

## 1. Define Peripherals. Explain I/O Bus and Interface Modules.

### Peripherals:

Input-output device attached to the computer are also called peripherals.

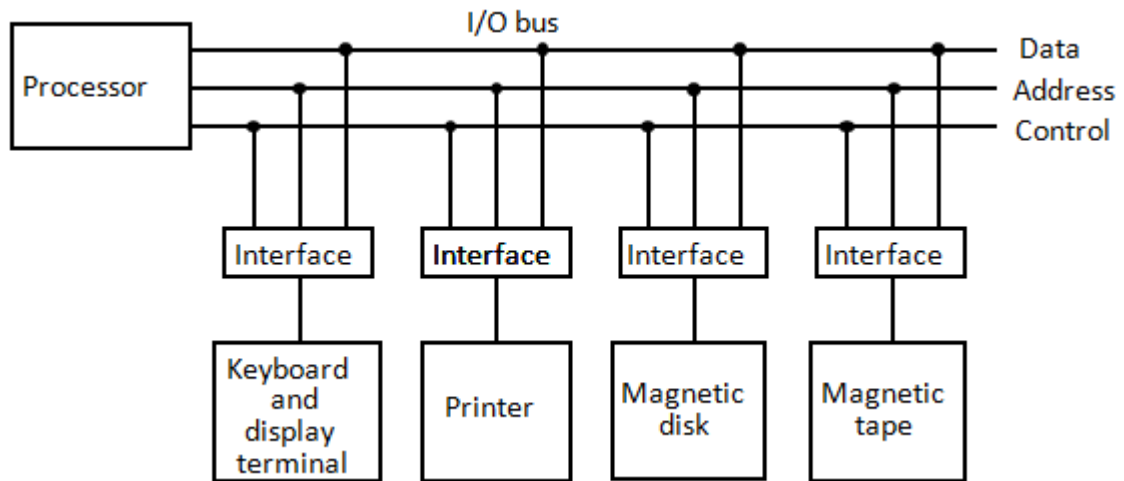


Figure 8.1: Connection of I/O bus to input-output device.

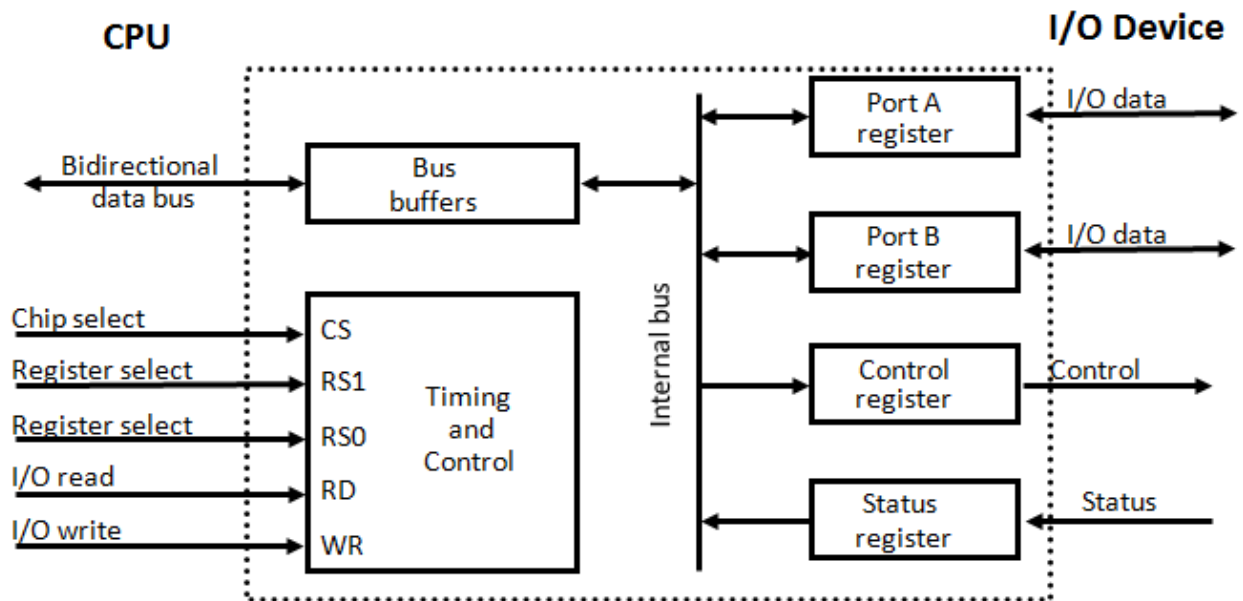
- A typical communication link between the processor and several peripherals is shown in figure 8.1.
- The I/O bus consists of data lines, address lines, and control lines.
- The magnetic disk, printer, and terminal are employed in practically any general purpose computer.
- Each peripheral device has associated with it an interface unit.
- Each interface decodes the address and control received from the I/O bus, interprets them for the peripheral, and provides signals for the peripheral controller.
- It also synchronizes the data flow and supervises the transfer between peripheral and processor.
- Each peripheral has its own controller that operates the particular electromechanical device.
- For example, the printer controller controls the paper motion, the print timing, and the selection of printing characters.
- 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 lines.
- When the interface detects its own address, it activates the path between the bus lines and the device that it controls.
- All peripherals whose address does not correspond to the address in the bus are disabled by their interface selected responds to the function code and proceeds to execute it.

- The function code is referred to as an I/O command.
- There are four types of commands that an interface may receive. They are classified as control, status, data output, and data input.
- A control command is issued to activate the peripheral and to inform it what to do.  
For example, a magnetic tape unit may be instructed to backspace the tape by one record, to rewind the tape, or to start the tape moving in the forward direction.
- A status command is used to test various status conditions in the interface and the peripheral.  
For example, the computer may wish to check the status of the peripheral before a transfer is initiated.
- During the transfer, one or more errors may occur which are detected by the interface.
- These errors are designated by setting bits in a status register that the processor can read at certain intervals.
- A data output command causes the interface to respond by transferring data from the bus into one of its registers.
- The computer starts the tape moving by issuing a control command.
- The processor then monitors the status of the tape by means of a status command.
- When the tape is in the correct position, the processor issues a data output command.
- The interface responds to the address and command and transfers the information from the data lines in the bus to its buffer register.
- The interface that communicates with the tape controller and sends the data to be stored on tape.
- 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.
- The processor checks if data are available by means of a status command and then issues a data input command.
- The interface places the data on the data lines, where they are accepted by the processor.

## 2. Explain I/O interface with example.

- An example of an I/O interface units is shown in block diagram from in figure 8.2.
- It consists of two data registers called ports, a control register, a status register, bus buffers, and timing and control circuit.
- The interface communicates with the CPU through the data bus.
- The chip select and register select inputs determine the address assigned to the interface.
- The I/O read and writes are two control lines that specify an input or output, respectively.
- The four registers communicate directly with the I/O device attached to the interface.
- The I/O data to and from the device can be transferred into either port A or port B.

- If the interface is connected to a printer, it will only output data, and if it services a character reader, it will only input data.
- A magnetic disk unit transfers data in both directions but not at the same time, so the interface can use bidirectional lines.
- A command is passed to the I/O device by sending a word to the appropriate interface register.
- The control register receives control information from the CPU. By loading appropriate bits into the control register, the interface and the I/O device attached to it can be placed in a variety of operating modes.



CS	RS1	RS0	Register Selected
0	X	X	None: data bus in high impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Figure 8.2: Example of I/O interface unit

- For example, port A may be defined as an input port and port B as an output port.
- A magnetic tape unit may be instructed to rewind the tape or to start the tape moving in the forward direction.
- The bits in the status register are used for status conditions and for recording errors that may occur during the data transfer.  
For example, a status bit may indicate that port A has received a new data item from the I/O device.
- Another bit in the status register may indicate that a parity error has occurred during the transfer.

- The interface registers communicate with the CPU through the bidirectional data bus.
- The address bus selects the interface unit through the chip select and the two register select inputs.
- A circuit must be provided externally (usually, a decoder) to detect the address assigned to the interface registers.
- This circuit enables the chip select (CS) input when the interface is selected by the address bus.
- The two register select inputs RS1 and RS0 are usually connected to the two least significant lines of the address bus.
- These two inputs select one of the four registers in the interface as specified in the table accompanying the diagram.
- The content of the selected register is transfer into the CPU via the data bus when the I/O read signal is enables.
- The CPU transfers binary information into the selected register via the data bus when the I/O write input is enabled.

### 3. What do you mean by Asynchronous data transfer? Explain Strobe control in detail.

#### *Asynchronous data transfer*

Data transfer between two independent units, where internal timing in each unit is independent from the other. Such two units are said to be asynchronous to each other.

#### *Strobe Control*

- The Strobe control method of asynchronous data transfer employs a single control line to time each transfer.

#### *Source-initiated strobe for data transfer*

- The strobe may be activated by either the source or the destination unit. Figure 8.3 shows a source-initiated transfer.
- The data bus carries the binary information from source unit to the destination unit.
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus.
- The source unit first places the data on the data bus.
- After a 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.

### Source-Initiated Strobe for Data Transfer

Block Diagram

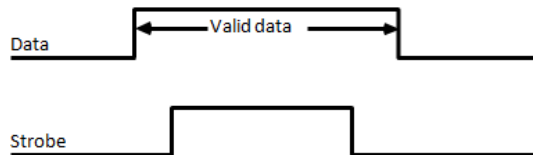
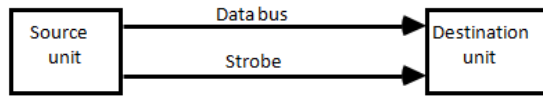
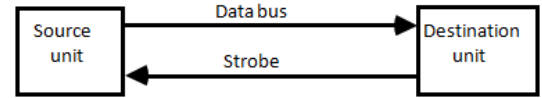


Figure 8.3: Source-initiated strobe for data transfer

### Destination-Initiated Strobe for Data Transfer

Block Diagram



Timing Diagram

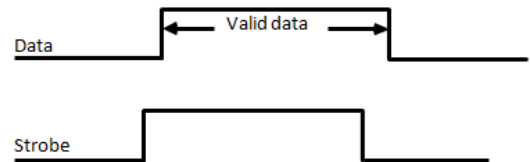


Figure 8.4: Destination-initiated strobe for data transfer

### *Destination-initiated strobe for data transfer*

- Figure 8.4 shows a data transfer initiated by the destination unit. 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 data must be valid and remain in the bus long enough for the destination unit to accept it.
- 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.
- The transfer of data between the CPU and an interface unit is similar to the strobe transfer just described.

### *Disadvantage of Strobe method:*

- 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 actually received the data item that was placed in the bus
- Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

#### 4. Explain Asynchronous data transfer with Handshaking method.

- The handshake method solves the problem of Strobe method by introducing a second control signal that provides a reply to the unit that initiates the transfer.

##### *Source-initiated transfer using 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.

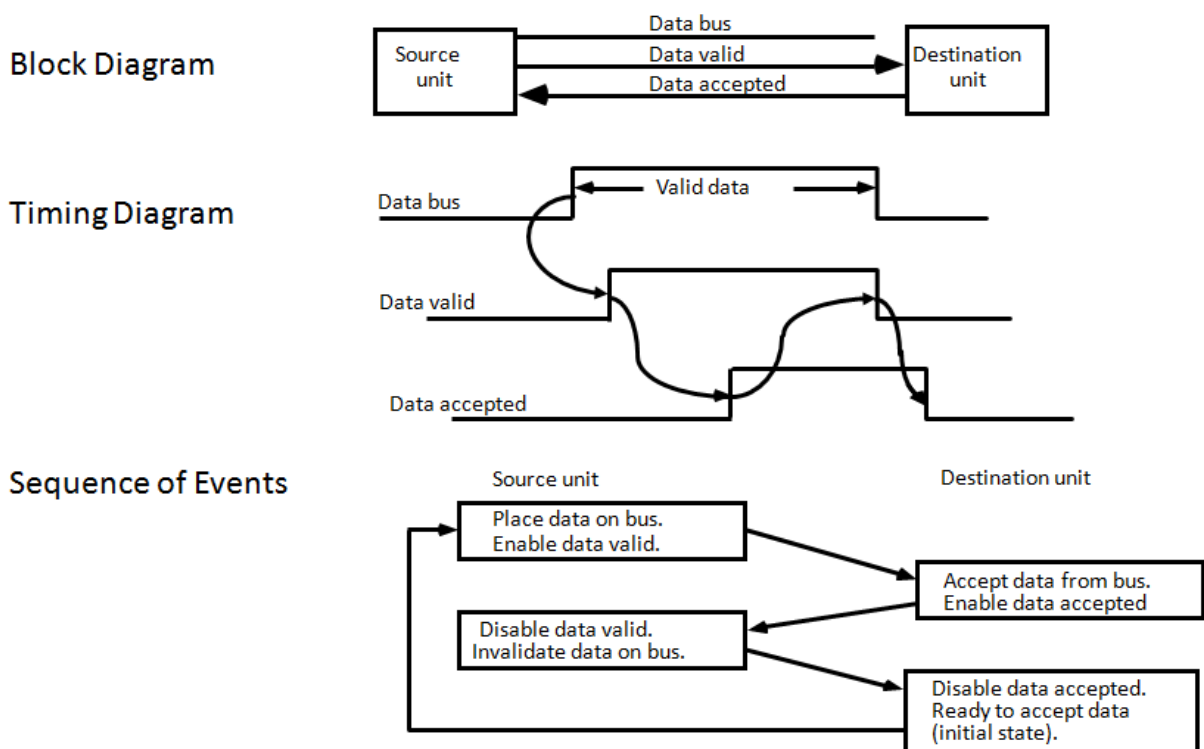


Figure 8.5: Source-initiated transfer using handshaking

- The other control line is in the other direction from the destination to the source.
- It is used by the destination unit to inform the source whether it can accept data.
- The sequence of control during the transfer depends on the unit that initiates the transfer.
- Figure 8.5 shows the data transfer procedure initiated by the source.
- The two handshaking lines the 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.
- The sequence of events listed in figure 8.5 shows the four possible states that the system can be at any given time.
- The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.

- The data accepted signal is activated by the destination unit after it accepts the data from the bus.
- The source unit then disables its data valid signal, which invalidates the data on the bus.
- The destination unit then disables its data accepted signal and the system goes into its initial state.
- The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.
- This scheme allows arbitrary delays from one state to the next and permits each unit to respond at its own data transfer rate.

### *Destination-initiated transfer using handshaking*

- The destination-initiated transfer using handshaking lines is shown in figure 8.6.
- Note that the name of the signal generated by the destination unit has been changed to ready for data to reflect its new meaning.
- The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.
- From there on, the handshaking procedure follows the same pattern as in the source-initiated case.
- Note that the sequence of events in both cases would be identical if we consider the ready for data signal as the complement of data accepted.
- In fact, the only difference between the source-initiated and the destination-initiated transfer is in their choice of initial state.

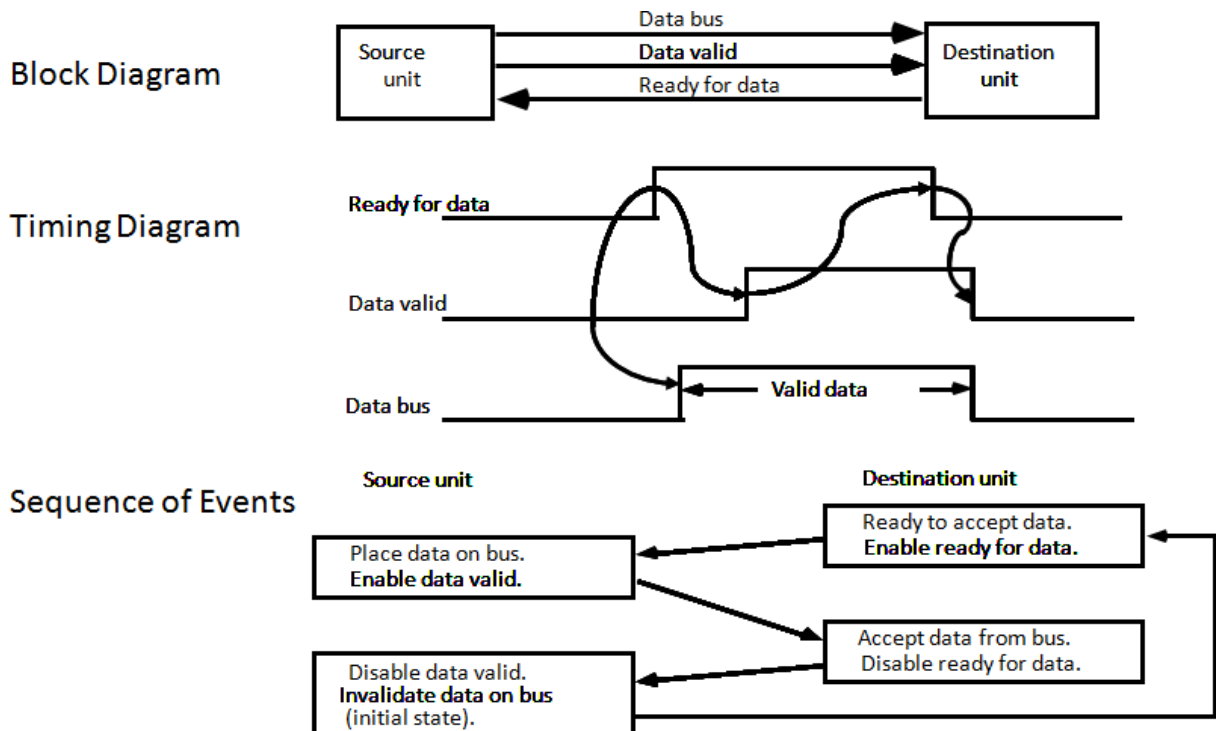


Figure 8.6: Destination-initiated transfer using handshaking

## 5. Explain Programmed I/O with example.

### *Programmed I/O:*

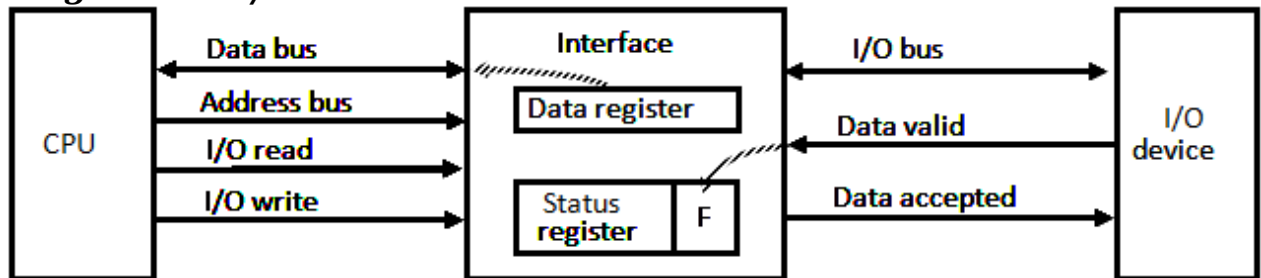


Figure 8.7: Data transfer from I/O device to CPU

- In the programmed I/O method, the I/O device does not have direct access to memory.
- An example of data transfer from an I/O device through an interface into the CPU is shown in figure 8.7.
- 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.
- 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.
- This is done by reading the status register into a CPU register and checking the value of the flag bit.
- Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

### *Example of Programmed I/O:*

- A flowchart of the program that must be written for the CPU is shown in figure 8.8.
- 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.
- 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.



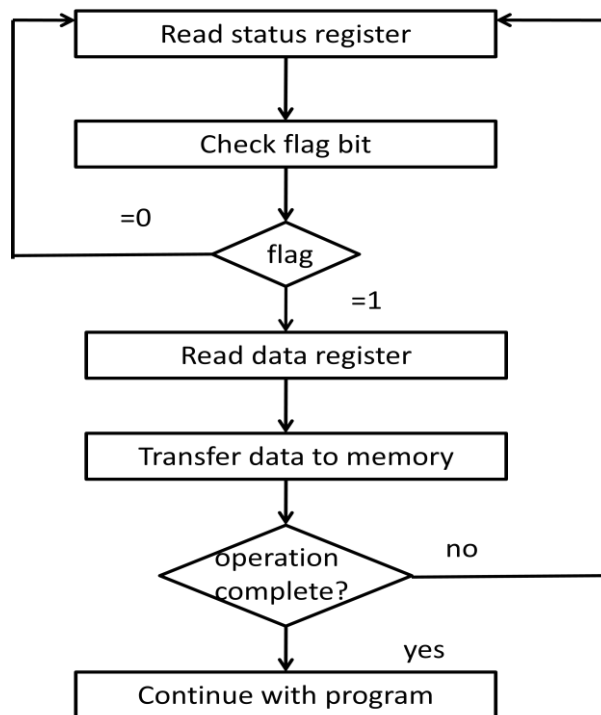


Figure 8.8: Flowchart for CPU program to input data

## 6. Write a note on Interrupt Initiated I/O

- In programmed initiated, 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.
- It can be avoided by using an interrupt facility and a special command to inform the interface to issue an interrupt request signal when data are available from the device.
- In the meantime CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device.
- When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- While the CPU is running a program, it does not check the flag. However, 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 deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.
- 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 way that the processor chooses the branch address of the service routine varies from one unit to another.
- In **non-vectored interrupt**, branch address is assigned to a fixed location in memory.
- In a **vectored interrupt**, the source that interrupts supplies the branch information to the computer. The information is called vector interrupt.
- 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.

## 7. What is priority interrupt? Explain Daisy Chaining.

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

### *Daisy Chaining Priority*

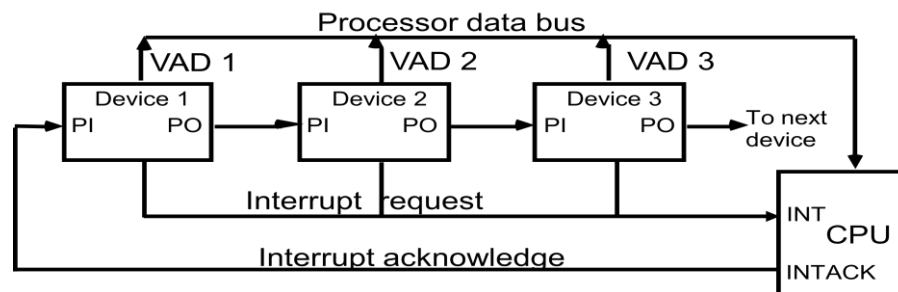


Figure 8.9: Daisy-chain priority interrupt

- 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 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.
- This method of connection between three devices and the CPU is shown in figure 8.9.
- If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU.
- When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU.
- The CPU responds to an interrupt request by enabling the interrupt acknowledge line.
- This 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.

- 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 its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked.
- A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output.
- If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output.
- Thus 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.
- The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU.
- The farther the device is from the first position; the lower is its priority.

## 8. Write a detailed note on Direct Memory Access (DMA).

### *Direct Memory Access*

- Transfer of data under programmed I/O is between CPU and peripheral.
- In direct memory access (DMA), Interface transfers data into and out of memory 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, DMA transfers the data directly into memory.

### *DMA controller*

- DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks
- CPU initializes DMA Controller by sending memory address and the block size (number of words).

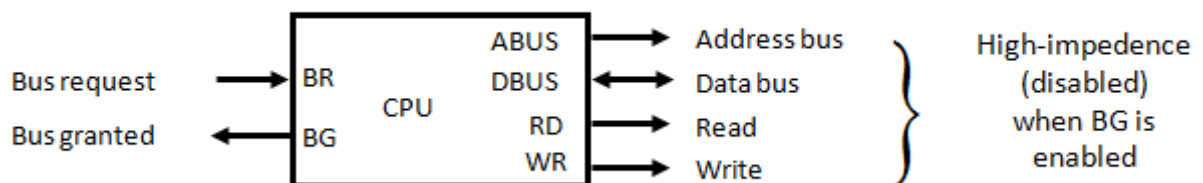


Figure 8.10: CPU bus signals for DMA transfer

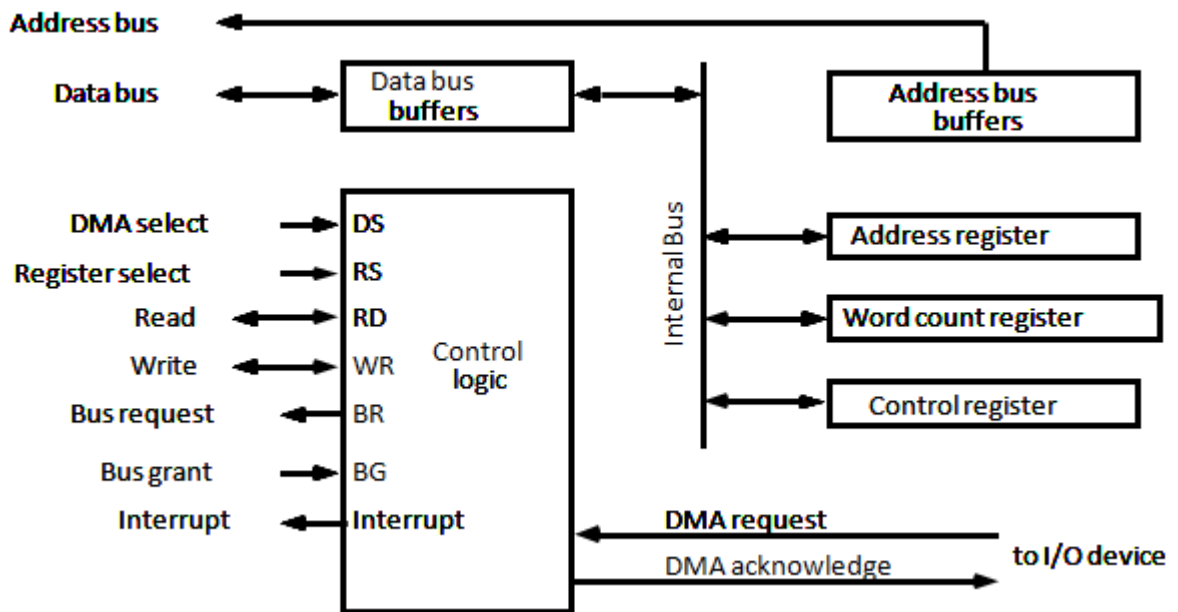


Figure 8.11: Block diagram of DMA controller

- The DMA controller needs the usual circuits 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 data transfer may be done directly between the device and memory under control of the DMA.
- Figure 8.11 shows the block diagram of a typical DMA controller.
- The unit communicates with the CPU via the data bus and control lines.
- The register 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 the buses and the DMA can communicate directly with the memory by specifying an address in the address but 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.
- 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 word count register holds the number of words to be transferred.

- This register is decremented by one after each word transfer and internally tested for zero.
- 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 register 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 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. The starting address is stored in the address register.

## 9. Explain Input- Output Processor (IOP)

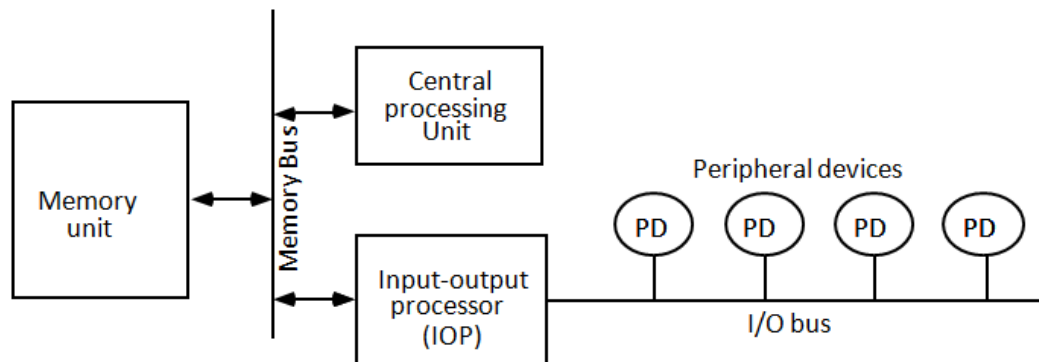


Figure 8.12: Block diagram of a computer with I/O processor

- IOP is similar to a CPU except that it is designed to handle the details of I/O processing.
- Unlike the DMA controller that must be setup entirely by the CPU, the IOP can fetch and execute its own instruction.
- IOP instructions are specifically designed 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 processors is shown in figure 8.12.
- The memory unit occupies 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 computational tasks.
- The IOP provides a path of for transfer of data between various peripheral devices and memory unit.

- The CPU is usually assigned the task of initiating the I/O program.
- From then, 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. The IOP must structure data words from many different sources.  
For example, 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.
- Similarly, an output word transferred from memory to the IOP is directed from the IOP to the output word transferred from memory to the IOP.
- 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.
- CPU instructions provide operations to start an I/O transfer and also to test I/O status conditions needed for making decisions on various I/O activities.
- The IOP, in turn, typically asks for CPU attention by means of an interrupt.
- Instructions that are read from memory by an IOP are sometimes called commands, to distinguish them from instructions that are read by the CPU.

### 10. Explain CPU-IOP Communication.

- The communication between CPU and IOP may take different forms, depending on the particular computer considered.
- In most cases the memory unit acts.
- The sequence of operations may be carried out as shown in the flowchart of figure 8.13.
- The CPU sends an instruction to test the IOP path.
- The IOP responds by inserting a status word in memory for the CPU to check.
- The bits of the status word indicate the condition of the IOP and I/O device, such as IOP overload condition, device busy with another transfer, or device ready for I/O transfer.
- The CPU refers to the status word in memory to device what do next.
- If all is in order, the CPU sends the instruction to start I/O transfer.
- The memory address received with this instruction tells the IOP where to find its program.
- The CPU can now continue with another program while the IOP is busy with the I/O program.
- Both programs refer to memory by means of DMA transfer.
- When the IOP terminates the execution of its program, it sends an interrupt request to the CPU.
- The CPU responds to the interrupt by issuing an instruction to read the status from the IOP.

- The IOP responds by placing the contents of its status report into a specified memory location.
- The status word indicates whether the transfer has been completed or if any errors occurred during the transfer.
- From inspection of the bits in the status word, the CPU determines if the I/O operation was completed satisfactorily without errors.
- The IOP takes care of all data transfers between several I/O units and the memory while the CPU is processing another program.
- The IOP and CPU are competing for the use of memory, so the number of devices that can be in operation is limited by the access time of the memory.

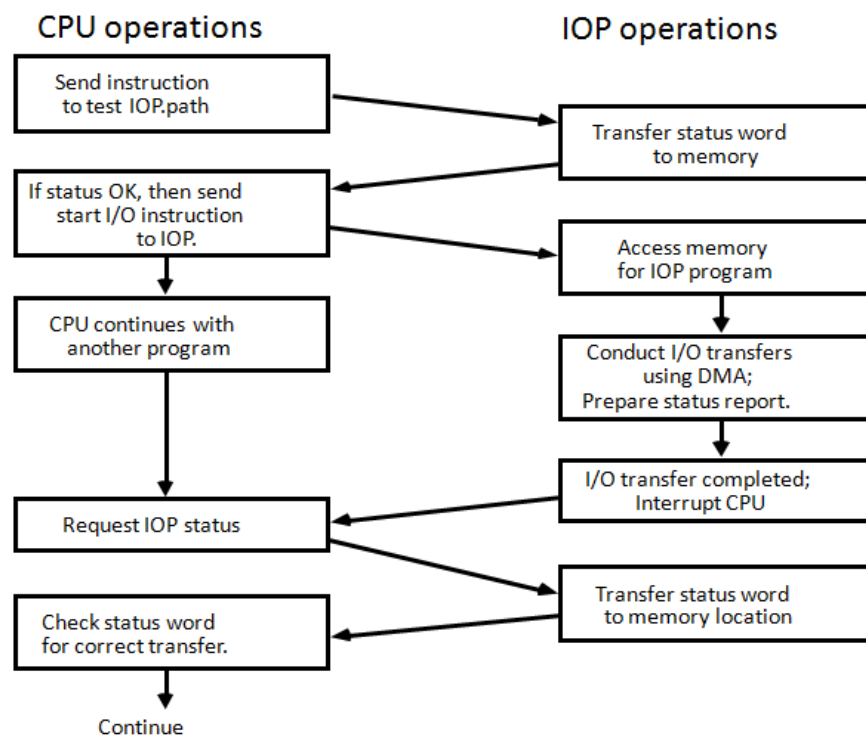


Figure 8.13: CPU-IOP communication