Computer Organization and Embedded System

Input-Output Organization and Multiprocessor (ACtE0403)

Ishwar Kumar Singh
Computer Engineer
Government of Nepal

4.3 Input-Output Organization and Multiprocessor(ACtE0403):

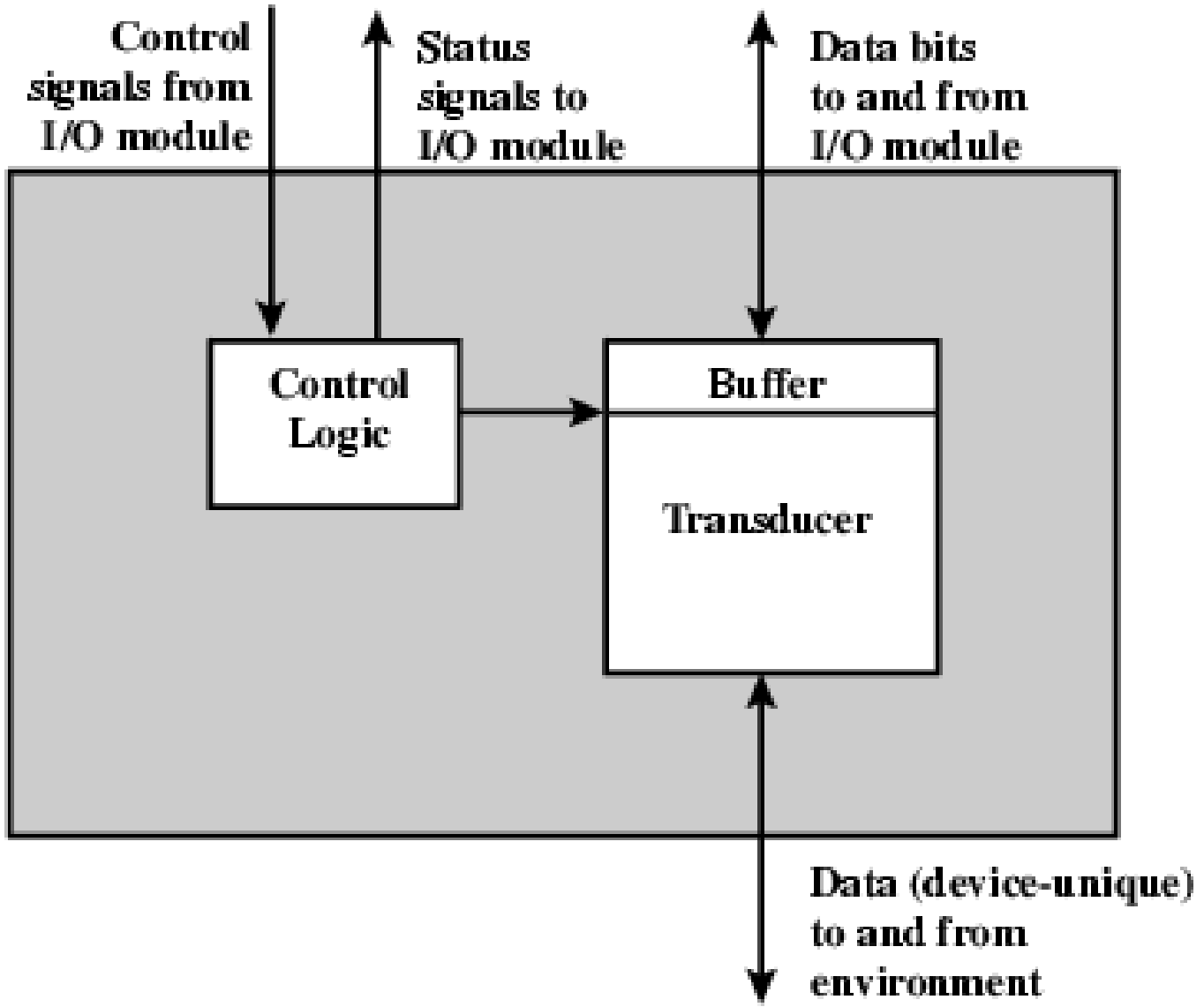
- Peripheral devices,
- I/O modules,
- Input-output interface,
- Modes of transfer
- Direct Memory access,
- Multiprocessors,
- Interconnection Structure,
- Inter-processor Communication
- Inter-process Synchronization

Peripheral Devices

- Input or output devices attached to the computer are called as peripherals: Keyboard, Monitors, etc.
- Peripherals are electromechanical and electromagnetic devices.
- Can be divided 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, etc.
  - **Communication:** Communicating with remote devices, means exchanging data with that, e.g. MODEM, NIC, etc

Peripheral Devices

- 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



Peripheral Devices

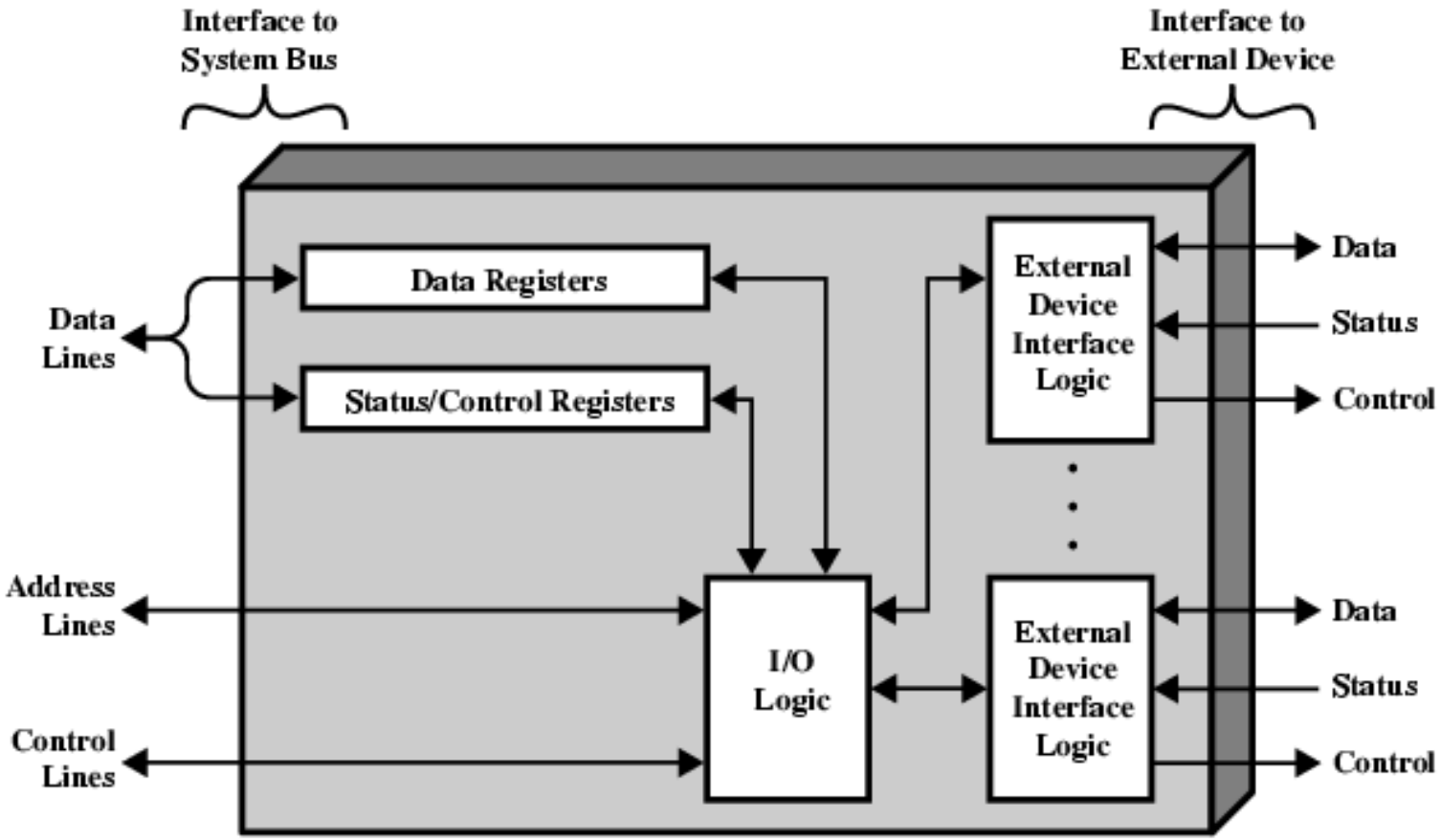
- Control logic associated with the device controls the device's operation in response to direction from the I/O module.
- The transducer converts data from electrical to other forms of energy during output and from other forms to electrical during input.
- Buffer is associated with the transducer to temporarily hold data being transferred between the I/O module and external devices.

I/O Modules

- I/O modules interface to the system bus and controls one or more peripheral devices.
- The reasons due to which peripherals do not directly connected to the system bus are:
  - Wide variety of peripherals with various methods of operation.
  - Slow data transfer rate of peripherals: it is impractical to use high speed system bus to communicate directly with a peripheral and vice versa.
  - Different data format and word length

I/O Modules

- An I/O module 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 Modules Functions

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.

I/O Modules Functions

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.

I/O Modules Functions

- **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.

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

  which resolves the following **differences** between the computer and peripheral devices.
- **Devices and signals**
  - Peripherals - Electromechanical Devices
  - CPU or Memory - Electronic Device

Input Output Interface

- **Data Transfer Rate**
  - Peripherals - Usually slower
  - CPU or Memory - Usually faster than peripherals
- **Unit of Information**
  - Peripherals - Byte
  - CPU or Memory - Word
- **Operating Modes**
  - Peripherals - Autonomous, Asynchronous
  - CPU or Memory – Synchronous

Modes of Transfer

- Binary Information received from an external device is usually stored in memory for latter processing.
- Information transferred from CPU into an external device originates in memory unit.
- CPU merely execute I/O instructions and may accept data temporarily, but the ultimate source or destination is the memory unit.
- There are three possible modes
  - Programmed I/O
  - Interrupt Driven I/O
  - Direct Memory Access

Programmed I/O

- Also known as Polling.
- Data are exchanged between processor and I/O module.
- The processor executes a **programs** that gives it direct control of the I/O operations, including sensing device status, sending a read or write command and transferring the data.
- The processor must **wait** until the I/O operation is completed after issuing a command.
- The I/O module will **perform** the requested action and then **set** the appropriate bits in I/O status register.
- The I/O Module takes no further action to alert the processor i.e. No Interrupt.

Programmed I/O

- The processor bears the responsibility to **check** the states of I/O Module periodically until it finds that the operation is completed.
- Involves a number of I/O commands issued by them.
- I/O instructions are easily mapped into I/O commands i.e. one-to-one Relationship
- I/O devices have unique identifier or address while connected through I/O Modules.
- There are two modes of addressing
  - Memory Mapped I/O
  - Isolated I/O

Memory Mapped I/O

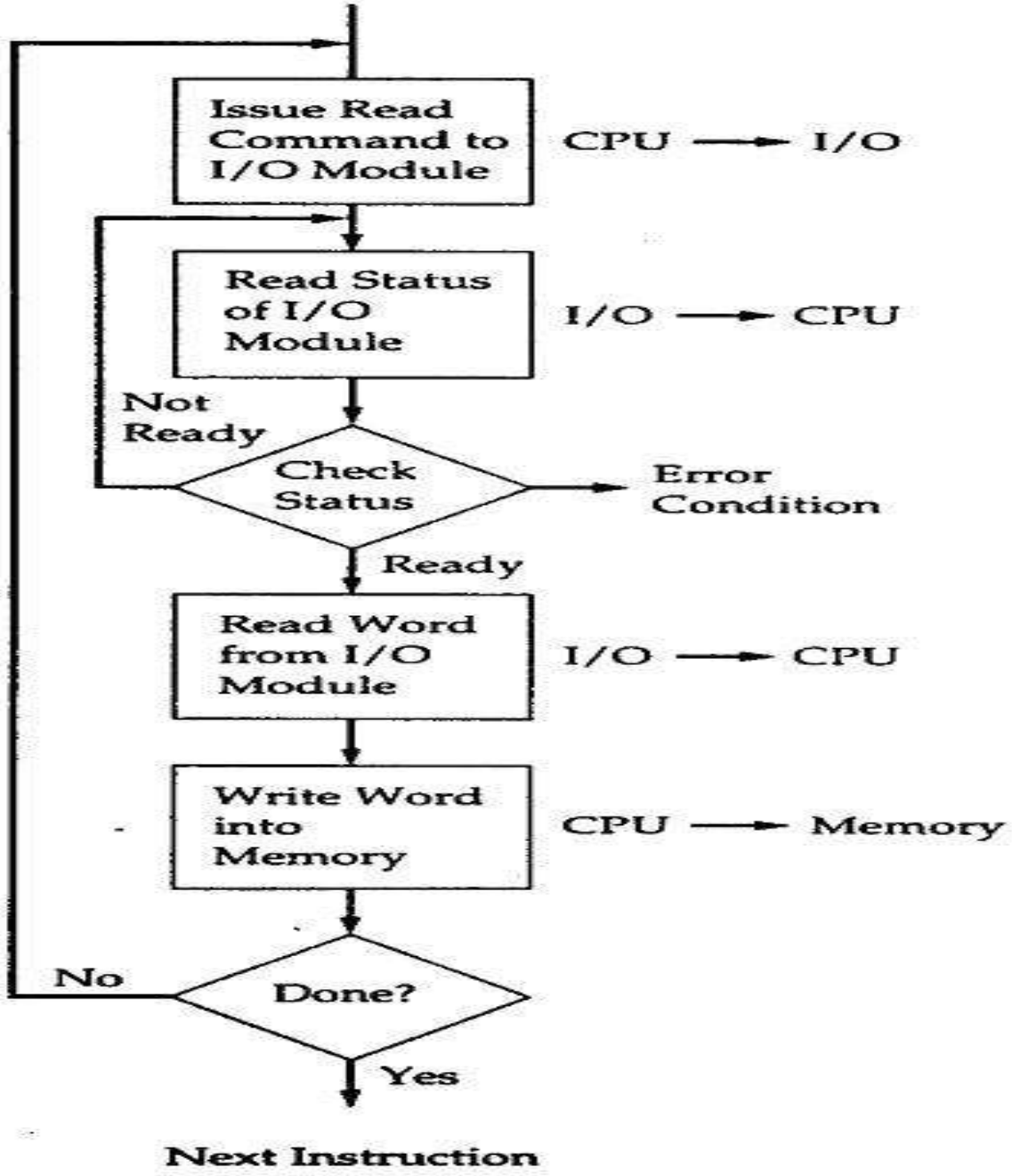
- In this single address space is available for Memory location and I/O devices.
- The processor treats status and data register of I/O Modules as memory location.
- The processor uses the same machine instruction to access both memory and I/O devices.

Isolated I/O

- In this, the bus is equipped with memory read and write plus I/O command lines
- The command lines specifies whether the address refers to a memory location or I/O devices
- Address space for I/O is isolated from that for memory

Programmed I/O

To read a block of data from a peripheral Device ⇒



Problem with Programmed I/O

- 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.

Interrupt Driven I/O

- An alternative to fix the problems encountered with Programmed I/O
- In this, the processor **issues** an I/O command to the I/O 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.

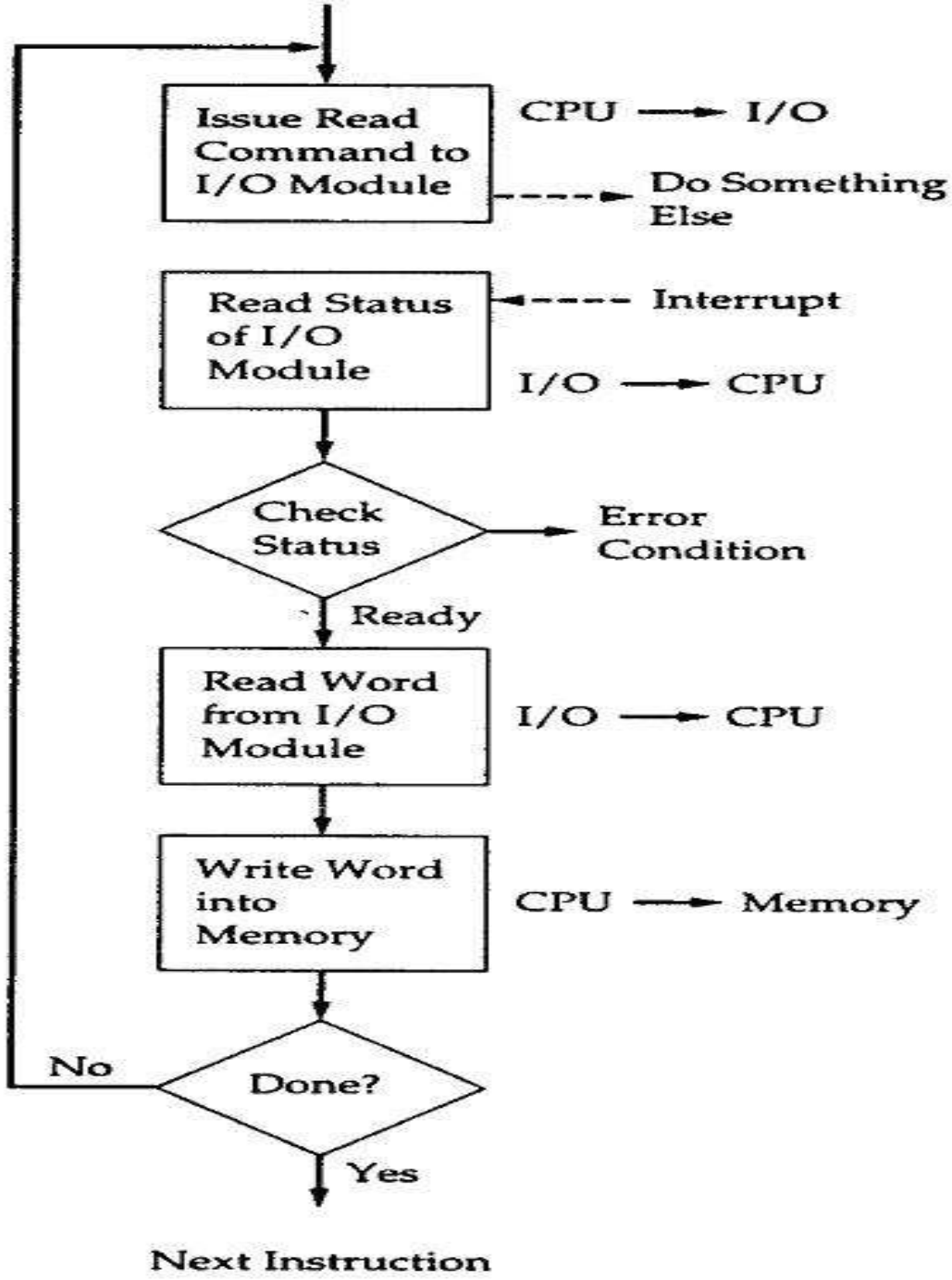
Interrupt Driven I/O

Steps:

- The processer issues a Read command and goes off to do something else.
- At the end of each instruction cycle, the processer checks for interrupts.
- When an Interrupt from I/O module occurs, the processer saves the context (PC, Processor, Register) of current program and processes the interrupt
- In this case, the processor reads the word of date from I/O module and stores it in memory.
- The processor then restores the context of program it was working and resumes the execution

Interrupt Driven I/O

To read a block of data
from a peripheral Device ⇒



Differences

S.N.	Programmed I/O	Interrupt Driven I/O
1	Data transfer is initiated by the means of instructions stored in the computer program.	The I/O transfer is initiated by the interrupt command issued to the CPU.
2	The CPU stays in the loop to know if the device is ready for transfer and has to continuously monitor the peripheral device.	There is no need for the CPU to stay in the loop as the interrupt command interrupts the CPU when the device is ready for data transfer.
3	This leads to the wastage of CPU cycles as CPU remains busy needlessly and thus the efficiency of system gets reduced.	The CPU cycles are not wasted as CPU continues with other work during this time and hence this method is more efficient.

Differences

S.N.	Programmed I/O	Interrupt Driven I/O
4	CPU cannot do any work until the transfer is complete as it has to stay in the loop to continuously monitor the peripheral device.	CPU can do any other work until it is interrupted by the command indicating the readiness of device for data transfer
5	Its module is treated as a slow module.	Its module is faster than programmed I/O module.
6	The performance of the system is severely degraded.	The performance of the system is enhanced to some extent.

Problem with both Programmed I/O and Interrupt Driven I/O

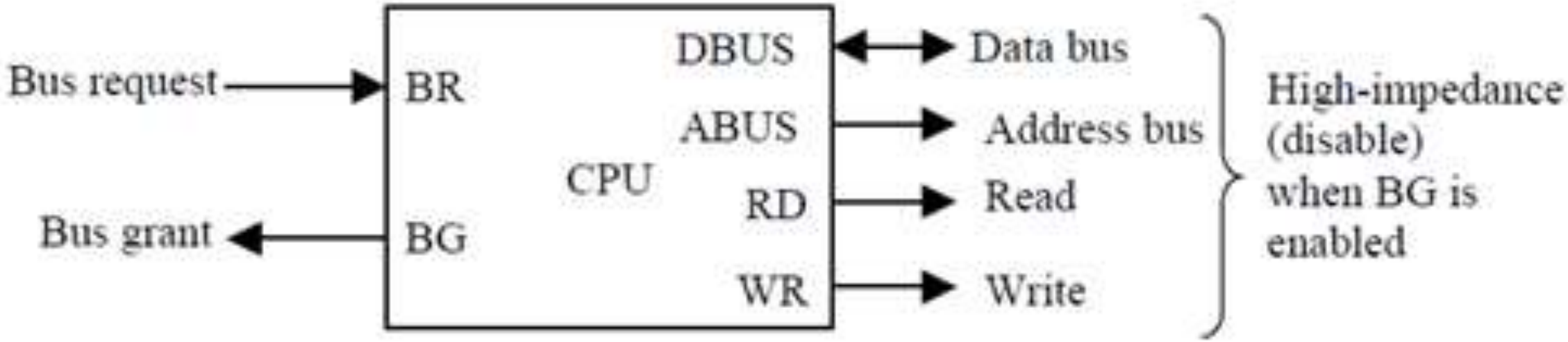
- The I/O transfer rate is limited by the speed with which the processer can test and service a device.
- The processer is tied up in managing an I/O transfer i.e. A number of instructions must be executed for each I/O transfer.

Direct Memory Access

- The transfer of data between the peripheral and memory without the interaction of CPU and letting the peripheral device to manage the memory bus directly is known as Direct Memory Access (DMA)
- During DMA transfer, the CPU remains idle and has no control of the memory buses.
- A DMA controller takes over the control of buses to manage direct transfer between I/O devices and memory.

Direct Memory Access

- The four major control function involved during DMA transfer are:
  - Bus Request
  - Bus Grant
  - Burst Transfer
  - Cycle Stealing



Bus Request

- This input is used by DMA controller to request the CPU for the control of the Buses.
- When it is active, CPU terminates the execution of current instruction and places the data, address, read and write lines to the high impedance state.

Bus Grant

- CPU actives this signal output to inform the external DMA that the buses are in high impedance state
- Then DMA controller takes the control of buses to conduct direct data transfer between I/O devices and Memory without the interaction of CPU
- When DMA terminates the transfer, it disables the Bus Request line and CPU disables Bus Grant and takes control of the buses and return to its normal operation.

Burst Transfer

- When DMA takes control of Buses, it communicates directly with memory, the transfer can be made in several ways.
- One of the way is Burst Transfer in which a block of sequence consisting of a number of memory word, is transferred in a continuous burst while the DMA controller is the master of memory buses.

Cycle Stealing

- Another approach of transfer is Cycle Stealing which allows the DMA controller to transfer one data word at a time, after which it must return the control of buses to the CPU.
- CPU is usually much faster than DMA, thus CPU uses the most of the memory cycles and allows DMA Controller to steals some of the memory cycles from CPU.
- For those stolen cycles, CPU remains idle and DMA controller transfer the data word.

Thank You.