Input Output Organization

Peripheral devices

- In addition to the processor and a set of memory modules, the third key element of a computer system is a set of input-output subsystem referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.
- Programs and data must be entered into computer memory for processing and results obtained from computations must be recorded or displayed for the user.
- Devices that are under the direct control of the computer are said to be connected on-line. These devices are designed to read information into or out of the memory unit upon command from CPU.
- Input or output devices attached to the computer are also called peripherals.
- Among the most common peripherals are keyboards, display units, and printers.
- Perhaps those provide auxiliary storage for the systems are magnetic disks and tapes.
- Peripherals are electromechanical and electromagnetic devices of some complexity.
- We can broadly classify peripheral devices into three categories:
    - **Human Readable**: Communicating with the computer users, e.g. video display terminal, printers etc.
    - **Machine Readable**: Communicating with equipment, e.g. magnetic disk, magnetic tape, sensor, actuators used in robotics etc.
    - **Communication**: Communicating with remote devices means exchanging data with that, e.g. modem, NIC (network interface Card) etc.

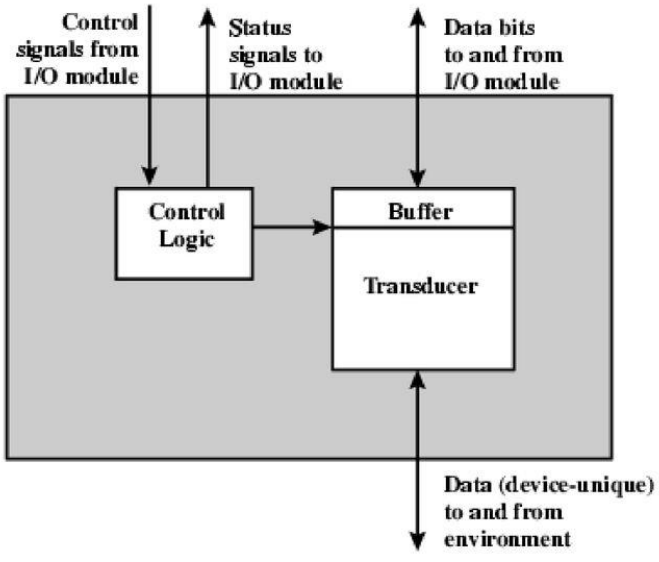


Fig: Block diagram of peripheral device

- Control signals determine the function that the device will perform such as send data to I/O module, accept data from I/O module.
- Status signals indicate the state of the device i.e. device is ready or not.
- Data bits are actual data transformation.
- Control logic associated with the device controls the device's operation in response to direction from the I/O module.
- The transducer converts data from electrical to other forms of energy during output and from other forms to electrical during input.
- Buffer is associated with the transducer to temporarily hold data being transferred between the I/O module and external devices i.e. peripheral environment.

Input Device

- Keyboard
- Optical input devices
    - Card Reader
    - Paper Tape Reader
    - Optical Character Recognition (OCR)
    - Optical Bar code reader (OBR)
    - Digitizer
    - Optical Mark Reader
- Magnetic Input Devices
    - Magnetic Stripe Reader
    - Magnetic Ink Character Recognition (MICR)
- Screen Input Devices
    - Touch Screen
    - Light Pen
    - Mouse
- Analog Input Devices

Output Device

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

I/O modules

- I/O modules interface to the system bus or central switch (CPU and Memory), interfaces and controls to one or more peripheral devices. I/O operations are accomplished through a wide assortment of external devices that provide a means of exchanging data between external environment and computer by a link to an I/O module. The link is used to exchange control status and data between I/O module and the external devices.

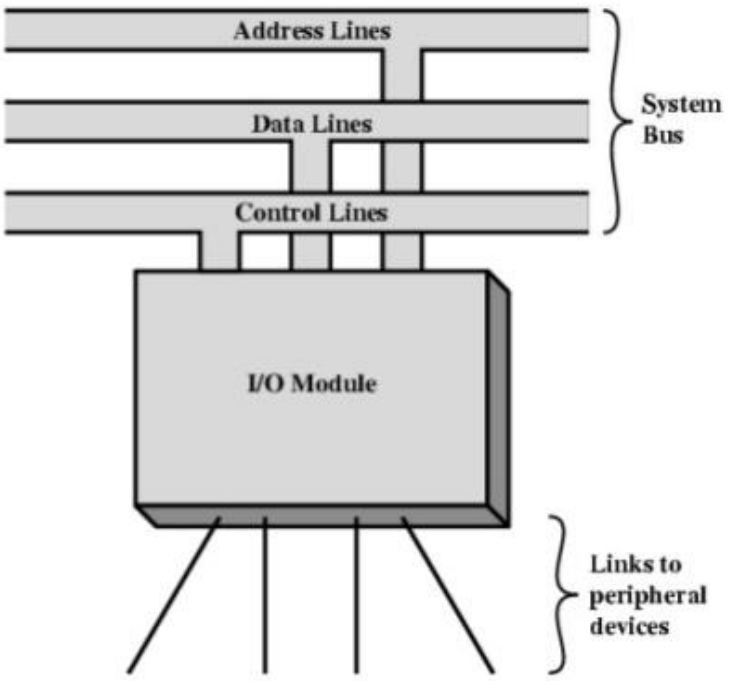


Fig: Model of I/O module

- Peripherals are not directly connected to the system bus instead an I/O module is used which contains logic for performing a communication between the peripherals and the system bus. The reasons due to which peripherals do not directly connected to the system bus are:
    - There are a wide variety of peripherals with various methods of operation. It would be impractical to incorporate the necessary logic within the processor to control a range of devices.
    - The data transfer rate of peripherals is often much slower than that of the memory or processor. Thus, it is impractical to use high speed system bus to communicate directly with a peripheral and vice versa.
    - Peripherals often use different data format and word length than the computer to which they are connected.
- Thus an I/O module is required which performs two major functions.
    - Interface to the processor and memory via the system bus
    - Interface to one or more peripherals by tailored data links

I/O Module Functions

- The I/O module is a special hardware component interface between the CPU and peripherals to supervise and synchronize all I/O transformation The detailed functions of I/O modules are;
  Control & Timing: I/O module includes control and timing to coordinate the flow of traffic between internal resources and external devices. The control of the transfer of data from external devices to processor consists following steps:
    - The processor interrogates the I/O module to check status of the attached device.
    - The I/O module returns the device status.

- If the device is operational and ready to transmit, the processor requests the transfer of data by means of a command to I/O module.
- The I/O module obtains the unit of data from the external device.
- The data are transferred from the I/O module to the processor.

Processor Communication: I/O module communicates with the processor which involves:

- Command decoding: I/O module accepts commands from the processor.
- Data: Data are exchanged between the processor and I/O module over the bus.
- Status reporting: Peripherals are too slow and it is important to know the status of I/O module.
- Address recognition: I/O module must recognize one unique address for each peripheral it controls.

Device Communication: It involves commands, status information and data.

Data Buffering: I/O module must be able to operate at both device and memory speeds. If the I/O device operates at a rate higher than the memory access rate, then the I/O module performs data buffering. If I/O devices rate slower than memory, it buffers data so as not to tie up the memory in slower transfer operation.

Error Detection: I/O module is responsible for error detection such as mechanical and electrical malfunction reported by device e.g. paper jam, bad ink track & unintentional changes to the bit pattern and transmission error.

I/O Module Structure

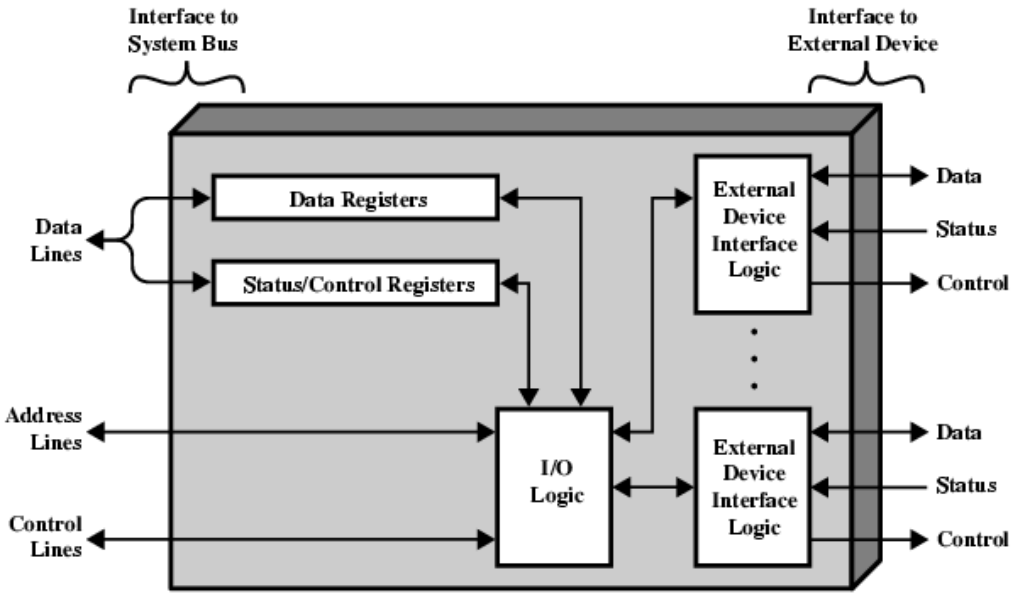


Fig: Block diagram of I/O Module

The I/O bus from the processor is attached to all peripheral interfaces

- To communicate with the particular devices, the processor places a device address on the address bus.
- Each interface contains an address decoder that monitors the address line. When the interface detects the particular device address, it activates the path between the data line and devices that it controls.

- At the same time that the address is made available in the address line, the processor provides a function code in the control way includes control command, output data and input data.

I/O Module Decisions

- Hide or reveal device properties to CPU
- Support multiple or single device
- Control device functions or leave for CPU
- Also O/S decisions
    - e.g. Unix treats everything it can as a file

Input-Output interface

- Input-Output interface provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices.
- Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.
- The communication link resolves the following *differences* between the computer and peripheral devices.

  Devices and signals
    - Peripherals - Electromechanical Devices
    - CPU or Memory - Electronic Device

  Data Transfer Rate
    - Peripherals - Usually slower
    - CPU or Memory - Usually faster than peripherals
    - Some kinds of Synchronization mechanism may be needed

  Unit of Information
    - Peripherals - Byte
    - CPU or Memory - Word

  Operating Modes
    - Peripherals - Autonomous, Asynchronous
    - CPU or Memory – Synchronous
- To resolve these differences, computer systems include special hardware components (Interfaces) between the CPU and peripherals to supervise and synchronize all input and output interfaces.

I/O Bus and Interface Modules

- The I/O bus consists of data lines, address lines and control lines.

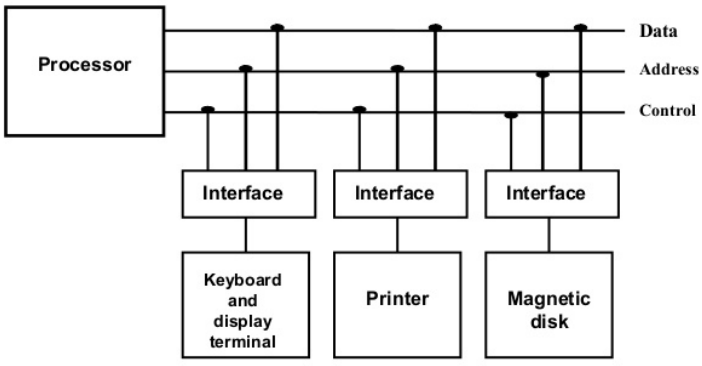


Fig: Connection of I/O bus to input-output devices

- Interface performs the following:
    - Decodes the device address (device code)
    - Decodes the commands (operation)
    - Provides signals for the peripheral controller
    - Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory
- I/O commands that the interface may receive:
    - Control command: issued to activate the peripheral and to inform it what to do.
    - Status command: used to test various status conditions in the interface and the peripheral.
    - Output data: causes the interface to respond by transferring data from the bus into one of its registers.
    - Input data: is the opposite of the data output.

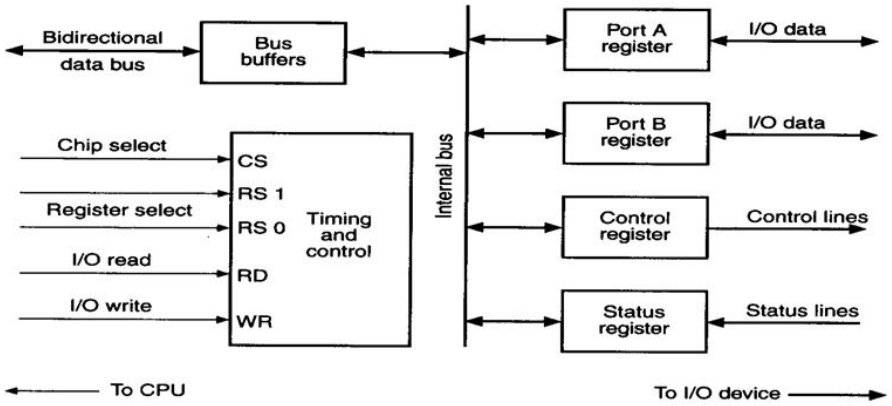
I/O versus Memory Bus

- Computer buses can be used to communicate with memory and I/O in three ways:
    - Use two separate buses, one for memory and other for I/O. In this method, all data, address and control lines would be separate for memory and I/O.
    - Use one common bus for both memory and I/O but have separate control lines. There is a separate read and write lines; I/O read and I/O write for I/O and memory read and memory write for memory.
    - Use a common bus for memory and I/O with common control line. This I/O configuration is called memory mapped.

Isolated I/O versus Memory Mapped I/O

- **Isolated I/O**
    - Separate I/O read/write control lines in addition to memory read/write control lines
    - Separate (isolated) memory and I/O address spaces
    - Distinct input and output instructions

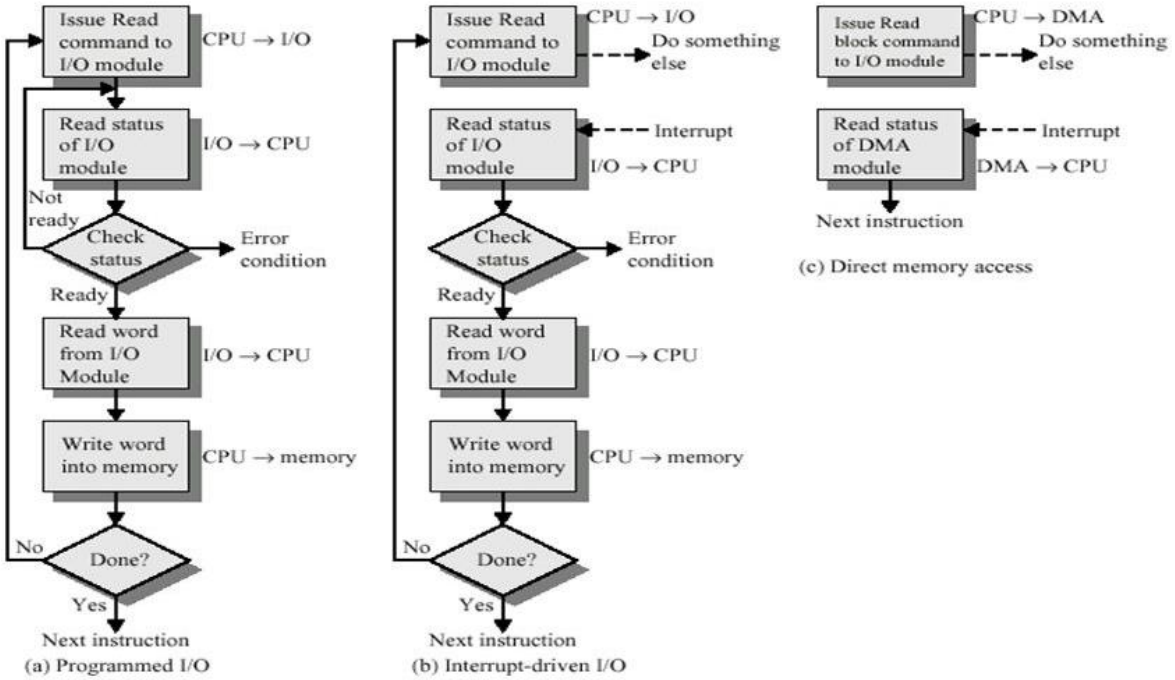
- **Memory-mapped I/O**
    - A single set of read/write control lines (no distinction between memory and I/O transfer)
    - Memory and I/O addresses share the common address space which reduces memory address range available
    - No specific input or output instruction so the same memory reference instructions can be used for I/O transfers
    - Considerable flexibility in handling I/O operations
    - 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
        - Allows CPU can define the mode of operation of each port
        - *Programmable Port*: By changing the bits in the control register, it is possible to change the interface characteristics



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

Modes of transfer

- Data Transfer between the central computer and I/O devices may be handled in a variety of modes.
- Some modes use CPU as an intermediate path, others transfer the data directly to and from the memory unit.
- Data transfer to and from peripherals may be handled in one of three possible modes.
    - Programmed I/O
    - Interrupt Driven I/O
    - Direct Memory Access (DMA)



Programmed I/O

- Programmed I/O operations are the result of I/O instructions written in the computer program.
- In programmed I/O, each data transfer in initiated by the instructions in the CPU and hence the CPU is in the continuous monitoring of the interface.
- Input instruction is used to transfer data from I/O device to CPU, store instruction is used to transfer data from CPU to memory and output instruction is used to transfer data from CPU to I/O device.
- This technique is generally used in very slow speed computer and is not a efficient method if the speed of the CPU and I/O is different.

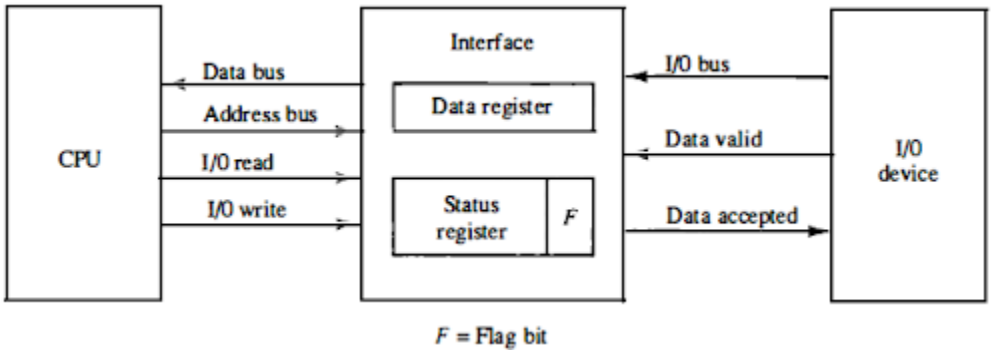


Fig: Data transfer from I/O device to CPU

- I/O device places the data on the I/O bus and enables its data valid signal
- The interface accepts the data in the data register and sets the F bit of status register and also enables the data accepted signal.
- Data valid line is disables by I/O device.
- CPU is in a continuous monitoring of the interface in which it checks the F bit of the status register.
    - If it is set i.e. 1, then the CPU reads the data from data register and sets F bit to zero
    - If it is reset i.e. 0, then the CPU remains monitoring the interface.
- Interface disables the data accepted signal and the system goes to initial state where next item of data is placed on the data bus.

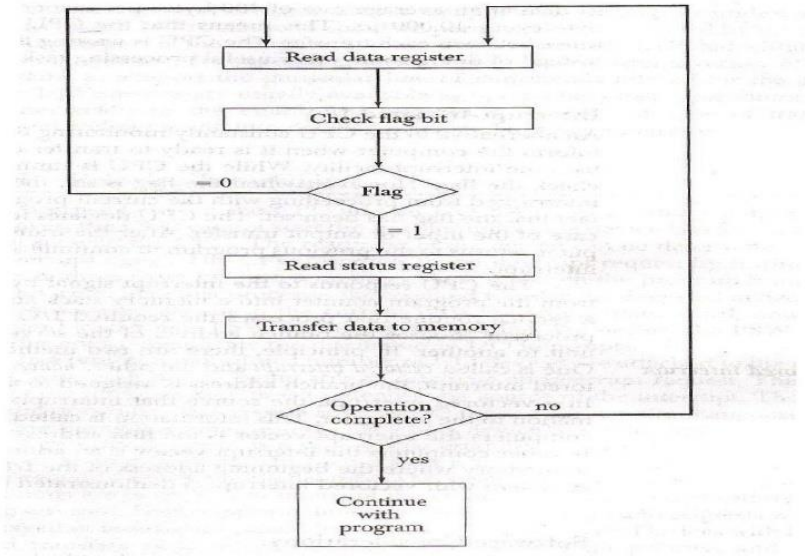


Fig: flowchart for CPU program to input data

Characteristics:

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

Polling, or polled operation, in computer science, refers to actively sampling the status of an external device by a client program as a synchronous activity. Polling is most often used in terms of input/output (I/O), and is also referred to as **polled I/O or software driven I/O.**

Interrupt-driven 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

The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded. An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work. The I/O module will then interrupt the processor to request service when it is ready to exchange data with processor. The processor then executes the data transfer, and then resumes its former processing. The interrupt can be initiated either by software or by hardware.

Interrupt Driven I/O basic operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Interrupt Processing from CPU viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
    - Save context (registers)
    - Process interrupt
    - Fetch data & store

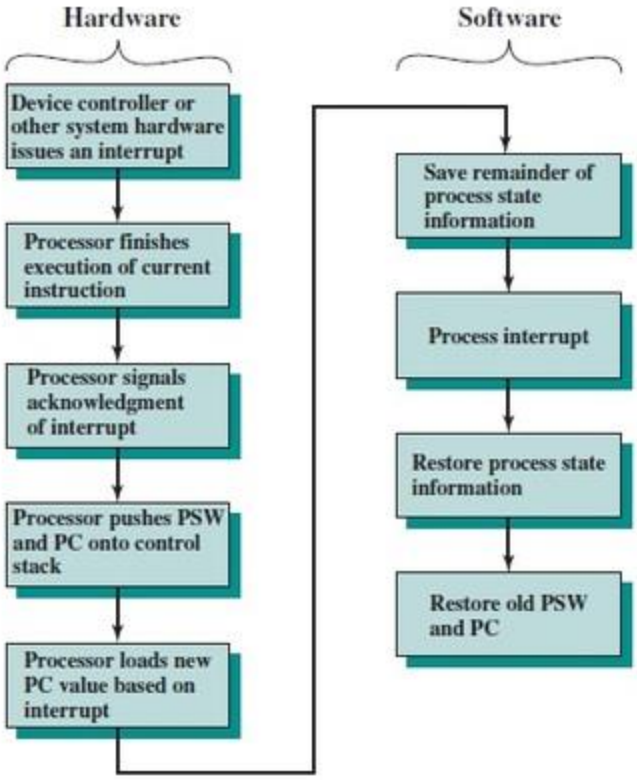


Fig: Simple Interrupt Processing

Priority Interrupt

- 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

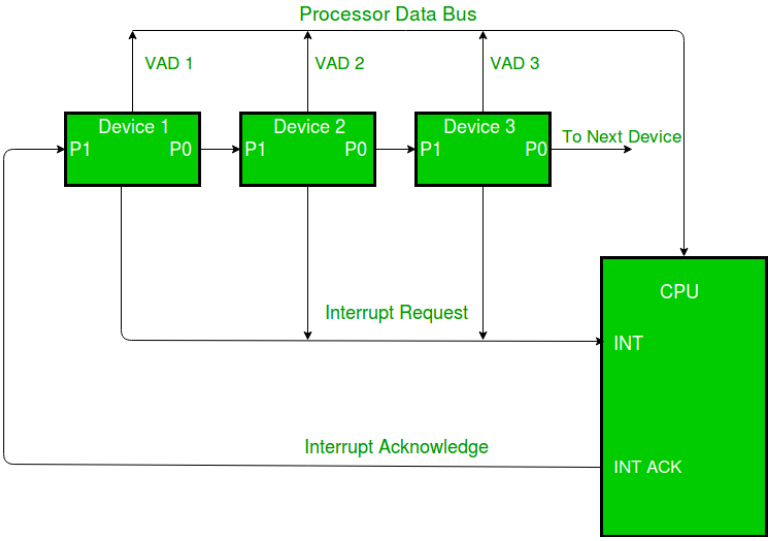
Priority Interrupt by Software (Polling)

- Priority is established by the order of polling the devices (interrupt sources), that is identify the highest-priority source by software means
- One common branch address is used for all interrupts
- Program polls the interrupt sources in sequence
- The highest-priority source is tested first
- Flexible since it is established by software
- Low cost since it needs a very little hardware
- Very slow

Priority Interrupt by Hardware

- Require a priority interrupt manager which accepts all the interrupt requests to determine the highest priority request
- Fast since identification of the highest priority interrupt request is identified by the hardware
- Fast since each interrupt source has its own interrupt vector to access directly to its own service routine

1. **Daisy Chain Priority (Serial)**



- Interrupt Request from any device
- CPU responds by INTACK
- Any device receives signal(INTACK) at PI puts the VAD on the bus
- Among interrupt requesting devices the only device which is physically closest to CPU gets INTACK and it blocks INTACK to propagate to the next device

WORKING:

There is an interrupt request line which is common to all the devices and goes into the CPU.

- When no interrupts are pending, the line is in HIGH state. But if any of the devices raises an interrupt, it places the interrupt request line in the LOW state.
- The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.
- This signal is received at the PI(Priority in) input of device 1.
- If the device has not requested the interrupt, it passes this signal to the next device through its PO(priority out) output. (PI = 1 & PO = 1)
- However, if the device had requested the interrupt, (PI =1 & PO = 0)
  - The device consumes the acknowledge signal and block its further use by placing 0 at its PO(priority out) output.
  - The device then proceeds to place its interrupt vector address(VAD) into the data bus of CPU.
  - The device puts its interrupt request signal in HIGH state to indicate its interrupt has been taken care of.

  NOTE: VAD is the address of the service routine which services that device.

- If a device gets 0 at its PI input, it generates 0 at the PO output to tell other devices that acknowledge signal has been blocked. (PI = 0 & PO = 0)

Hence, the device having PI = 1 and PO = 0 is the highest priority device that is requesting an interrupt. Therefore, by daisy chain arrangement we have ensured that the highest priority interrupt gets serviced first and have established a hierarchy. The farther a device is from the first device, the lower its priority.

2. Parallel Priority

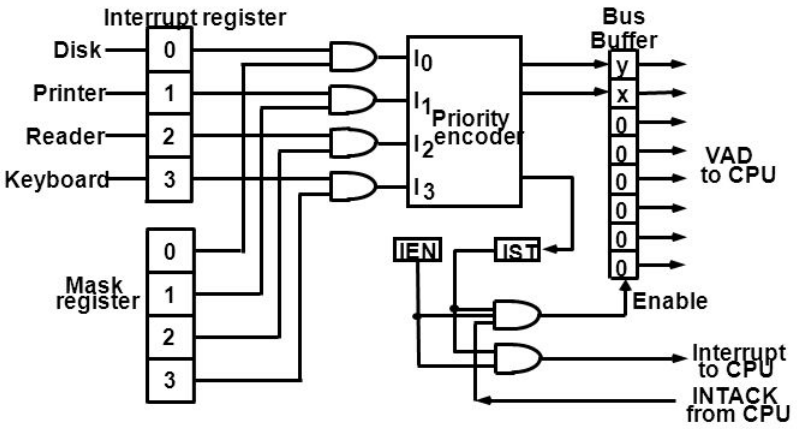


Fig: Parallel priority interrupts hardware

- IEN: Set or Clear by instructions ION or IOF
- IST: Represents an unmasked interrupt has occurred. INTACK enables tristate Bus Buffer to load VAD generated by the Priority Logic
- Interrupt Register:
  - Each bit is associated with an Interrupt Request from different Interrupt Source - different priority level
  - Each bit can be cleared by a program instruction
- Mask Register:
  - Mask Register is associated with Interrupt Register
  - Each bit can be set or cleared by an Instruction

Priority Encoder

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

Inputs				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	×	×	0
1	0	0	0	0	0	1
×	1	0	0	0	1	1
×	×	1	0	1	0	1
×	×	×	1	1	1	1

Interrupt Cycle

- At the end of each Instruction cycle
- CPU checks IEN and IST
- If IEN and IST = 1, CPU -> Interrupt Cycle
    - $SP \leftarrow SP - 1$; Decrement stack pointer
    - $M[SP] \leftarrow PC$; Push PC into stack
    - $INTACK \leftarrow 1$; Enable interrupt acknowledge
    - $PC \leftarrow VAD$; Transfer vector address to PC
    - $IEN \leftarrow 0$; Disable further interrupts
    - Go To Fetch to execute the first instruction in the interrupt service routine

Direct Memory access

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller -> Freeing CPU for other tasks

The transfer of data between the peripheral and memory without the interaction of CPU and letting the peripheral device manage the memory bus directly is termed as Direct Memory Access (DMA).

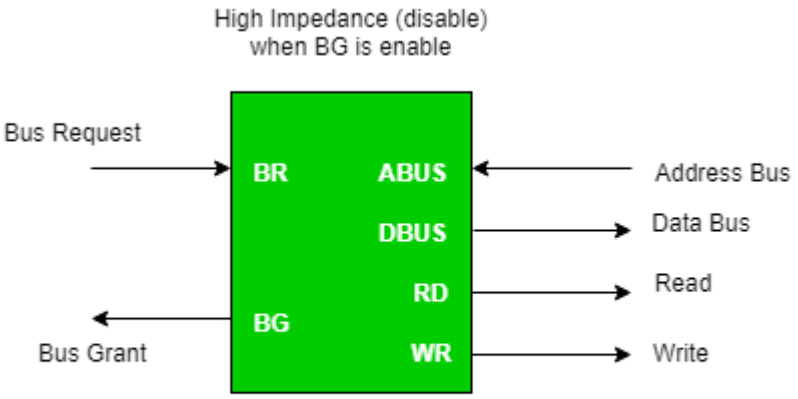


Figure - CPU Bus Signals for DMA Transfer

The two control signals Bus Request and Bus Grant are used to fascinate the DMA transfer. The bus request input is used by the DMA controller to request the CPU for the control of the buses. When BR signal is high, the CPU terminates the execution of the current instructions and then places the address, data, read and write lines to the high impedance state and sends the bus grant signal. The DMA controller now takes the control of the buses and transfers the data directly between memory and I/O without processor interaction. When the transfer is completed, the bus request signal is made low by DMA. In response to which CPU disables the bus grant and again CPU takes the control of address, data, read and write lines.

The transfer of data between the memory and I/O of course facilitates in two ways which are DMA Burst and Cycle Stealing.

DMA Burst: The block of data consisting a number of memory words is transferred at a time.

Cycle Stealing: DMA transfers one data word at a time after which it must return control of the buses to the CPU.

- CPU is usually much faster than I/O (DMA), thus CPU uses the most of the memory cycles
- DMA Controller steals the memory cycles from CPU
- For those stolen cycles, CPU remains idle
- For those slow CPU, DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

DMA Controller

The DMA controller communicates with the CPU through the data bus and control lines. DMA select signal is used for selecting the controller, the register select is for selecting the register. When the bus grant signal is zero, the CPU communicates through the data bus to read or write into the DMA register. When bus grant is one, the DMA controller takes the control of buses and transfers the data between the memory and I/O.

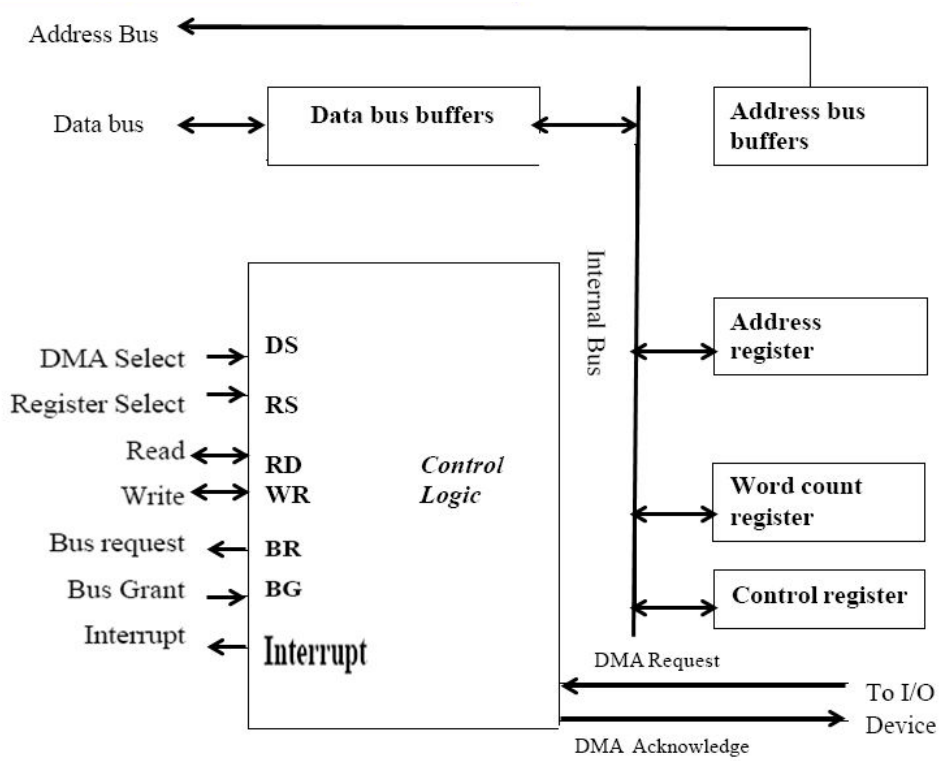


Fig: Block diagram of DMA controller

The address register specifies the desired location of the memory which is incremented after each word is transferred to the memory. The word count register holds the number of words to be transferred which is decremented after each transfer until it is zero. When it is zero, it indicates the end of transfer. After which the bus grant signal from CPU is made low and CPU returns to its normal operation. The control register specifies the mode of transfer which is Read or Write.

DMA Transfer

- DMA request signal is given from I/O device to DMA controller.
- DMA sends the bus request signal to CPU in response to which CPU disables its current instructions and initialize the DMA by sending the following information.
    - The starting address of the memory block where the data are available (for read) and where data to be stored (for write)
    - The word count which is the number of words in the memory block
    - Control to specify the mode of transfer
    - Sends a bust grant as 1 so that DMA controller can take the control of the buses
    - DMA sends the DMA acknowledge signal in response to which peripheral device puts the words in the data bus (for write) or receives a word from the data bus (for read).

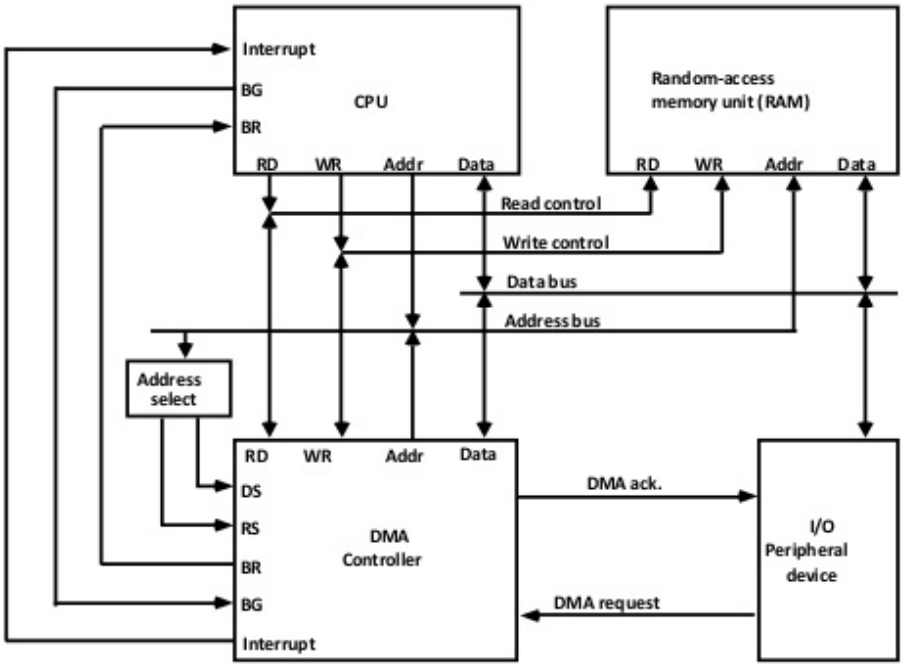


Fig: DMA transfer in a computer system

DMA Operation

- CPU tells DMA controller:-
    - Read/Write
    - Device address
    - Starting address of memory block for data
    - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

I/O Processors

- Processor with direct memory access capability that communicates with I/O devices
- Channel accesses memory by cycle stealing
- Channel can execute a Channel Program
- Stored in the main memory

- Consists of Channel Command Word(CCW)
- Each CCW specifies the parameters needed by the channel to control the I/O devices and perform data transfer operations
- CPU initiates the channel by executing a channel I/O class instruction and once initiated, channel operates independently of the CPU

A computer may incorporate one or more external processors and assign them the task of communicating directly with the I/O devices so that no each interface need to communicate with the CPU. An I/O processor (IOP) is a processor with direct memory access capability that communicates with I/O devices. IOP instructions are specifically designed to facilitate I/O transfer. The IOP can perform other processing tasks such as arithmetic logic, branching and code translation.

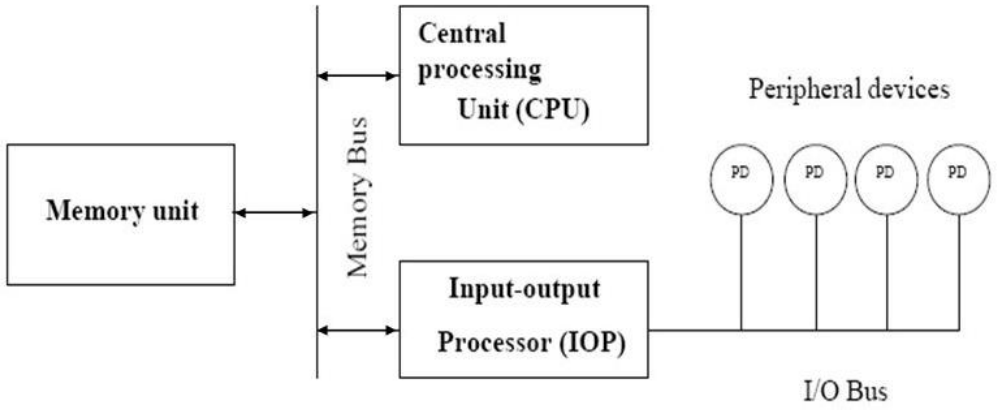


Fig: Block diagram of a computer with I/O Processor

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 computational tasks. The IOP provides a path for transferring data between various peripheral devices and memory unit.

In most computer systems, the CPU is the master while the IOP is a slave processor. The CPU initiates the IOP and after which the IOP operates independent of CPU and transfer data between the peripheral and memory. For example, the IOP receives 5 bytes from an input device at the device rate and bit capacity. After which the IOP packs them into one block of 40 bits and transfer them to memory. Similarly the O/P word transfer from memory to IOP is directed from the IOP to the O/P device at the device rate and bit capacity.

CPU – IOP Communication

The memory unit acts as a message center where each processor leaves information for the other. The operation of typical IOP is appreciated with the example by which the CPU and IOP communication.

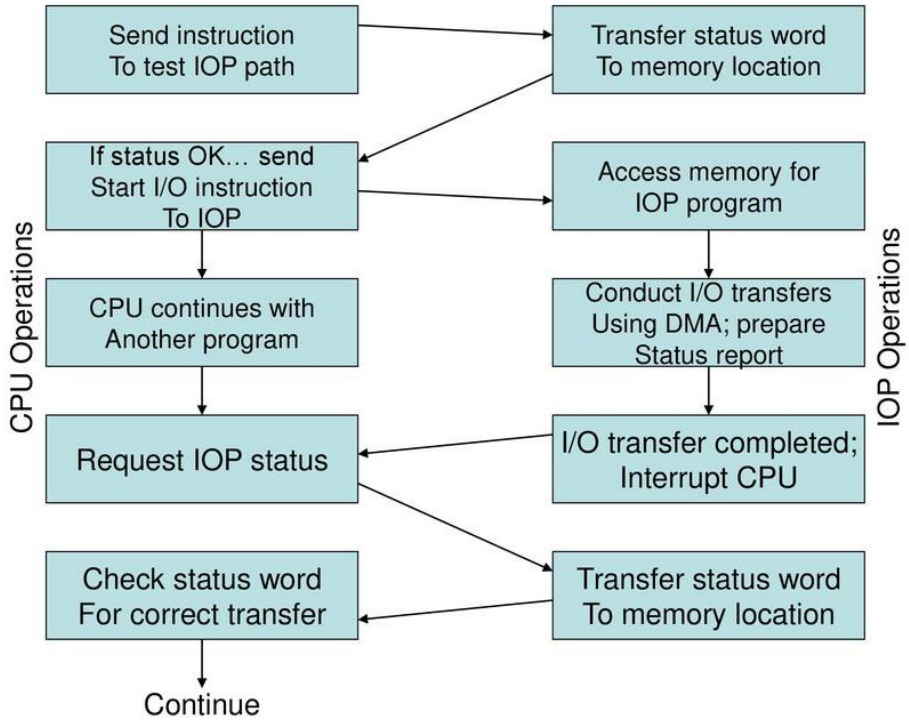


Fig: CPU-IOP communication

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 in memory to decide what to do next.
- If all right up to this, the CPU sends the instruction to start I/O transfer.
- The CPU now continues with another program while IOP is busy with I/O program.
- When IOP terminates the execution, it sends an interrupt request to CPU.
- CPU responds by issuing an instruction to read the status from the IOP.
- IOP responds by placing the contents to its status report into specified memory location.
- Status word indicates whether the transfer has been completed or with error.

Input/Output Channels

A channel is an independent hardware component that co-ordinate all I/O to a set of controllers. Computer systems that use I/O channel have special hardware components that handle all I/O operations.

Channels use separate, independent and low cost processors for its functioning which are called Channel Processors.

Channel processors are simple, but contains sufficient memory to handle all I/O tasks. When I/O transfer is complete or an error is detected, the channel controller communicates with the CPU using an interrupt, and informs CPU about the error or the task completion.

Each channel supports one or more controllers or devices. **Channel programs** contain list of commands to the channel itself and for various connected controllers or devices. Once the operating system has prepared a list of I/O commands, it executes a single I/O machine instruction to initiate the channel program, the channel then assumes control of the I/O operations until they are completed.

IBM 370 I/O Channel

The I/O processor in the IBM 370 computer is called a **Channel**. A computer system configuration includes a number of channels which are connected to one or more I/O devices.

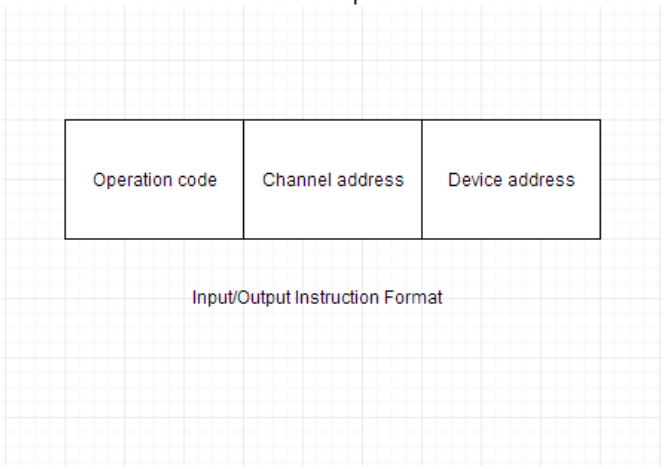
Categories of I/O Channels

Following are the different categories of I/O channels:

Multiplexer
The Multiplexer channel can be connected to a number of slow and medium speed devices. It is capable of operating number of I/O devices simultaneously.

Selector
This channel can handle only one I/O operation at a time and is used to control one high speed device at a time.

Block-Multiplexer
It combines the features of both multiplexer and selector channels.
The CPU directly can communicate with the channels through control lines. Following diagram shows the word format of channel operation.



The computer system may have number of channels and each is assigned an address. Each channel may be connected to several devices and each device is assigned an address.

External Interfaces

Universal Serial Bus (or USB) Port

- It can connect all kinds of external USB devices such as external hard disk, printer, scanner, mouse, keyboard, etc.
- It was introduced in 1997.
- Most of the computers provide two USB ports as minimum.
- Data travels at 12 megabits per seconds.
- USB compliant devices can get power from a USB port.

Firewire Port

- Transfers large amount of data at very fast speed.
- Connects camcorders and video equipment to the computer.
- Data travels at 400 to 800 megabits per seconds.
- Invented by Apple.
- It has three variants: 4-Pin FireWire 400 connector, 6-Pin FireWire 400 connector, and 9-Pin FireWire 800 connector.