Computer Architecture

V semester

Chapter 7: Input Output Organization

4 hours

Compiled by :

Ankit Bhattarai, Assistant Professor
Email: ankitbhattarai@cosmoscollege.edu.np

Topics to be discussed

- External Devices
- I/O Module
- Modes of Transfer : Programmed I/O, Interrupt Driven I/O
- DMA
- I/O Channels and I/O processors
- External Interfaces

External Device

- External devices provide the means of exchanging data between the external environment and the computer. An external device is attached to computer by a link to an ***I/O module***.
- The link is used to exchange control, status and data between the I/O module and external device.
- External device is referred to as peripheral device or peripheral. It is classified in three categories:

1. **Human readable**: suitable for communicating with computer user. E.g. Screen, Printer, Keyboard etc.
2. **Machine readable**: suitable for communicating with equipment. E.g. magnetic disk, tape, sensor, actuator etc.
3. **Communication**: suitable for communicating with remote devices. E.g. Modem, NIC etc.

External Devices are: Keyboard/ Monitor and Disk Drives.

Block Diagram of External Device

- ***Data*** are in the form of a set of bits to be sent to or received from I/O module. ***Control signals*** determine the function that the device will perform. (e.g. Input the data, output the data, determine head position etc.)
- ***Status signals*** indicate the state of the device. (e.g. acknowledge signal).
- ***Control Logic*** controls device operation according to direction from I/O module.
- ***Transducer*** converts electrical signals into other forms of energy for output and other forms to electrical signals for input.
- ***Buffer*** hold the data temporarily for transducer or from transducer.

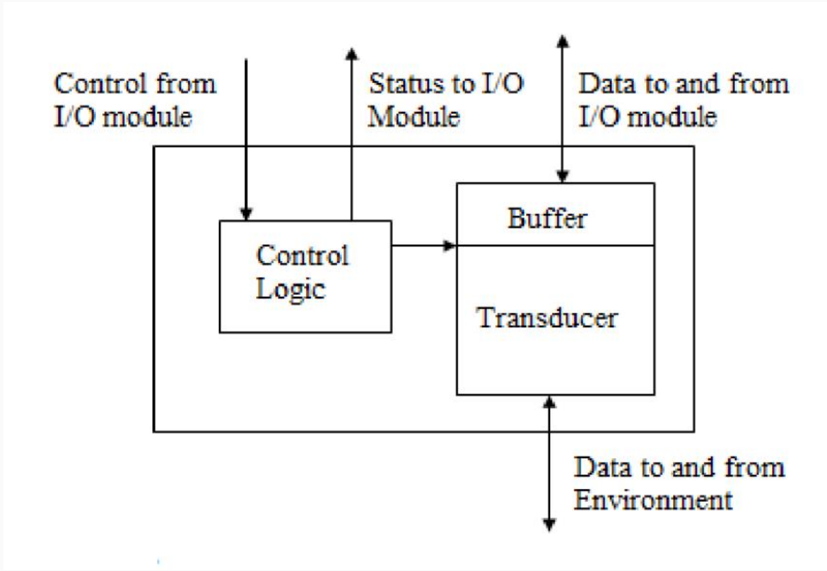
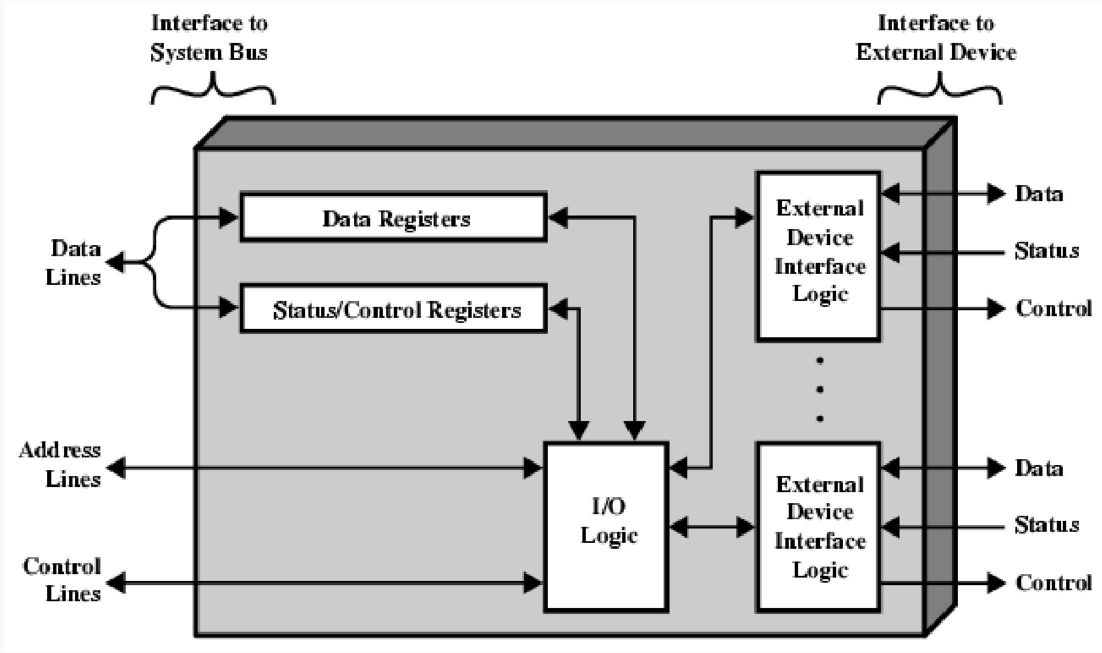


Fig. Block Diagram of External Device

I/O Module

- 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



I/O Module

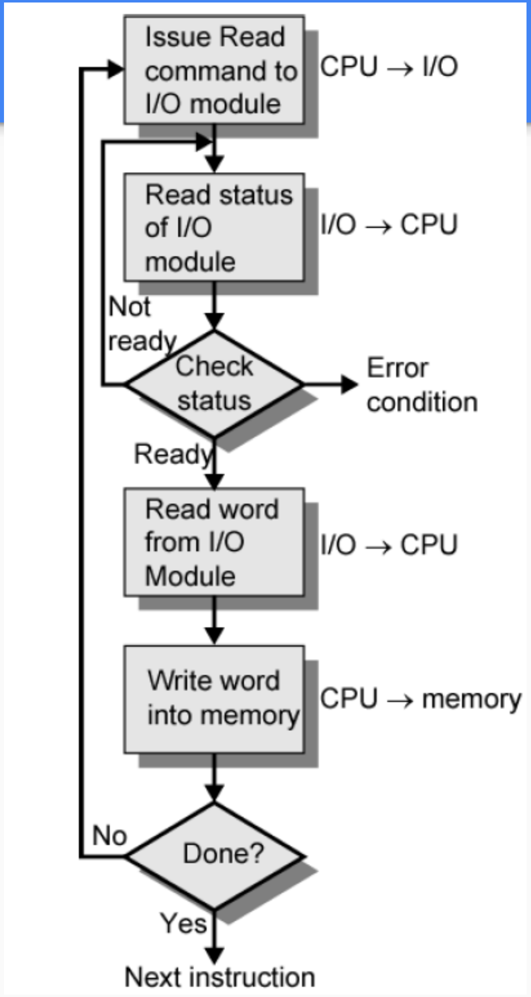
- The I/O bus from the processor is attached to all peripheral interfaces
- To communicate with the particular devices, the processor places a device address on the address bus.
- Each interface contains an address decoder that monitors the address line. When the interface detects the particular device 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 functions are control and timing, processor communication, Device communication, data buffering and error detection

I/O Module : 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)

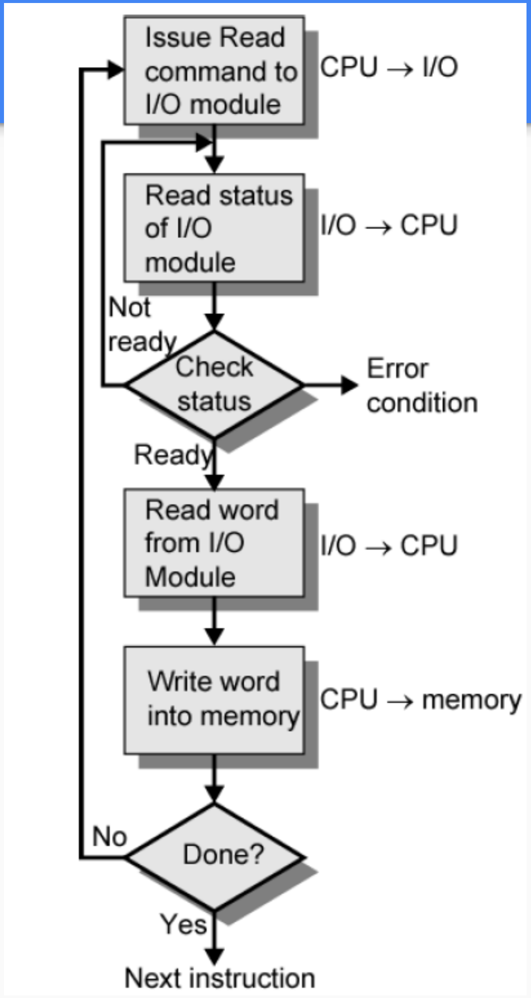
Programmed I/O

- 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 an efficient method if the speed of the CPU and I/O is different.

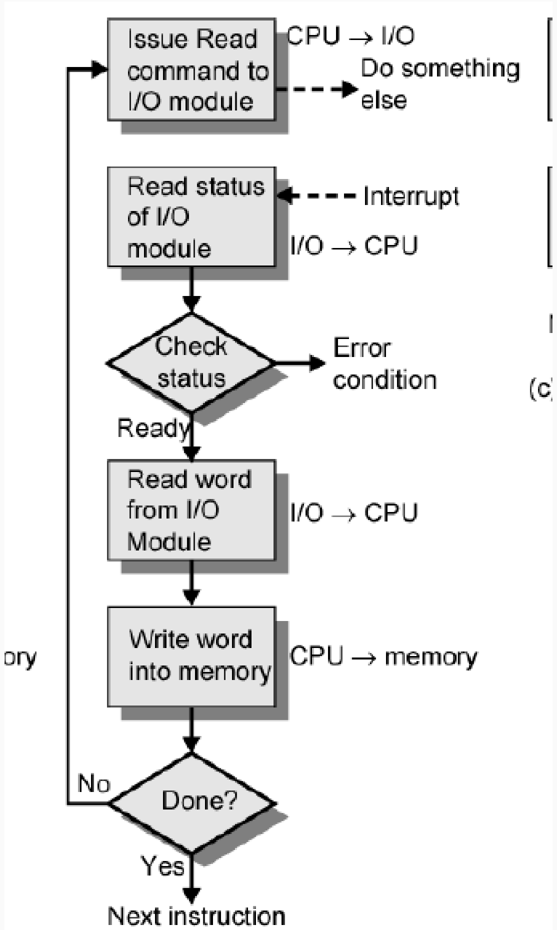


Programmed I/O: *Problem with programmed I/O*

- 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.



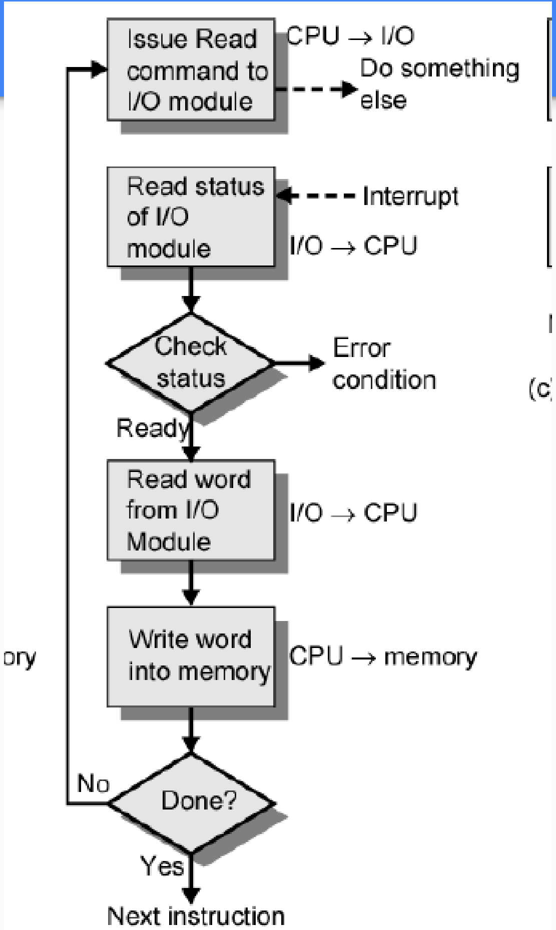
Interrupt Driven I/O



(b) Interrupt-driven I/O

Interrupt Driven I/O

- Polling takes valuable CPU time
    - (Note: **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)
- 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.



(b) Interrupt-driven I/O

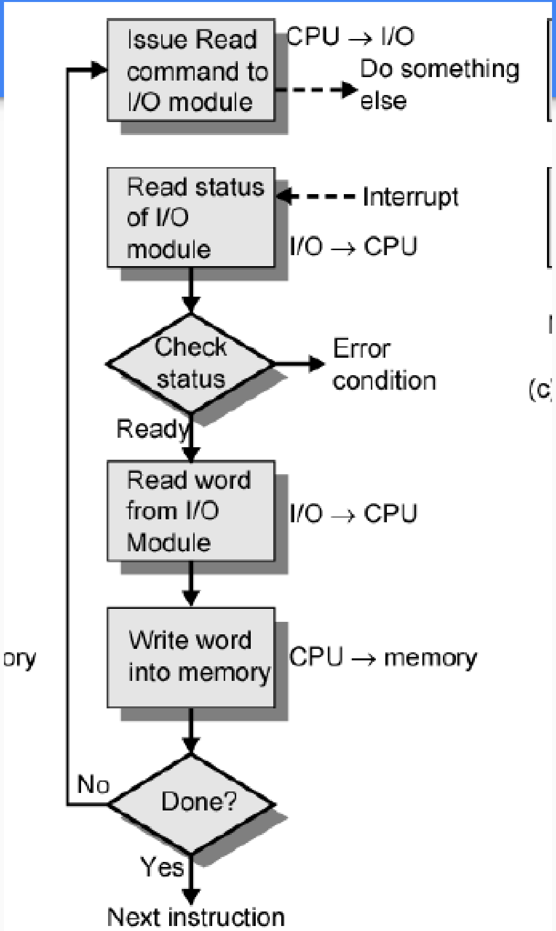
Interrupt Driven I/O

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



(b) Interrupt-driven I/O

Direct Memory Access (DMA)

- 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).

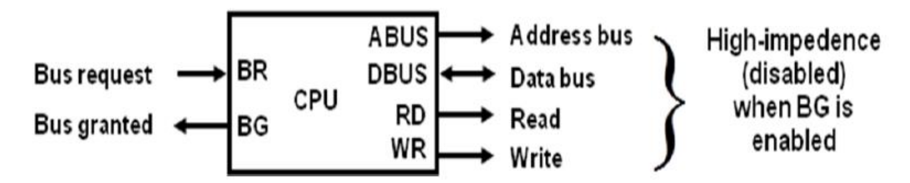


Fig: CPU bus signal for DMA transfer

- 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.

Direct Memory Access (DMA)

- 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.

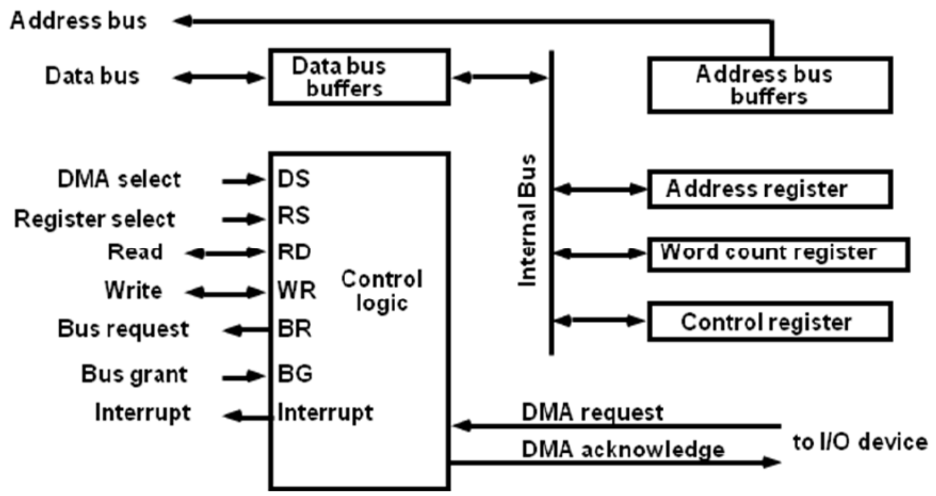


Fig: Block diagram of DMA controller

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.

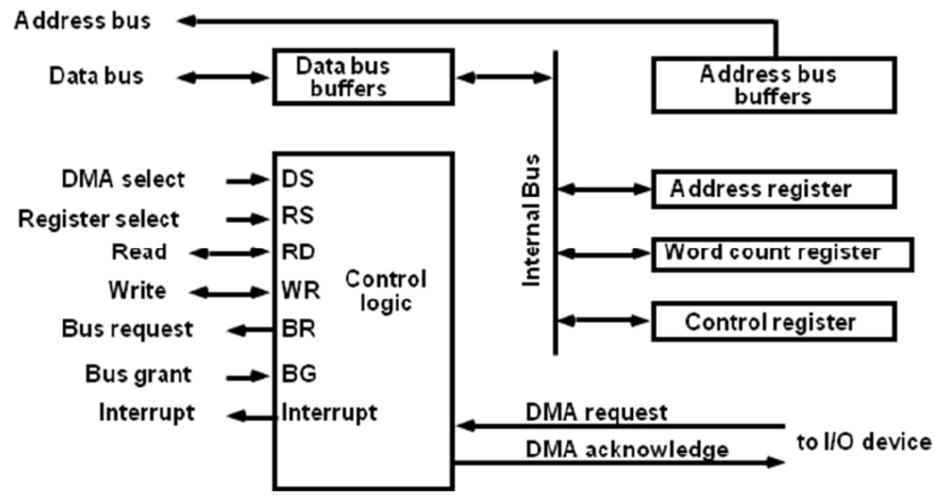
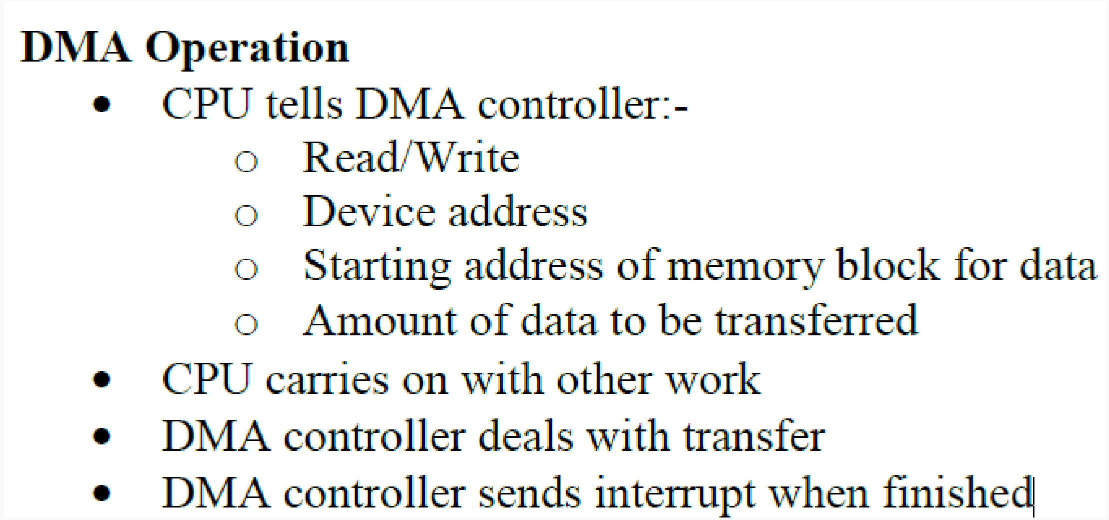


Fig: Block diagram of DMA controller

DMA Controller



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 Channels and Processors

- I/O channels are the module that behaves like a processor with specialized instruction set.
- CPU directs the processor to execute a I/O program in memory.
- I/O processor fetches an executes instruction without any involvement of CPU. CPU is interrupted only when I/O activities are completed.
- For I/O processors, module has its own local memory. With this large set of I/O devices can be controlled with minimal involvement of CPU. It controls the communication with interactive terminals.

I/O Channels and Processors

- I/O channel executes I/O instructions that completely control I/O operation. CPU does not executes I/O instructions instead all instructions are stored in main memory that can be executed by I/O processor. CPU only initiates I/O channels to perform transfer. The instruction specify device or devices, area or areas of memory storage, priority and actions to be taken.
- Two types of I/O channels:

1. **Selector:** A selector controls multiple high-speed devices but one at a time. I/O channel selects one device and data transfer takes place.

2. Multiplexer: A multiplexer channel can handle multiple I/O devices at same time. For high-speed device, a block multiplexer interleaves block of data from several devices.

I/O Channels and Processors: Selector Channel & Multiplexer Architecture

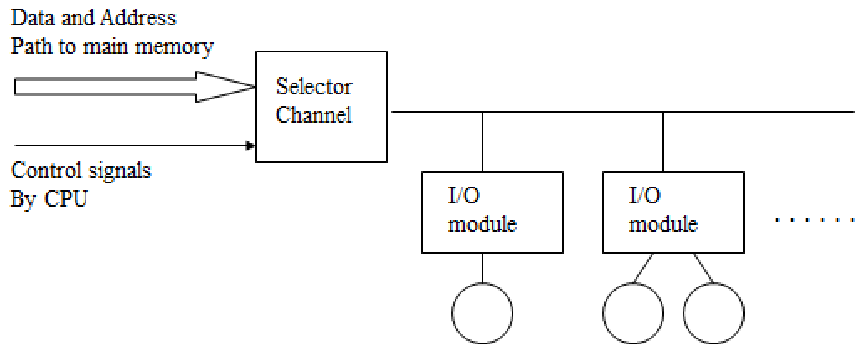


Fig. Selector Channel Architecture

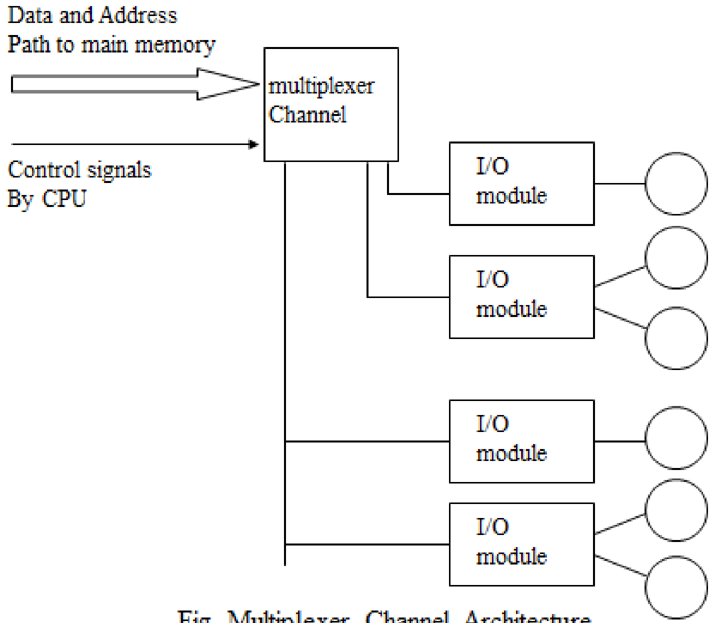


Fig. Multiplexer Channel Architecture

I/O Processor

- ✓ 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.

I/O Processor

- ✓ 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 needs 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.

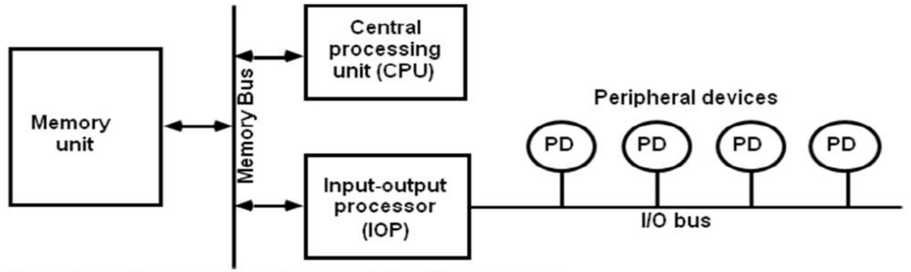
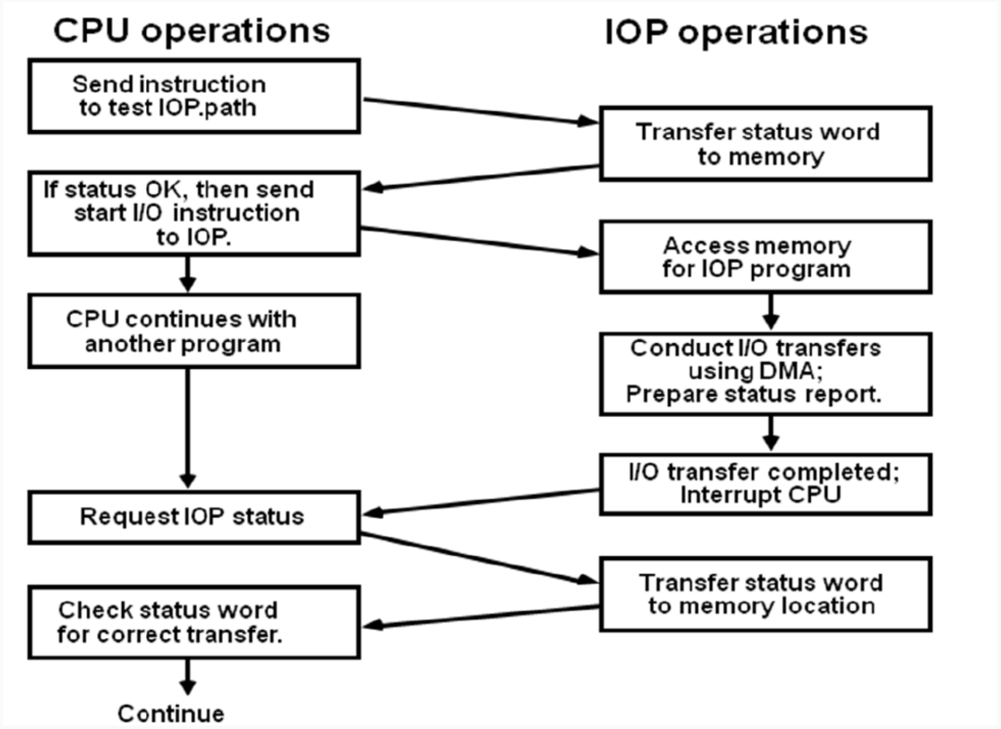


Fig: Block diagram of a computer with I/O Processor

I/O Processor

- ✓ 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.


CPU operations
IOP operations
Send instruction to test IOP.path
Transfer status word to memory
If status OK, then send start I/O instruction to IOP.
Access memory for IOP program
CPU continues with another program
Conduct I/O transfers using DMA; Prepare status report.
I/O transfer completed; Interrupt CPU
Request IOP status
Transfer status word to memory location
Check status word for correct transfer.
Continue

External Interface

- The external bus interface is a computer bus for interfacing small peripheral devices like flash memory with the processor.
- It is used to expand the internal bus of the processor to enable connection with external memories or other peripherals.
- External Bus Interface can be used to share I/O pins controlling memory devices that are connected to two different memory controllers.
- Use of External Bus Interface reduces the total number of system pins required causing the system cost to come down.

External Interface

Two types of I/O module configurations –

1. Point-to-point configuration : provides the dedicated line between the I/O module and the external device. (keyboard)

2. Multipoint configuration: use the external buses (like PCI) and is used to support external mass storage device (disk and tape drives) and multimedia devices (CD ROMs, audio, video etc.).

External Interface

Two types of external buses:

1. Fire Wire Serial Bus
2. InfiniBand

Fire Wire Serial Bus:

- Fire wire interface has advantages over older I/O interfaces. It is very high speed, low cost and easy to implement. Fire wire supports other electronic devices also.
- Fire wire uses a serial transmission rather than parallel.
- Fire wire provides a single I/O interface with a simple connector that can handle numerous devices through a single port so that mouse, printer, external disk drives, sound and LAN hookups can be replaced with single connector. Parallel interface requires shielding, compatibility, large set of wires, large area and expensive.

External Interface: Fire Wire Serial Bus

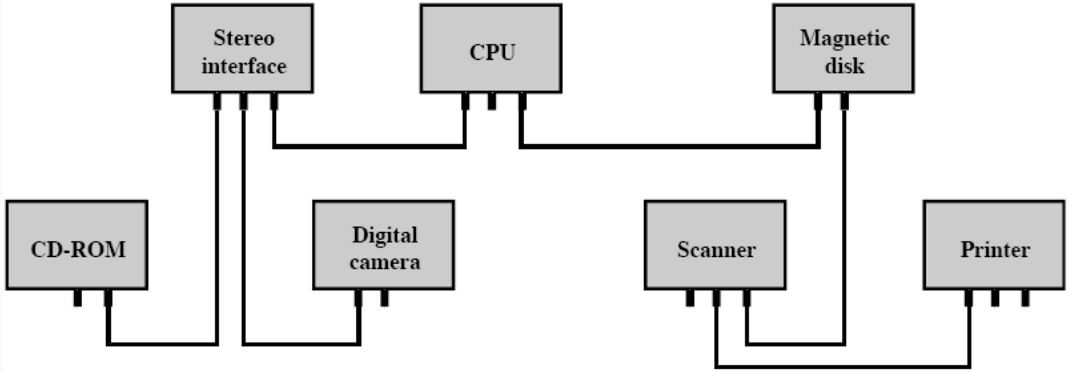


Figure 7.17 Simple FireWire Configuration

External Interface: Fire Wire Serial Bus

- It uses a Daisy-Chain configuration with up to 63 devices connected off a single port. 1022 fire wire buses can be interconnected using bridges to support many peripherals.
- Fire wire provides hot plugging, that is it can connect or disconnect peripherals without having power supply or reconfiguring the system. It also provides automatic configuration i.e. not necessary to set ID's manually. It should not be strictly daisy chained; it also can be tree like structure.

- Fire wire provides a three layer protocol:

✓ Physical: Transmission medium, electrical and signaling characteristics

✓ Link: Transmission of data in packets

✓ Transaction: Request-response protocol

External Interface: Fire Wire Serial Bus

- ✓ **Physical Layer**

Physical layer specifies several transmission media and their connectors with different physical and data transmission properties. Data rates from 25 to 400Mbps. It converts binary data into electric signals. It also provides two forms of arbitration.

- Fair arbitration
- Urgent arbitration

- ✓ **Link Layer**

This layer defines transmission of data in the form of packets. Two transmission types

- Asynchronous: Variable amount of data and several bytes of transaction data are transferred as a packet to an explicit address and then acknowledgement is returned.

- Isochronous: Variable amount of data in sequence of fixed size packets are transmitted at regular intervals with no acknowledgement.

External Interface: InfiniBand

- I/O specification designed for high end servers. Version 1 was released in early 2001.It is the architecture and specification for data flow between processor and intelligent I/O devices.
- Infiniband are intended to replace PCI servers increasing capacity, expandability, and flexibility.
- InfiniBand (IB) is a computer networking communications standard used in high-performance computing that features very high throughput and very low latency.
- It is used for data interconnect both among and within computers.
- InfiniBand is also used as either a direct or switched interconnect between servers and storage systems, as well as an interconnect between storage systems.
- It is designed to be scalable and uses a switched fabric network topology. By 2014, it was the most commonly used interconnect in the TOP500 list of supercomputers, until about 2016.

External Interface: InfiniBand

InfiniBand Architecture

- Infiniband removes the necessity of basic I/O interface hardware. This allows greater server density and allows for a flexible and scalable data center as independent nodes may be added as needed.
- Infiniband channel enables I/O devices to be placed. 17m away from server using copper, 300m using multimode optical fiber and up to 10 km using single mode optical fiber.

The **key element of this architecture are:**

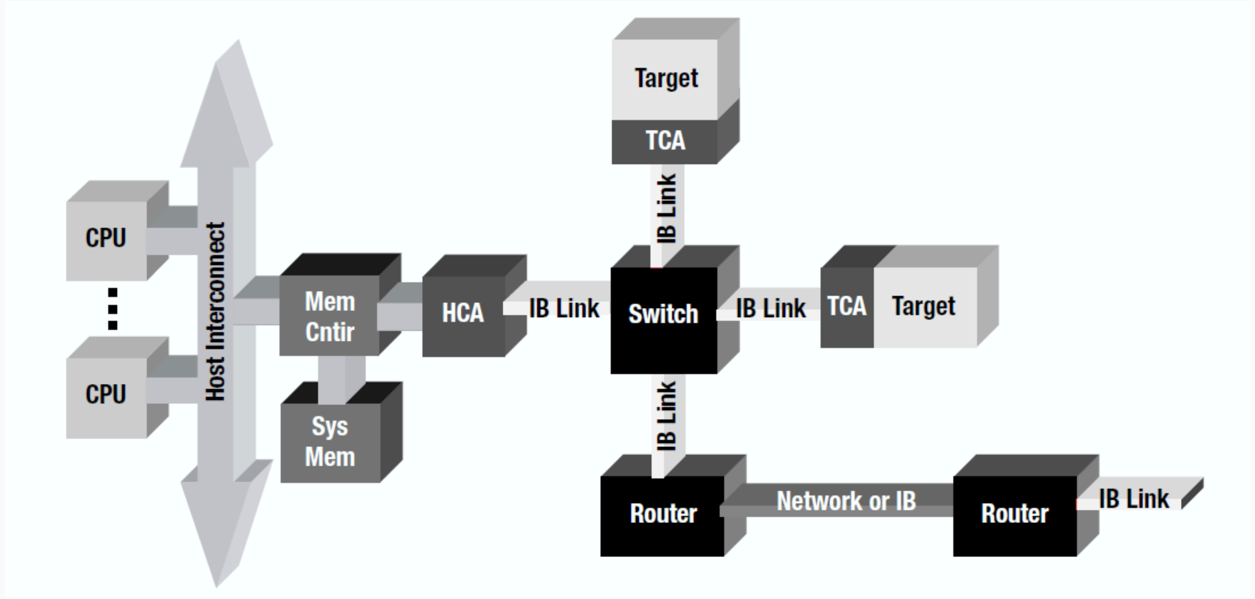
- Host Channel Adapter (HCA)- HCA links the server to an infiniband switch. HCA is attached to memory controller that has access on system bus and controls traffic between processor and memory and between HCA and memory. It uses DMA to read and write memory.

External Interface: InfiniBand

- Target Channel Adapter (TCA)- TCA connect storage systems routers and other peripheral devices to Infiniband.
- Infiniband Switch- A switch provides point to point physical connections to a variety of devices and switches traffic from one link to another. The switch is used for communication between servers and devices through adapters. They manages the linkage without interrupting server’s operation.
- Links- Link between switch and channel adapter or between two switches.
- Subnet- Consists of one or more interconnected switches plus links that connect other devices to switches. Subnet allows administrators to confine broadcast and multicast transmission within the subnet.
- Routers- Connects subnets or switch to a network.

External Interface: Infini Band Diagram

Infiniband



Infiniband architecture

THANK YOU

Any Queries ?

ankitbhattarai@cosmoscollege.edu.np