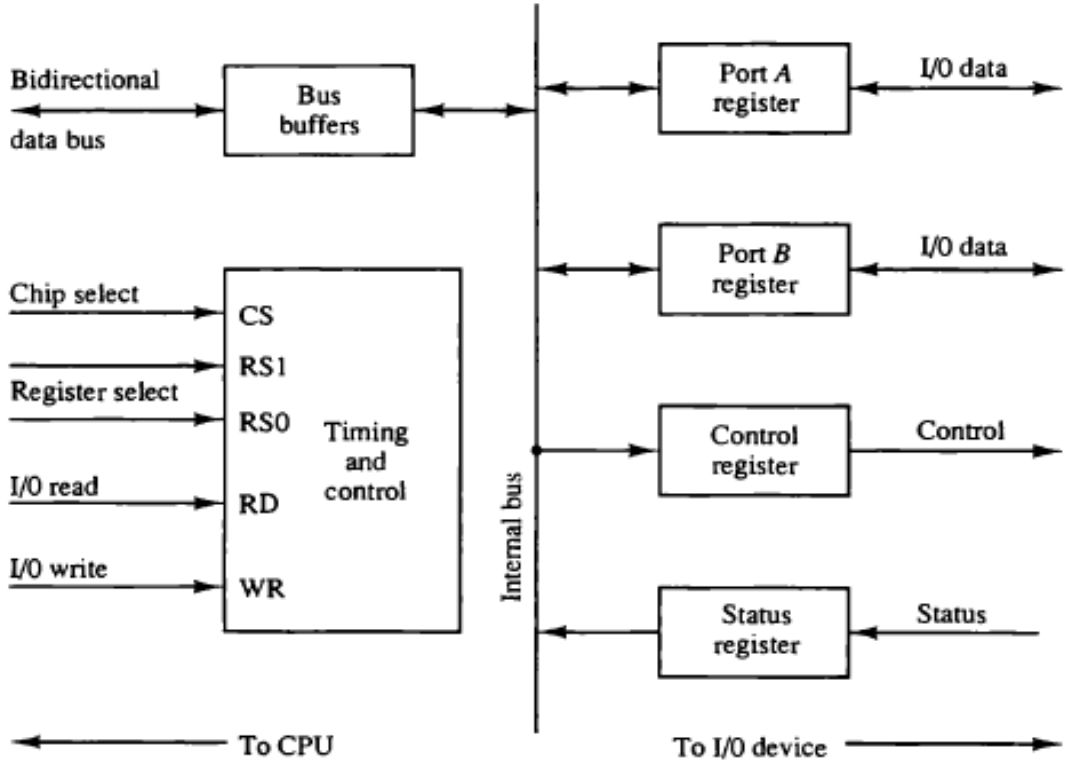
- The memory bus is the *only* bus in the system
- Device interfaces assigned to the *address space* of the CPU or processing element
- Most common way of interfacing devices to computer systems
- CPU can manipulate I/O data residing in interface registers with same instructions that are used to access memory words.
- Typically, a segment of total address space is reserved for interface registers.



Difference between Isolated I/O and memory mapped I/O

Isolated	Memory mapped
1. Separate I/O read/write control lines I addition to the memory read/write control lines.	1. Single set of read/write control lines.
2. Separate I/O and memory address spaces.	2. I/O and memory share common address space.
3. Distinct I/O instruction.	3. Same instruction for memory and I/O

Interface Unit



- Interface communicates with CPU through data bus
- Chip select (CS) and Register select (RS) inputs determine the address assigned to the interface.
- I/O read and write are two control lines that specify input and output.
- 4 registers directly communicates with the I/O device attached to the interface.

CS	RS1	RS0	Register selected
0	x	x	None: data bus in high-impedance
1	0	0	Port *A* register
1	0	1	Port *B* register
1	1	0	Control register
1	1	1	Status register

- Address bus selects the interface unit through CS and RS1 & RS0.
- Particular interface is selected by the circuit (decoder) enabling CS.
- RS1 and RS0 select one of 4 registers.

- Interface communicates with the CPU through the data bus. The chip select and register select inputs determine the address assigned to the interface. Control lines I/O read and write are used to specify the input and output respectively. Bidirectional lines represent both data in and out from the CPU. Information in each port can be assigned a meaning depending on the mode of operation of the I/O device: Port A = Data; Port B =Command; Port C = Status. CPU initializes (loads) each port by transferring a byte to the Control Register. CPU can define the mode of operation of each port.

Asynchronous Data Transfer

- Asynchronous data transfer between two independent units requires that control signal be transmitted between the communicating units to indicate the time at which data is being transmitted.
- One way of achieving this is by a means of ***strobe*** pulse applied by one of the units to indicate to the other unit when the transfer has to occur.
- Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as ***handshaking***.

Strobe Control

It is an asynchronous data transfer method that uses a single control line to time each transfer. The strobe may be activated by either the source or the destination unit. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

a) Source initiated Data Transfer:

- 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 the sufficient time period to allow the destination to receive the data.

- Then the strobe signal is disabled which indicates that the data bus does not contain the valid data. New valid data will be available only after the strobe is enabled again.

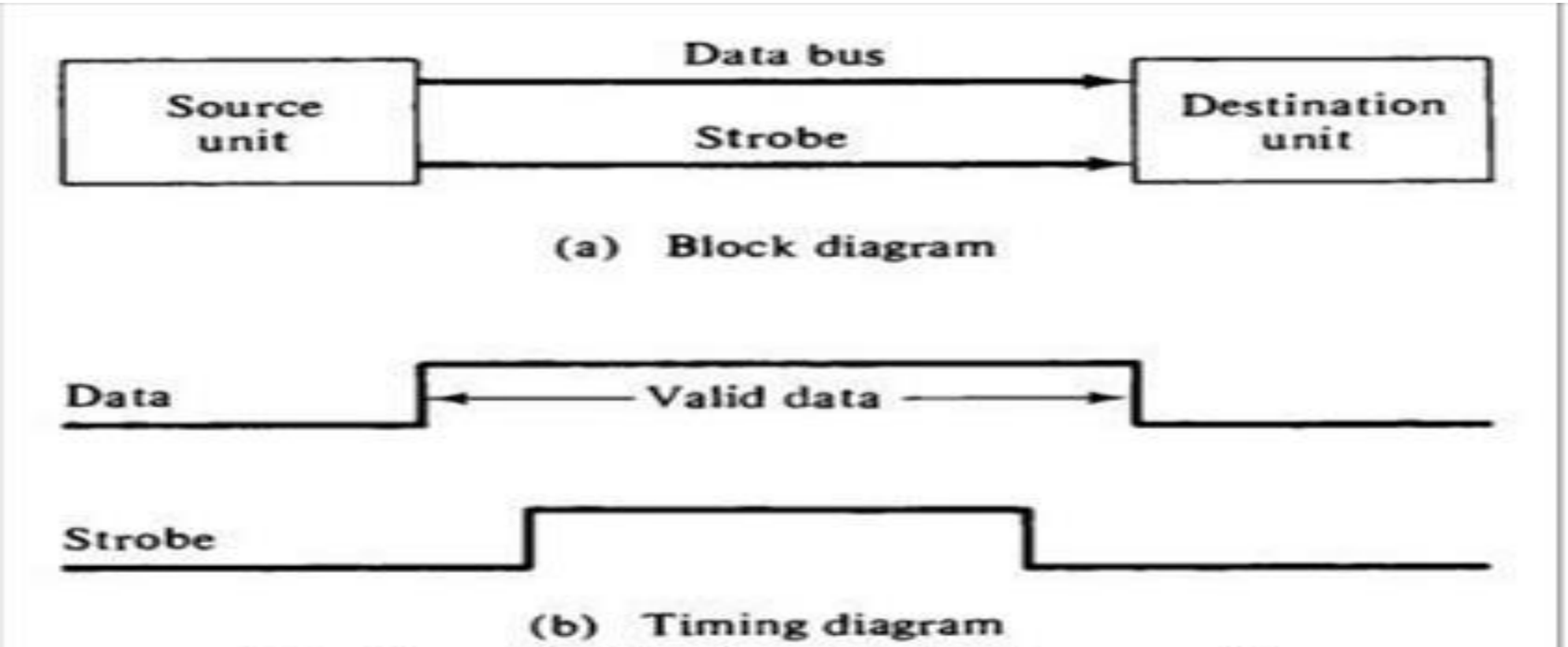


Fig: Source-initiated strobe for data transfer

b) **Destination Initiated Data Transfer:**

- In this case the destination activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it.
- The destination unit then disables the strobe. The source removes the data from the bus after a pre determined time interval.

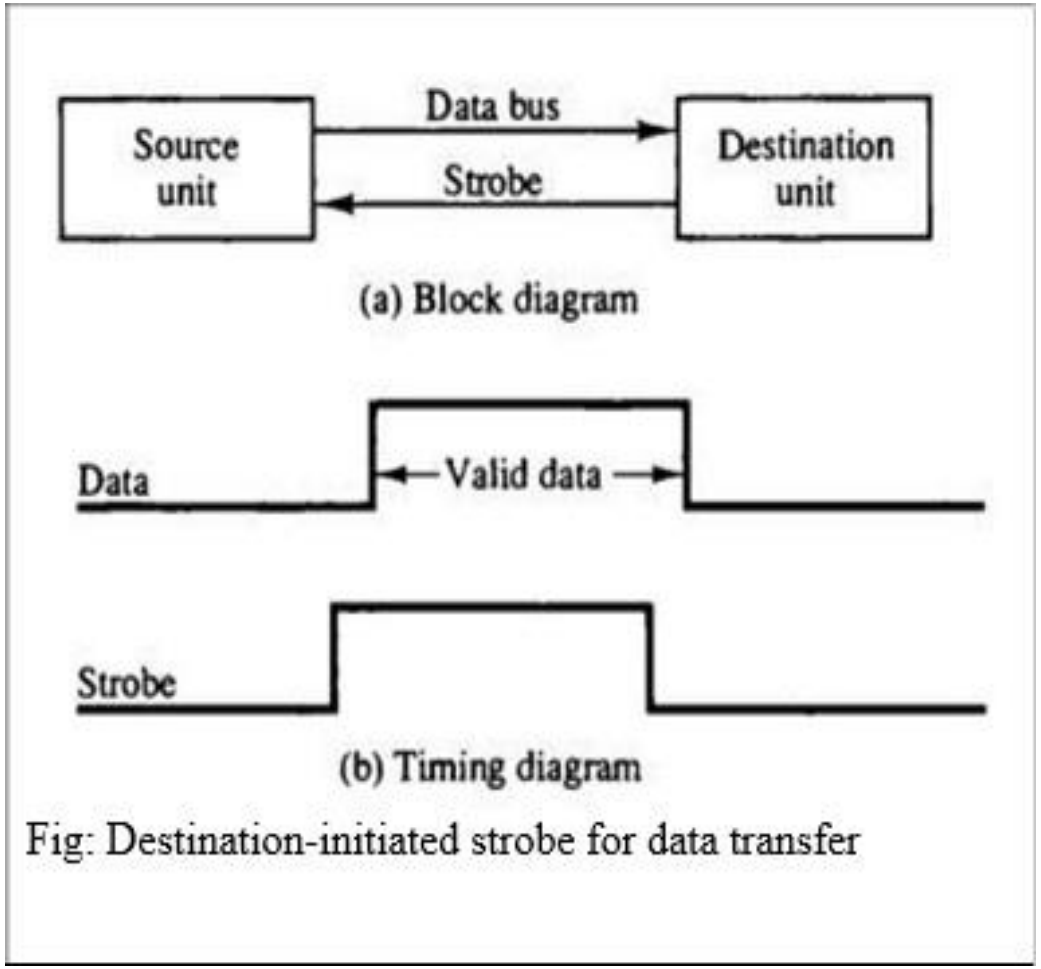


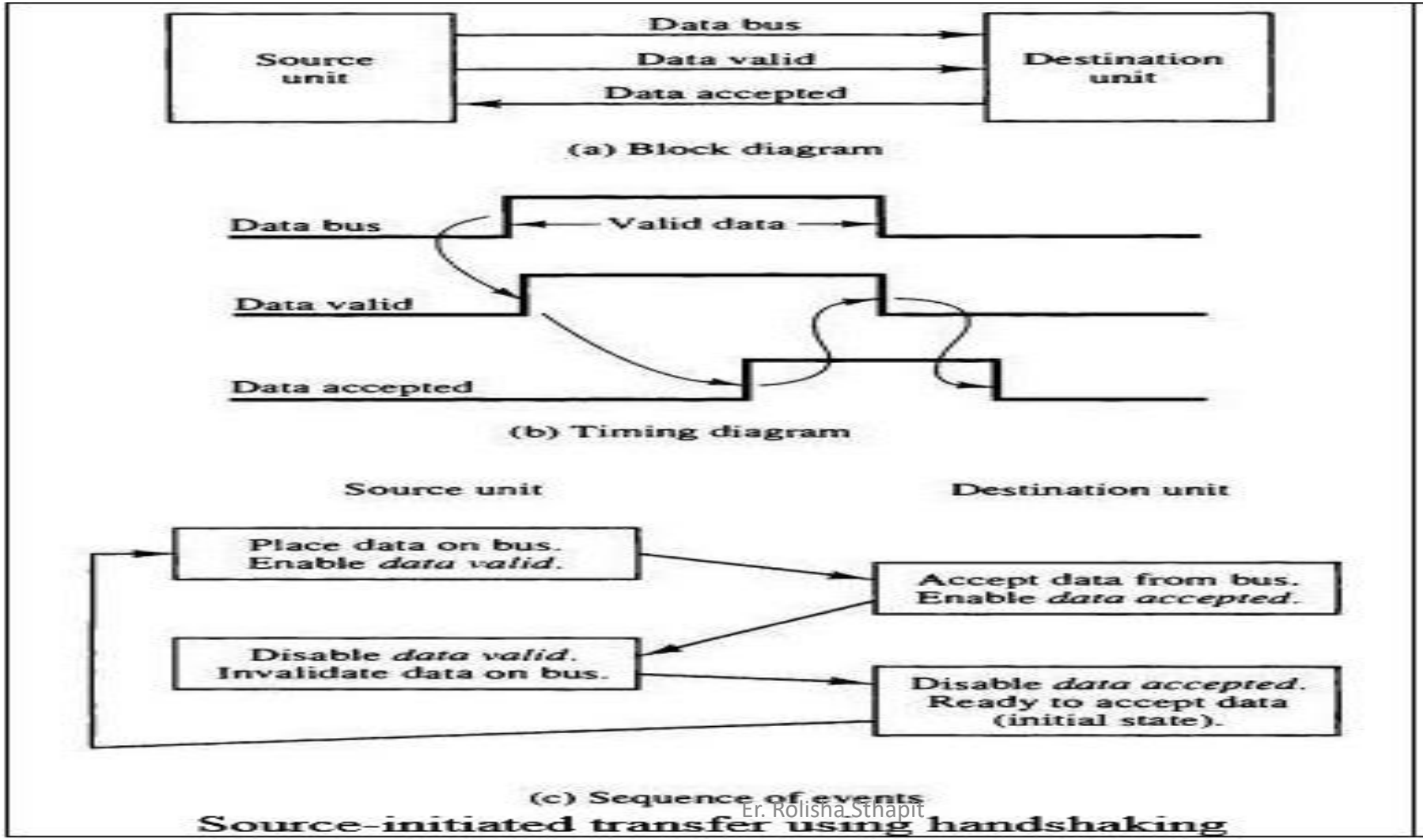
Fig: Destination-initiated strobe for data transfer

- The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has received the data or not. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus. This difficulty is solved by using handshaking method of data transfer.

Handshaking

The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer. The basic principle of the two wire handshaking method of data transfer is as follows: One control line is in the same direction as the data flow in the bus from the source to the destination to inform whether there are valid data in the bus or not. The other control line is in the other direction from the destination to the source that is used by the destination unit to inform the source whether it can accept data or not.

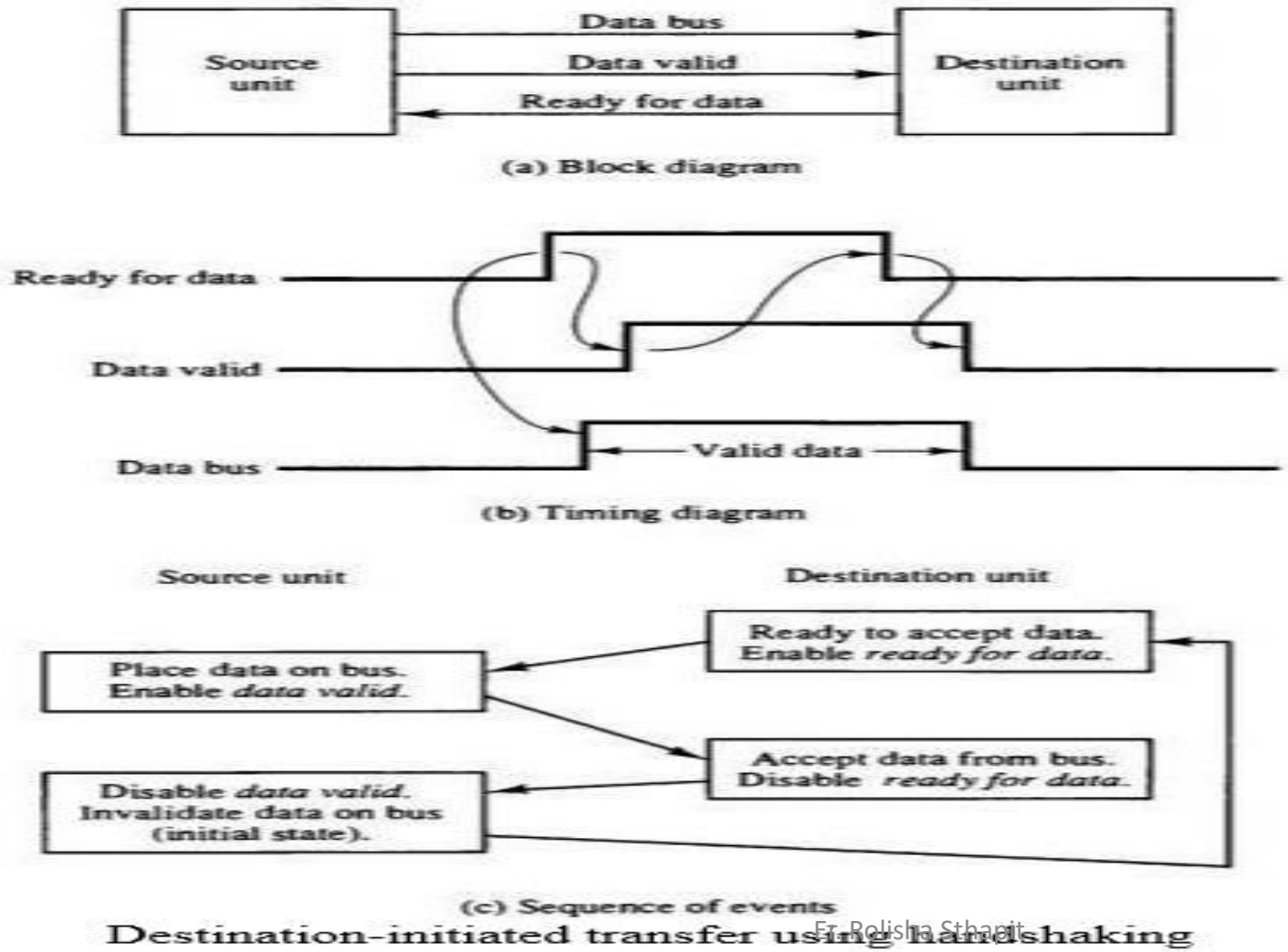
Source-initiated transfer using handshaking:



Source-initiated transfer using handshaking

EXPLAIN THIS FROM BOOK

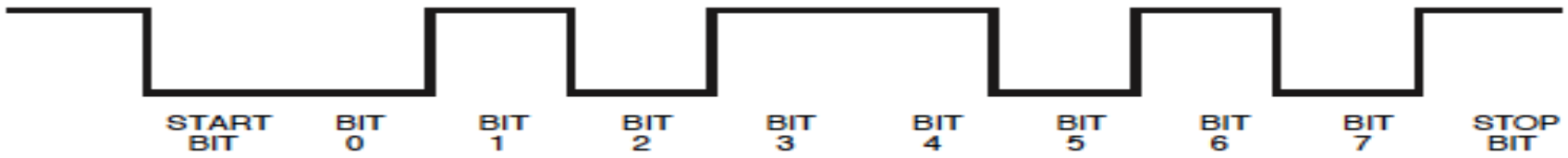
Destination -initiated transfer using handshaking:



EXPLAIN THIS FROM BOOK

Asynchronous Serial Transfer

- The transfer of data between two units may be done in parallel or serial. In parallel data transmission, total message is transmitted at the same time. In serial data transmission, each bit in the message is sent in sequence once at a time. In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.



Asynchronous serial transmission is character oriented. Each character transmitter consists of a start bit, character bits and stop bits. The first bit is called start bit. It is always 0 and is used to indicate the beginning of the character. The last bit is called the stop bit is always set to 1.

Modes of Transfer

There are three different ways by which the data can be transferred:

a) Programmed I/O
b) Interrupt Initiated I/O
c) Direct Memory Access (DMA)

a) Programmed I/O

- In this method of data transfer, the I/O instructions are written in the form of computer program. The instruction written in the program basically involves the data transfer to and from a CPU register and input output peripheral devices. It requires that all I/O operations executed under the direct control of the CPU i.e every data transfer operation involve in an I/O devices requires the execution of an instruction by the CPU. In programmed I/O mode, the CPU stays in the programmed loop until the I/O indicates that it is ready for data transfer. This is time consuming process and keeps the processor busy. This mode is useful in small and low speed systems where hardware cost must be minimized.

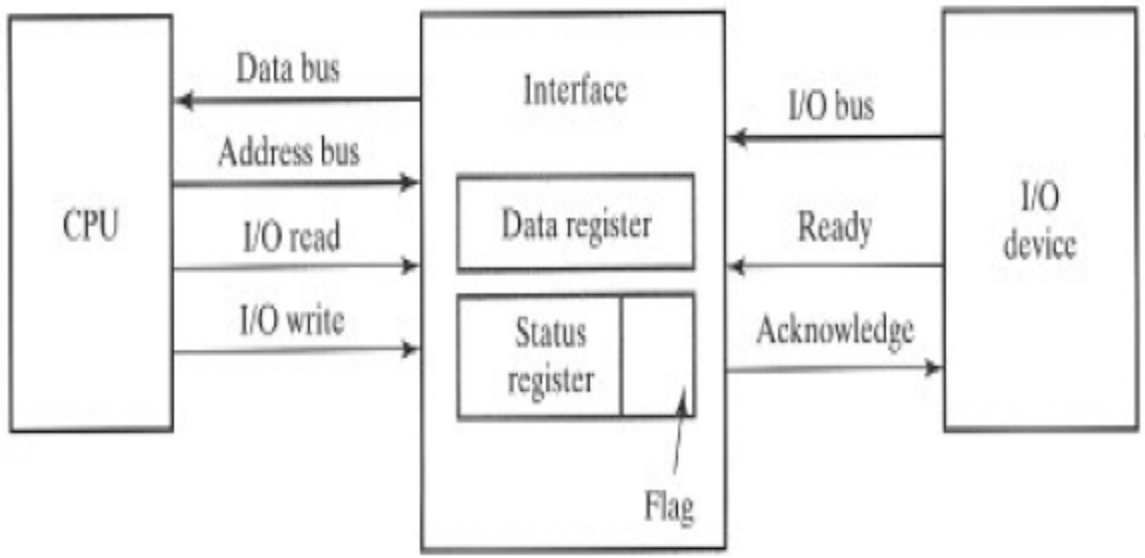
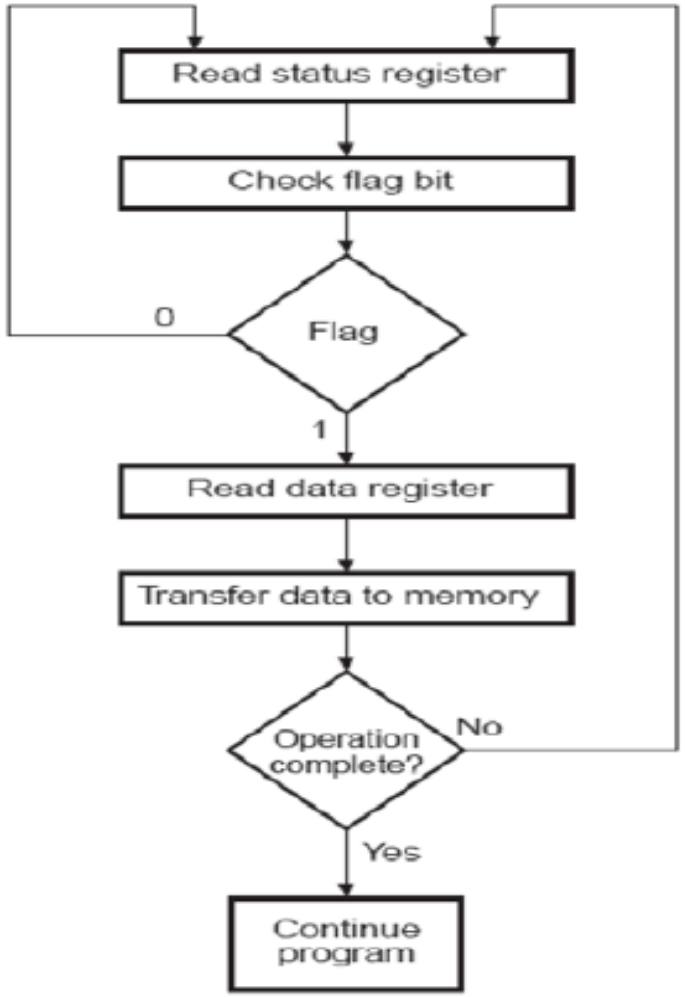


Diagram shows data transfer from I/O device to CPU. Device transfers bytes of data one at a time as they are available. When a byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line. The interface sets a bit in the status register that we will refer to as an F or "flag" bit.

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



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

a) Read status register
b) Check F bit. If not set branch to a) and if set branch to c).
c) Read data register

In general Programmed I/O:

- Continuous CPU involvement
- CPU slowed down to I/O speed
- Simple
- Least hardware

b) Interrupt Initiated I/O

In this mode of transfer the CPU concentrates on some other program. This method uses interrupt signal to inform the CPU that the data are available from the peripheral device and the input output flag is set to 1. When the flag is set, CPU deviates from what it is doing to take care of the input output transfer. After the transfer has been completed, the CPU returns to continue the previous program. When the interrupt signal is generated, the CPU responds to it by storing the return address from program counter into the memory stack and the control branches to a service routine that processes the required I/O transfer.

There are 2 methods to choose the branch address of the service routine. One is called vectored interrupt and the other is called non-vectored interrupt. In vectored interrupt, the source that interrupt gives the branch information to the computer and this interrupt is known as interrupt vector. In non vectored interrupt, the branch address is assigned to a fixed location in memory.

Note: Transfer of data under programmed I/O is between CPU and peripherals.

In general Interrupt Initiated I/O:

-Polling takes valuable CPU time

- Open communication only when some data has to be passed -> Interrupt.

- I/O interface, instead of the CPU, monitors the I/O device
- When the interface determines that the I/O device is ready for data transfer, it generates an Interrupt Request to the CPU
- - Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing

c) Direct Memory Access (DMA)

In this mode , the interface transfers between data into and out of the memory unit through 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 another task. 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.

The CPU merely delays in memory access operation to allow the direct memory I/O transfer.

Priority Interrupt

Interrupt

- When a Process is executed by the CPU and when a user Request for another Process, then this will create disturbance for the Running Process. This is also called as the **Interrupt**.
- Interrupts can be generated by User, Some Error Conditions and also by Software's and the hardware's.

Types of Interrupts

Generally there are three types o Interrupts those are occurred. For Example

1. Internal Interrupt
2. Software Interrupt.
3. External Interrupt

- The **Internal Interrupts** are those which are occurred due to some problem in the execution. For example, when a user performing any operation which contains any type of error so that internal interrupts are those which are occurred by some operations or by some instructions and the operations those are not possible but a user is trying for that operation.
- The **software interrupts** are those which are made some call to the system. For example, while we are processing some instructions and when we want to execute one more application programs.
- The **External Interrupt** occurs when any input and output devices request for any operation and the CPU will execute those instructions first. For example, when a program is executed and when we move the mouse on the screen, then the CPU will handle this external interrupt first and after that he will resume with his operation.

Priority Interrupt:

A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more request arrive simultaneously. Devices with high speed transfer such as magnetic disks are given high priority and slow devices such as keyboards receive low priority. Priority can be established by hardware or software.

1) Software-polling
2) Hardware- Daisy chaining method, parallel priority interrupt.

Polling:

A polling procedure is used to identify the highest priority source by software means. In this method, there is one common branch address for all interrupts. The program that take cases of interrupt begins at the branch address and polls the interrupt sources in sequence. 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. Otherwise, the next lower priority source is tested and so on.

Daisy Chaining Priority (Serial method)(IMP)

The daisy chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority interrupt is placed in the first position followed by a lower priority which is placed last in the chain.

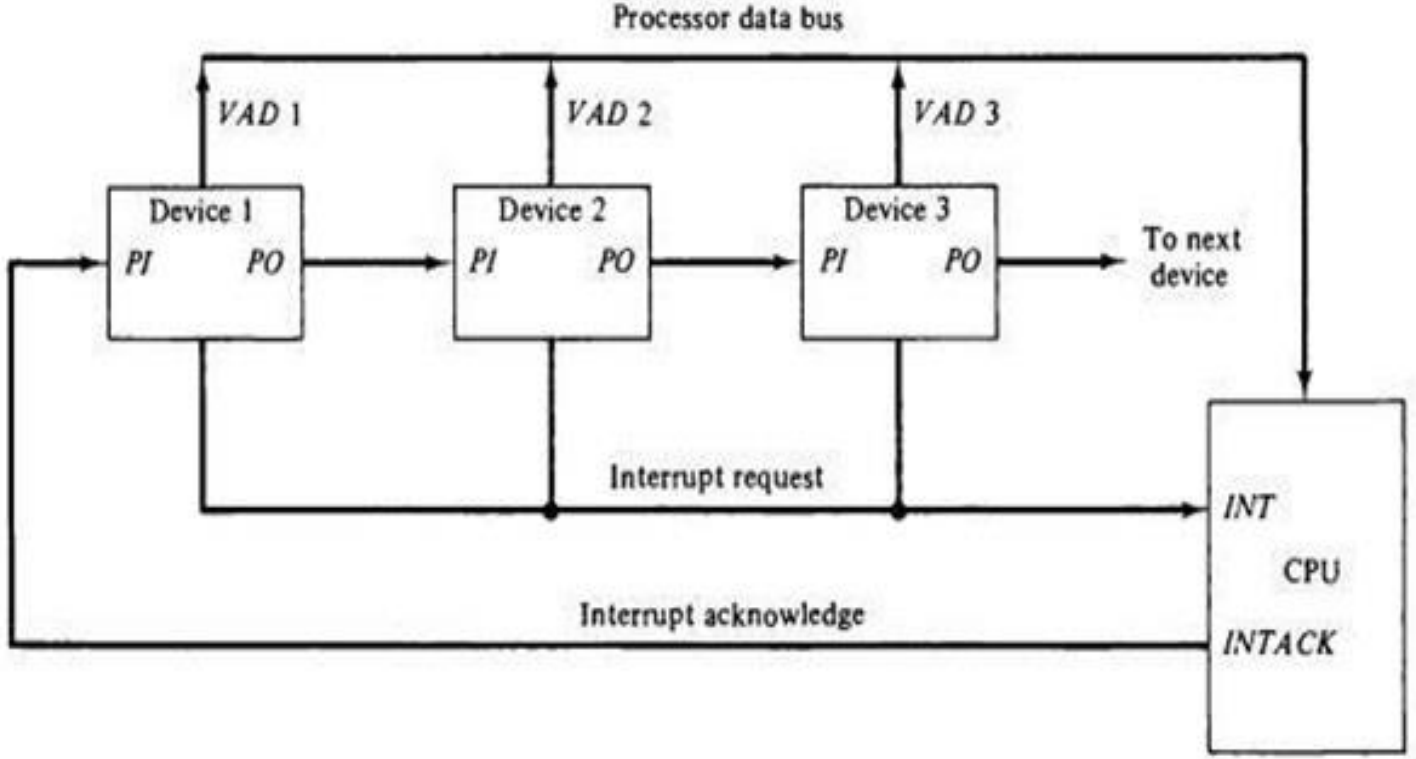


Fig: Daisy-Chain Priority Interrupt

- 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 acknowledged signal passes onto 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 acknowledged signal from the next device by placing a 0 in the PO output. It then proceeds to insert its own interrupt vector address(VAD) 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 PO output to inform the next lower priority device that the acknowledged signal has been blocked. A device that is requesting an interrupt and has a 1 in its PI will intercept (stop) the acknowledge the signal by placing a 0 in its PO. If the device doesnot have pending interrupt, it transmits the acknowledge signal to the next device by placing a 1 in its PO. The device with PI=1 and PO=0 is the one with the highest priority that is requesting an interrupt and this device places its VAD on the data bus.

Parallel Priority Interrupt

The parallel priority interrupt hardware is shown in figure. It has an *interrupt register* whose bits are connected to the interrupt request lines of different devices in the system. It also has a *mask register* whose bits can be used to control the status of each interrupt request. The mask register can be programmed to disable lower priority interrupt while a higher priority device is been serviced.

The mask register has the same number of bits as the interrupt register. By means of program instruction it is possible to set or reset any bit in the mask register. Each interrupt bit and its corresponding mask bit are applied to an AND gate.

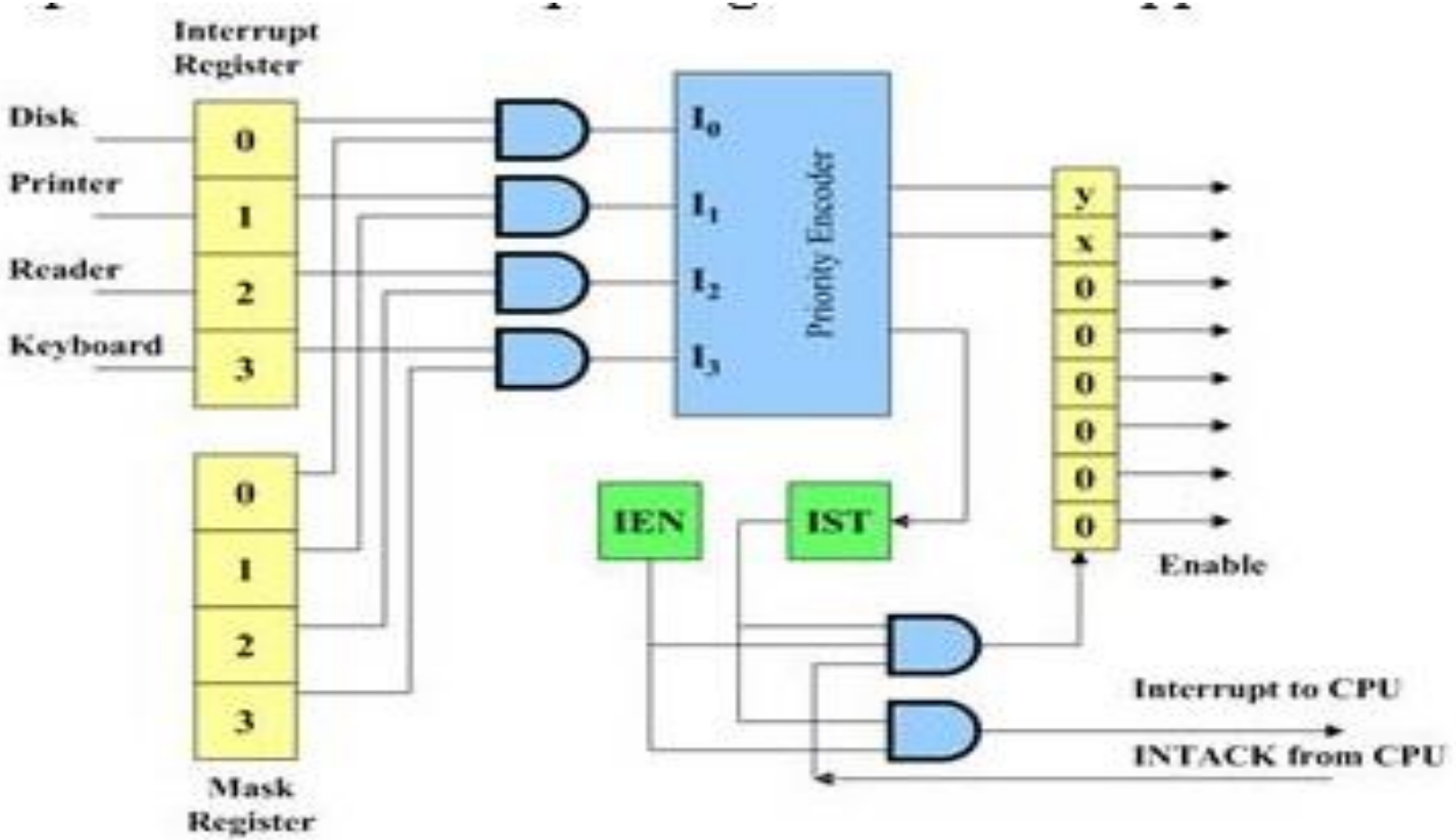


Fig: Parallel Priority Interrupt Hardware

This produces the four inputs to a priority encoder. The priority encoder generates two bits of the vector address. An interrupt is recognized only if its corresponding mask bit is set to 1 by the program. This is transferred to the CPU. Another output from the encoder sets an interrupt status flip flop IST. The outputs of interrupt enable flip-flop IEN and IST are applied to an AND gate. The outputs of this AND gate provide a common interrupt signal for the CPU. The interrupt acknowledge INTACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus.

Priority Encoder

The priority encoder is a circuit that implements the priority function. The logic of the priority encoder is such that if two or more inputs arrive at the same time, the input having the highest priority will take precedence. The truth table of a four-input priority encoder is given below.

Inputs				Outputs			Boolean functions
I_0	I_1	I_2	I_3	x	y	IST	
1	×	×	×	0	0	1	
0	1	×	×	0	1	1	$x = I_0' I_1'$
0	0	1	×	1	0	1	$y = I_0' I_1 + I_0' I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	×	×	0	

Fig: Priority Encoder Truth Table

The X's in the table designate don't care conditions. Input I_0 has the highest priority. When I_0 input is 1, the output generates an output xy=00. I_1 has the next priority level. The output is 01 if I_1=1 and I_0=0. The output for I_2 is generated only if higher priority inputs are 0 and so on. The interrupt status IST is set only when one or more inputs are equal to 1. If all inputs are 0, IST is cleared to 0 and the other outputs of the encoder are not used, so they are marked with don't care conditions.

Direct Memory Access (DMA)(IMP)

DMA is a sophisticated I/O technique in which a DMA controller replaces the CPU and takes care of the access of both, the I/O device and memory, for fast data transfers. To transfer large block of data at high speed, between external devices and main memory, DMA approach is often used.

DMA controller allows data transfer directly between I/O device and Memory, with minimal intervention of processor.

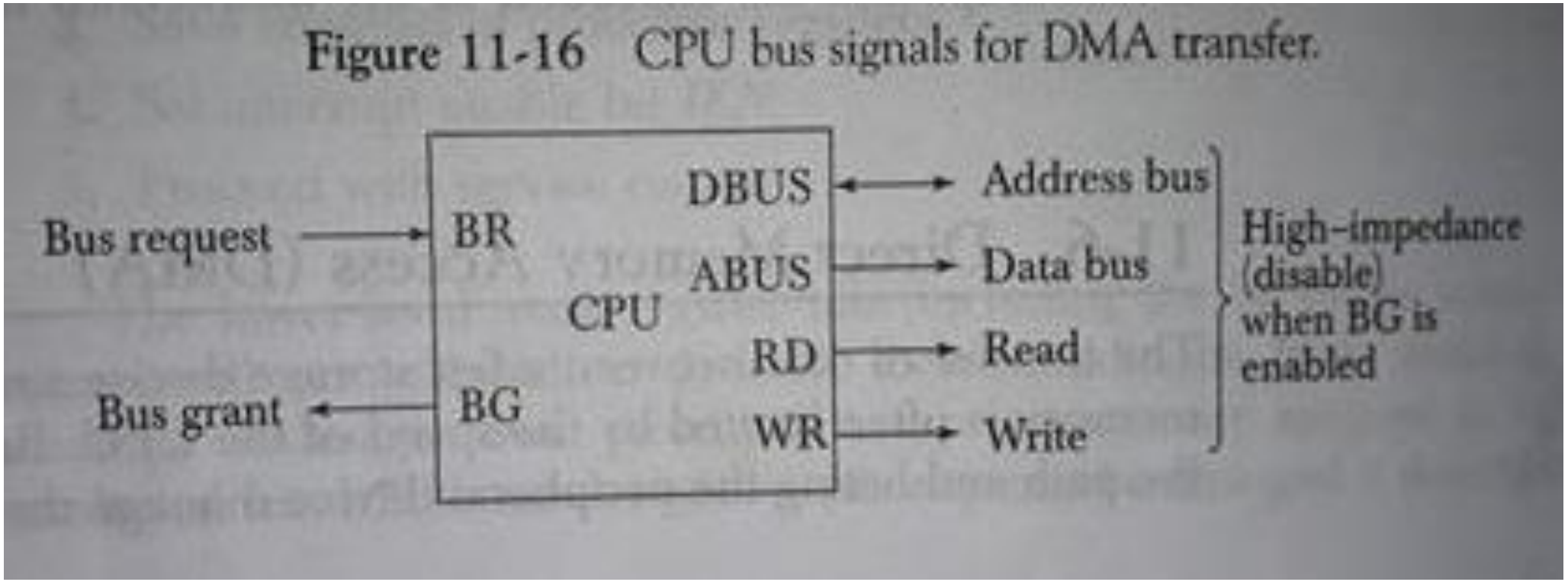
Momentum behind DMA:

Interrupt-driven and Programmed I/O require active CPU intervention (All data must pass through CPU). Transfer rate is limited by processor's ability to service the device and hence CPU is tied up managing I/O transfer. Removing CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.

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

- **Bus Request (BR):** It is used by DMA controller to request the CPU for buses. When this input is active, CPU terminates the execution of the current instruction and places the address bus; data bus and read & write lines into high impedance state. The high impedance state behaves like an open circuit i.e. output is disconnected.
- **Bus Grant (BG):** CPU activates BG output to inform DMA that buses are available (in high impedance state). DMA now take control over buses to conduct memory transfers without processor intervention. When DMA terminates the transfer, it disables the BR line and CPU disables BG and returns to normal operation

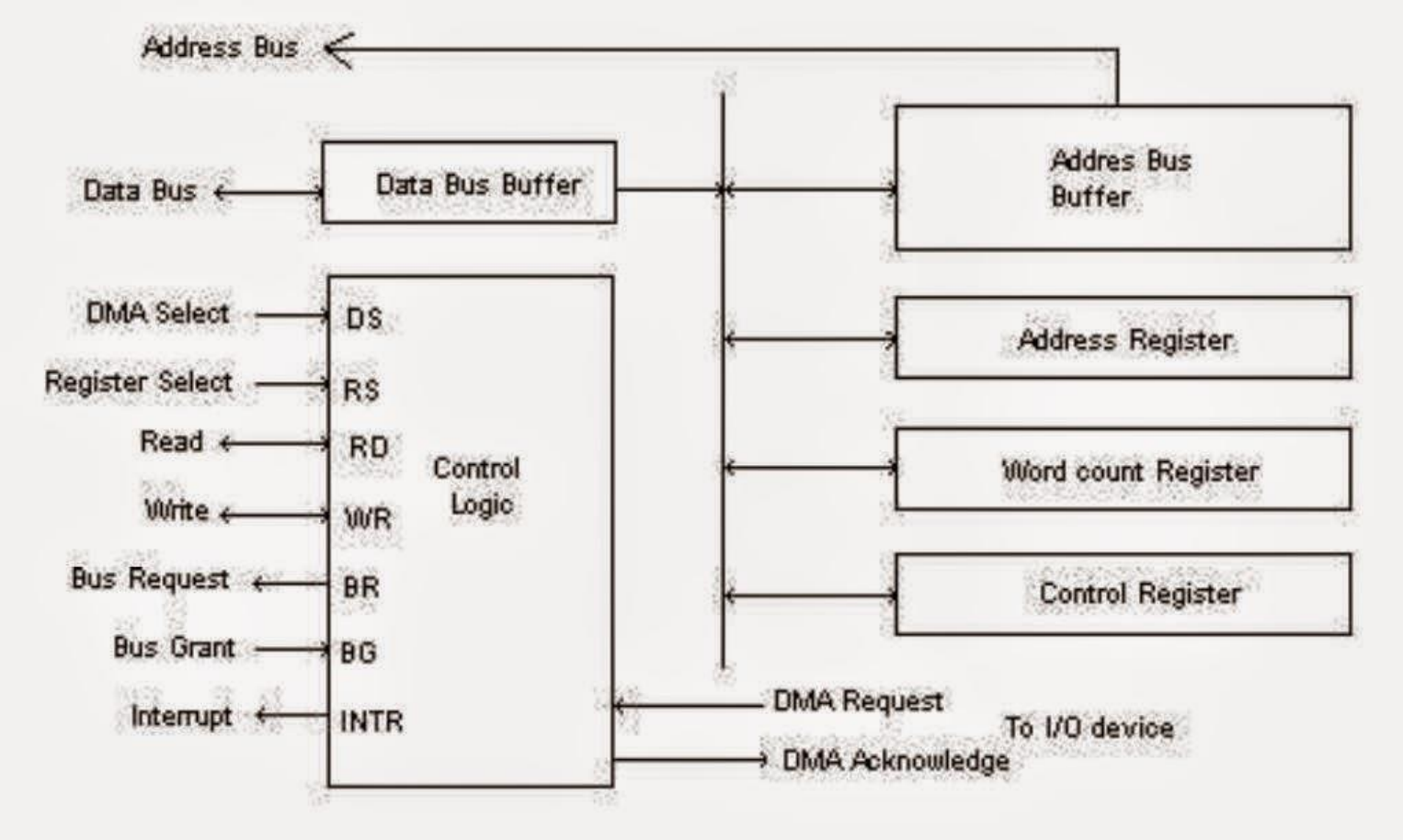
Figure 11-16 CPU bus signals for DMA transfer.



When DMA takes control of bus system, the transfer with memory can be made in the following ways:

- **Burst transfer mode:** A block sequence consisting of a number of memory words us transferred in continuous burst. Needed for fast devices such as magnetic disks where data transmission can not be stopped or slower down until whole block is transferred.
- **Cycle stealing:** This allows DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow DMA to steal one memory cycle.

DMA Controller



The DMA controller needs an usual circuit of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register and a set of address lines. The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred.

The unit communicates with CPU through 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 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 releases the bus and DMA can directly communicate 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 acknowledged lines by using a prescribed handshaking method.

The registers in DMA controller are:

a) Address register: It contains an address to specify the desired location in memory. The address register is incremented after each word that is transferred to memory.

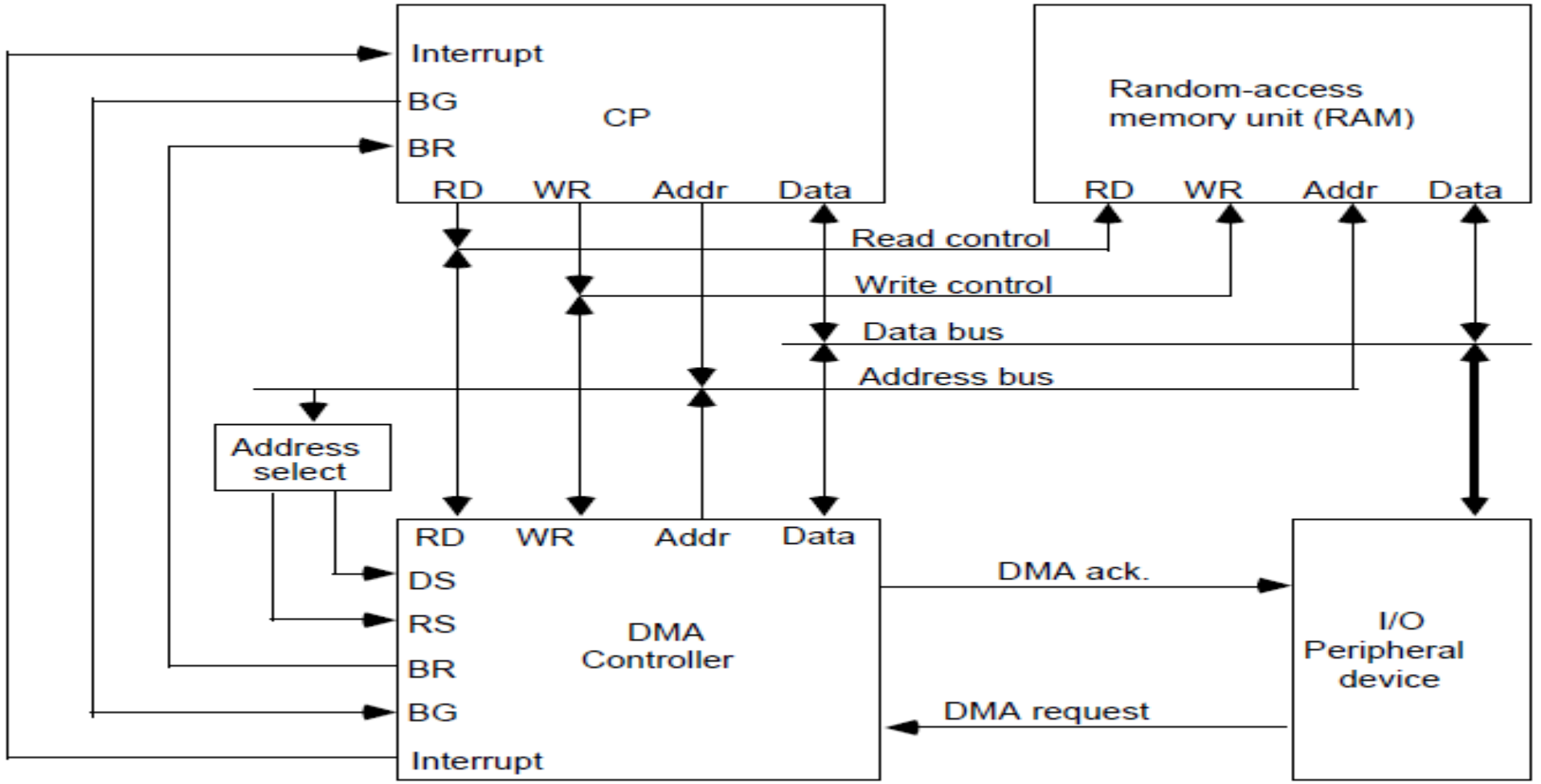
b) Word count register: It holds the number of words to be transferred. This register is decremented by 1 after each word transfer and internally tested for 0.

c) Control register: It specifies the mode of transfer.

The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until the entire block is transferred. The CPU initializes the DMA by sending the following information through the data bus.

1. The starting address of the memory block where data are available for read or where data are to be stored for write.
2. The word count which is the number of words in the memory block.
3. Control to specify the mode of transfer such as read or write.
4. A control to start the DMA transfer.

DMA Transfer



The CPU communicates with the DMA through the address and data bus. The DMA has its own address which activates the DS(DMA select) and RS (Register Select) lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.

When the peripheral devices sends a DMA request, the DMA controller activates the BR line informing the CPU to release the buses. The CPU responds with the BG line informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal and sends a DMA acknowledge to the peripheral device.

The RD and WR lines in the DMA controller are bidirectional. The direction of transfer depends on the status of BG line. When BG=0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers. When BG=1, RD and WR are output lines from the DMA controller to the RAM to specify the read or write operation for the data.

When the peripheral device receives a DMA acknowledge, it puts a word in the data bus for write or receives a word from the data bus for read. The DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.

CPU executes instruction to

- Load Memory Address Register
- Load Word Counter
- Load Function(Read or Write) to be performed
- Issue a GO (BG) command

Upon receiving a GO Command DMA performs I/O operation as follows independently from CPU

Input

– Send read control signal to Input Device

– DMA controller collects the input from input device byte by bye and assembles the byte into a word until word is full

– Send write control signal to memory

– Increment address register (Address Reg <- Address Reg +1)

– Decrement word count (WC <- WC – 1)

– If WC = 0, then Interrupt to acknowledge done, else repeat same process

Output

– Send read control signal to memory

– Read data from memory

– Increment address register (Address Reg <- Address Reg +1)

– Decrement word count (WC <- WC – 1)

– Disassemble the word

– Transfer data to the output device byte by byte

– If WC = 0, then Interrupt to acknowledge done, else repeat same process

I/O Processor (IOP)

I/O processor is a processor with DMA capability that communicates with I/O devices. In this configuration, the computer system can be divided into a memory unit, and a number of processors comprised of CPU and one or more IOPs.

IOP is similar to CPU expect that it is designed to handle the details of I/O processing. Unlike DMA controller which is set up by the CPU, IOP can fetch and execute its own instructions. IOP instructions are designed specifically to facilitate I/O transfers. In addition, IOP can perform other processing tasks such as arithmetic, logic, branching and code translation.

The block diagram of a computer with two processor is shown in figure. The memory unit occupies a central position and can communicate with each processor by means of direct memory access. The CPU is responsible for processing data needed in the solution of the computational tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit. The CPU is usually assigned the task of initiating the I/O program. From then, the IOP operates independent of the CPU and continues to transfer data from external devices and memory.

The data formats of peripheral devices differ from memory and CPU data formats. Communication between IOP and devices attached to it is similar to program control method of transfer. Communication with the memory is similar to direct memory access method.

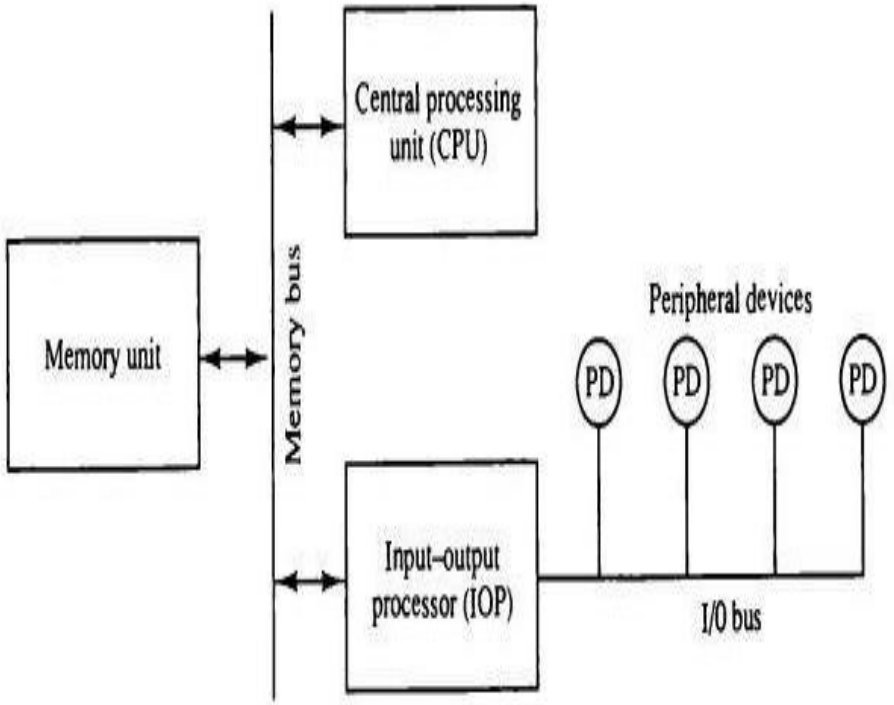
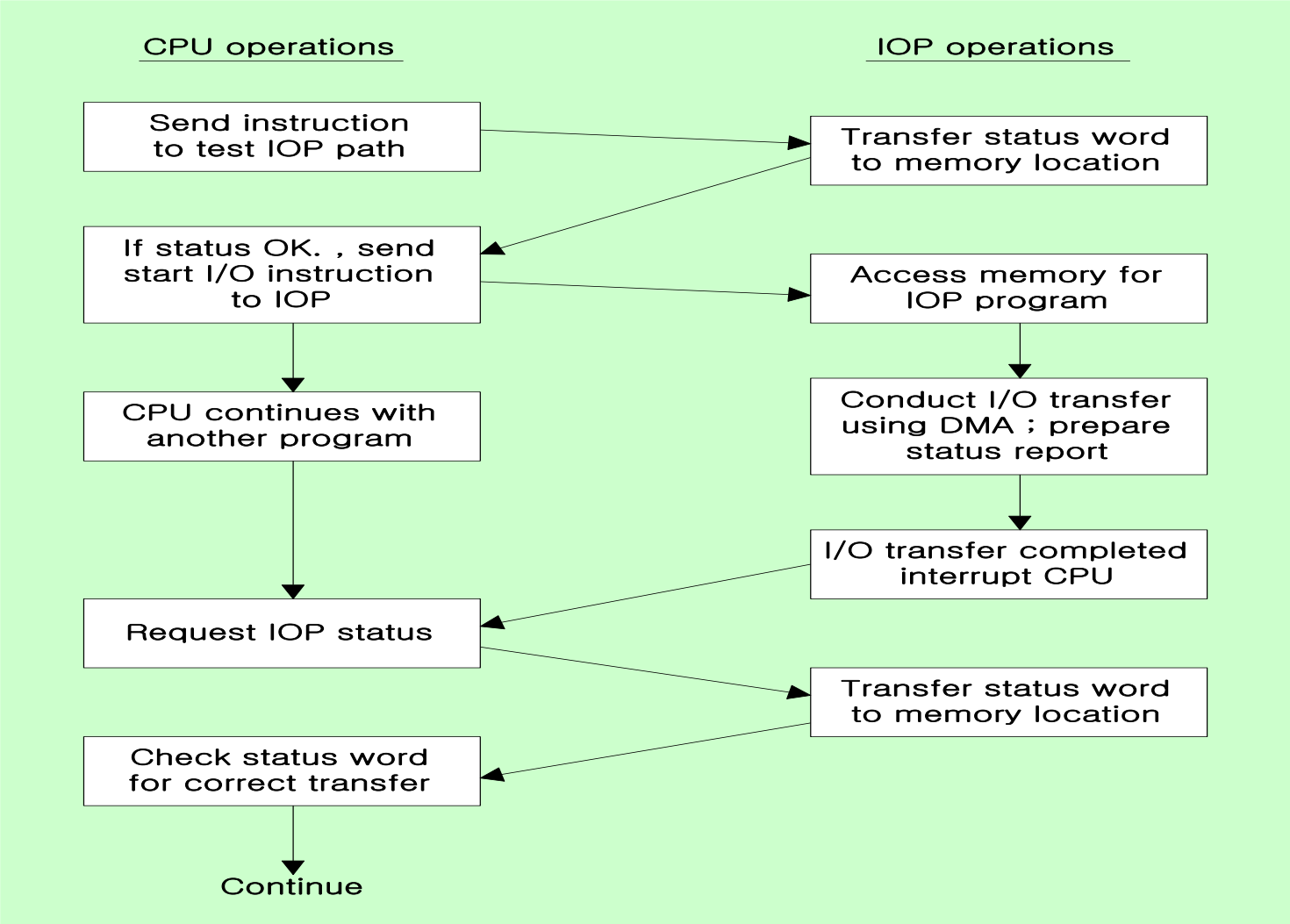


Fig: Block Diagram of computer with I/O Processor

CPU-IOP communication

- Communication between the CPU and IOP, may take different forms depending on the particular computer used. Mostly, memory unit acts as a memory center where each processor leaves information for the other.



Mechanism:

CPU sends an instruction to test the IOP path. The IOP responds by inserting a status word in memory for the CPU to check. The bits of the status word indicate the condition of IOP and I/O devices such IOP overload condition, device busy with another transfer etc. CPU then checks status word to decide what to do next. If all is in order, CPU sends the instruction to start the I/O transfer. The memory address received with this instruction tells the IOP where to find its program. CPU may continue with another program while the IOP is busy with the I/O program. When IOP terminates the transfer (using DMA), it sends an interrupt request to CPU. The CPU responds by issuing an instruction to read the status from the IOP, and IOP then answers by placing the status report into specified memory location. By inspecting the bits in the status word, CPU determines whether the I/O operation was completed satisfactorily and the process is repeated again.

Assignment: Difference Between DMA and IOP

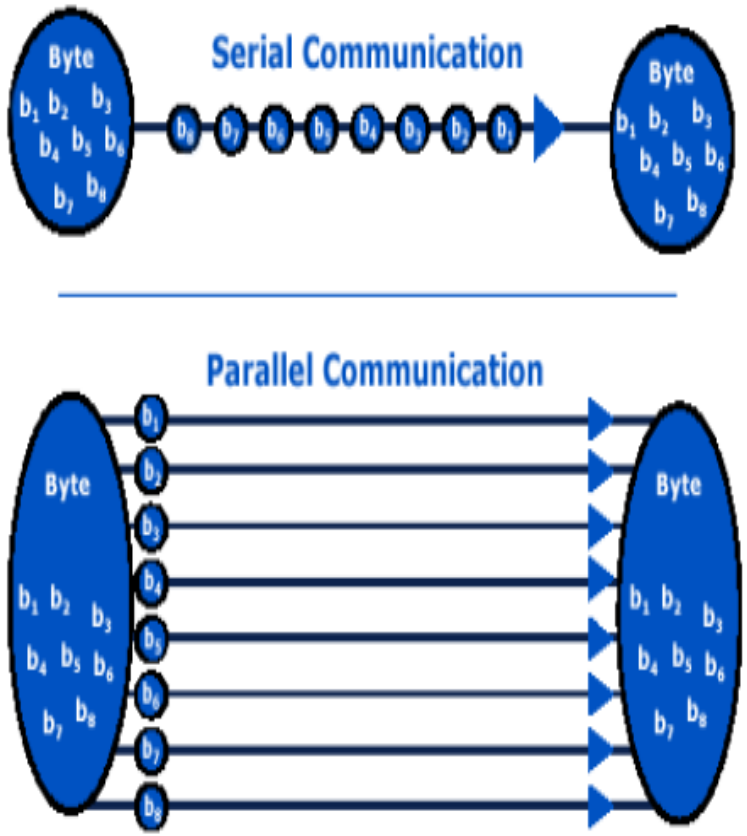
Serial and Parallel Communication

Serial Communication

- It is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus.
- Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties makes parallel communication impractical.

Parallel Communication

- It is a method of conveying multiple binary digits (bits) simultaneously. It contrasts with serial communication, which conveys only a single bit at a time; this distinction is one way of characterizing a communications link.



Data Communication Processor

- 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.
- It is a specialized I/O processor designed to communicate directly with data communication networks.
- A communication network may consist of any of a wide variety of devices, such as printers, interactive display devices, digital sensors, or a remote computing facility.
- With the use of a data communication processor, the computer can service fragments of each network demand in an interspersed manner and thus have the apparent behavior of serving many users at once. In this way the computer is able to operate efficiently in a time-sharing environment.

Modes of Data Transfer

Transmission

- Data can be transmitted between two points in three different modes: simplex, half-duplex, or full-duplex.

Simplex

- A simplex line carries information in one direction only. This mode is seldom used in data communication because the receiver cannot communicate with the transmitter to indicate the occurrence of errors. Examples of simplex transmission are radio and television broadcasting.

Half-Duplex

- A half-duplex transmission system is one that is capable of transmitting in both directions but data can be transmitted in only one direction at a time. A pair of wires is needed for this mode. A common situation is for one modem to ad as the transmitter and the other as the receiver. When transmission in one direction is completed, the role of the modems is reversed to enable transmission in the reverse direction. The time required to switch a half-duplex line from one direction to the other is called the turnaround time.

Full-Duplex

- A full-duplex transmission can send and receive data in both directions simultaneously. This can be achieved by means of a four-wire link, with a different pair of wires dedicated to each direction of transmission. Alternatively, a two-wire circuit can support full-duplex communication if the frequency spectrum is subdivided into two non-overlapping frequency bands to create separate receive and transmit channels in the same physical pair of wires.