काशी हिन्दू विश्वविद्यालय

BANARAS HINDU UNIVERSITY

Guided by Dr. Sarvesh Pandey Sir

Presented by :

Anchal Jaiswal (20229phy001)
Anusha Mishra (20229MAT008)
Arti Chaudhary (20229MAT001)
Komal Sharma (20229CMP009)
Shweta Singh (20229MAT013)
Shruti Chaurasia (20229MAT012)

Overview

- **Peripheral Devices**
- **Input-Output Interface**
- **Asynchronous Data Transfer**
- **Modes of Transfer**
- **Priority Interrupt**
- **Direct Memory Access**
- **Input-Output Processor**
- **Serial Communication**

Input Output Organization

– **I/O Subsystem**

- 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 computer must be recorded and displayed to user.

Peripheral Devices

- **Devices that are under direct control of computer are said to be connected on-line.**
- **Input or output devices attached to the computer are also called peripherals.**
- **There are three types of peripherals :**
  - **Input peripherals**
  - **Output peripherals**
  - **Input-output peripherals**

Peripheral (or I/O Device)
Monitor (*Visual Output Device*) : CRT, LCD
KeyBoard (*Input Device*) : light pen, mouse, touch screen, joy stick
Printer (Hard Copy Device) : **Daisy wheel**, **dot matrix** and **laser printer**
Storage Device : Magnetic tape, magnetic disk

Peripheral Devices

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
-

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Daisy Wheel, Dot Matrix, Laser)
- Plotter

I/O Interface

- **Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices.**
- **They are special hardware components between CPU and peripheral to supervise and synchronize all input and output transfer.**
- **They are called interface units because they interface between the processor bus processor bus and the peripheral device.**
- **Resolves the *differences* between the computer and peripheral devices**

(1) Peripherals – Electromechanical or Electromagnetic Devices

CPU or Memory - Electronic Device

- **Conversion of signal values required**

(2). Data Transfer Rate

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

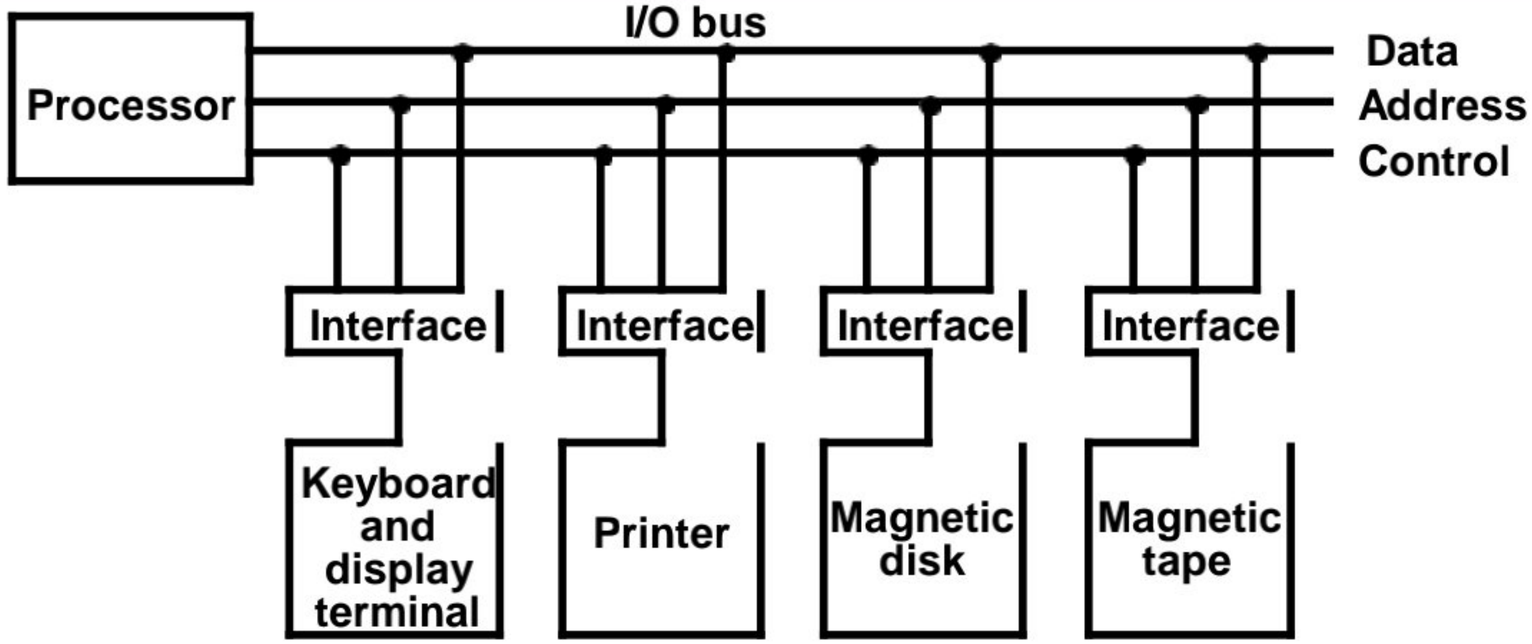
(3) Data formats or Unit of Information

- **Peripherals – Byte, Block, …**
- **CPU or Memory – Word**

(4) Operating modes of peripherals may differ

- **must be controlled so that not to disturbed other peripherals connected to CPU**

I/O Bus and Interface



Interface :

- **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**

4 types of command interface can receive : control, status, data o/p and data i/p

I/O Commands

I/O Command is an instruction that is executed in the interface and its attached peripheral units.

- **Control command :** is issued to activate peripheral and to inform what to do
- **Status command :** used to test various status condition in the interface and the peripherals
- **Data o/p command :** causes the interface to respond by transferring data from the bus into one of its registers
- **Data i/p command :** interface receives an item of data from the peripheral and places it in its buffer register.

I/O Bus and Memory Bus

Functions of Buses

- ***MEMORY BUS*** **is for information transfers between CPU and the MM**
- ***I/O BUS*** **is for information transfers between CPU and I/O devices through their I/O interface**

- **Three ways , bus can communicate with memory and I/O :**

(1) **use two separate buses, one to communicate with memory and the other with I/O interfaces**

(2) **Use one common bus for memory and I/O but separate control lines for each**

(3) **Use one common bus for memory and I/O with common control lines for both**

Isolated vs. Memory Mapped I/O

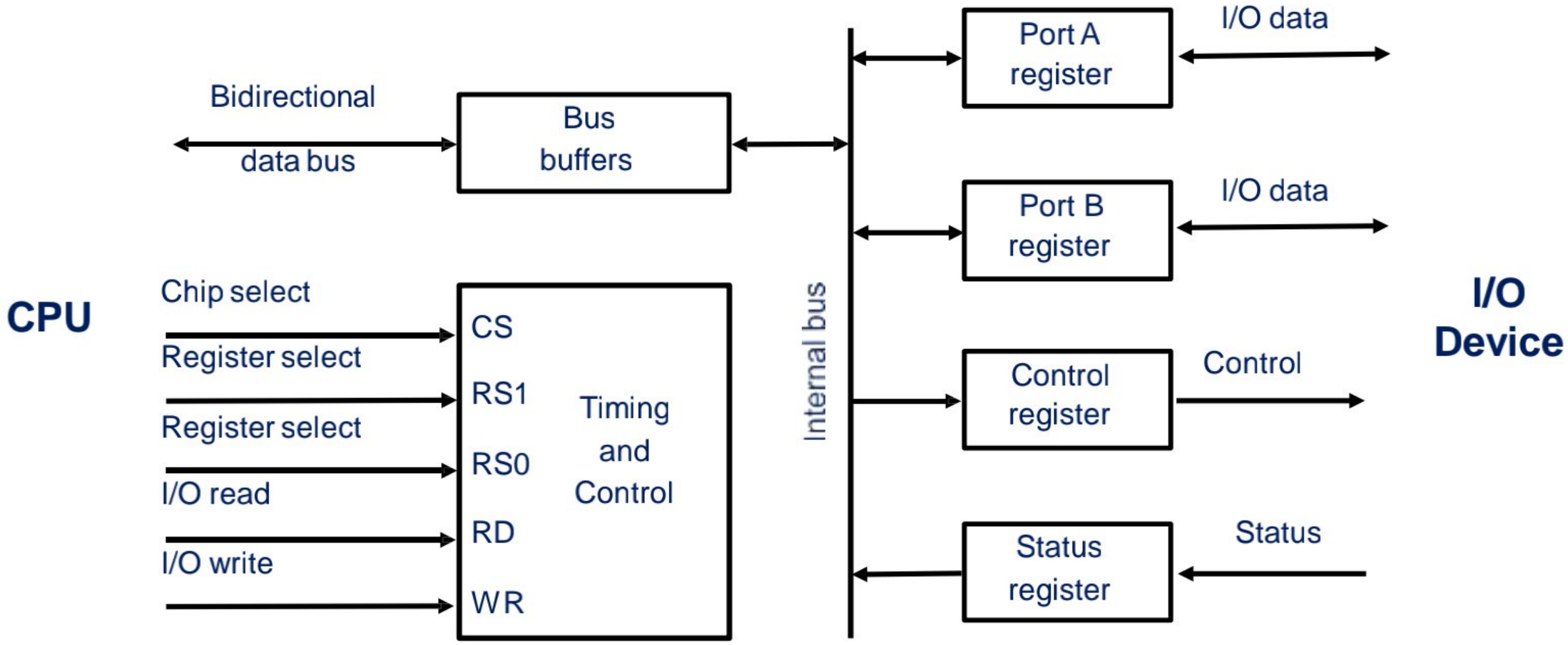
Isolated I/O

- Many computers use common bus to transfer information between memory or 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
  - each associated with address of interface register

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
  -> reduces memory address range available
- No specific input or output instruction
  -> The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

Example of I/O Interface



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

ASYNCHRONOUS DATA TRANSFER

- In a computer system, CPU and an I/O interface are designed independently of each other.

- When internal timing in each unit is independent from the other and when registers in interface and registers of CPU uses its own private clock.

- In that case the two units are said to be asynchronous to each other. CPU and I/O device must coordinate for data transfers.

METHODS USED IN ASYNCHRONOUS DATA TRANSFER

- Strobe Control: This is one way of transfer i.e. by means of strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

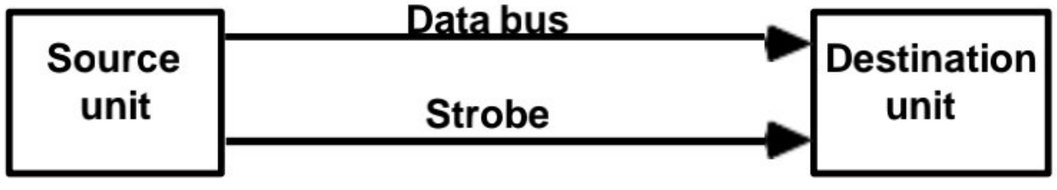
- Handshaking: This method is used to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data.

STROBE CONTROL

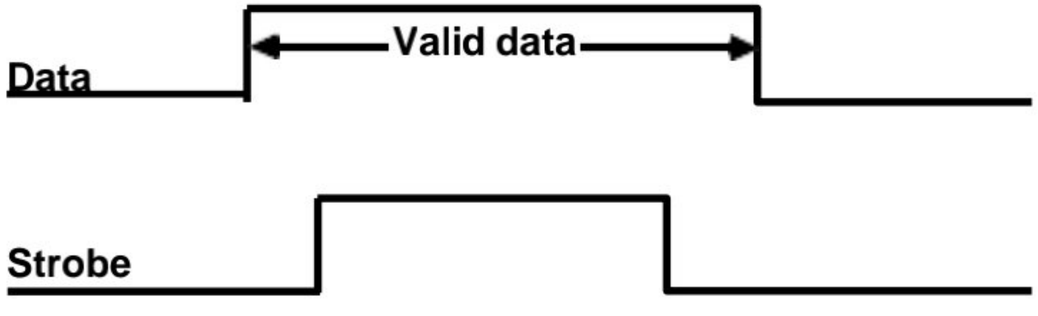
* Employs a single control line to time each transfer
* The strobe may be activated by either the source or the destination unit

Source-Initiated Strobe for Data Transfer

Block Diagram

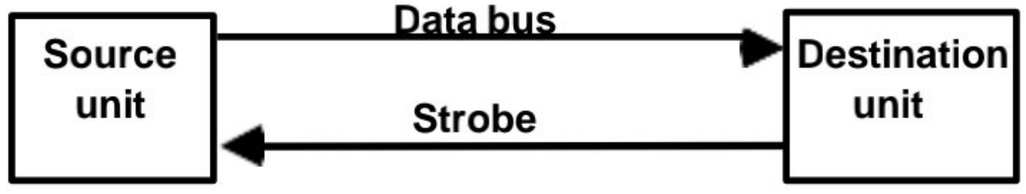


Timing Diagram

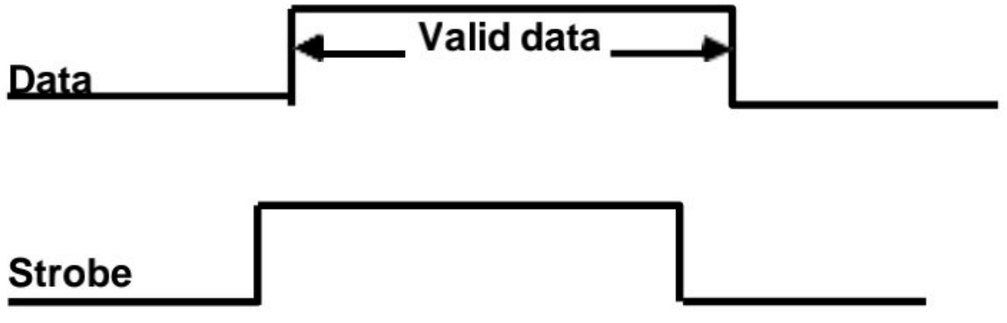


Destination-Initiated Strobe for Data Transfer

Block Diagram



Timing Diagram



HANDSHAKING

Problems in Strobe Methods

Source-Initiated

The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data

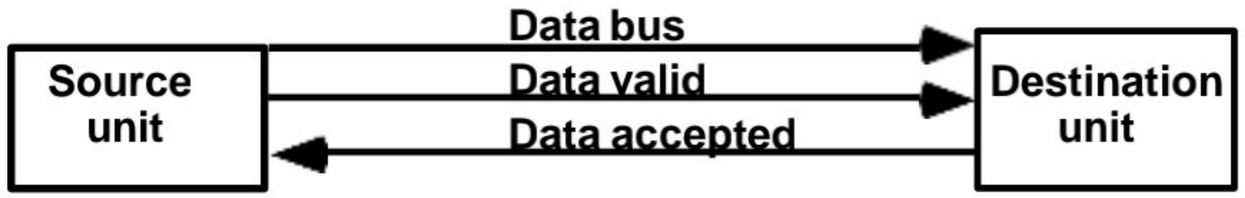
Destination-Initiated

The destination unit that initiates the transfer no way of knowing whether the source has actually placed the data on the bus

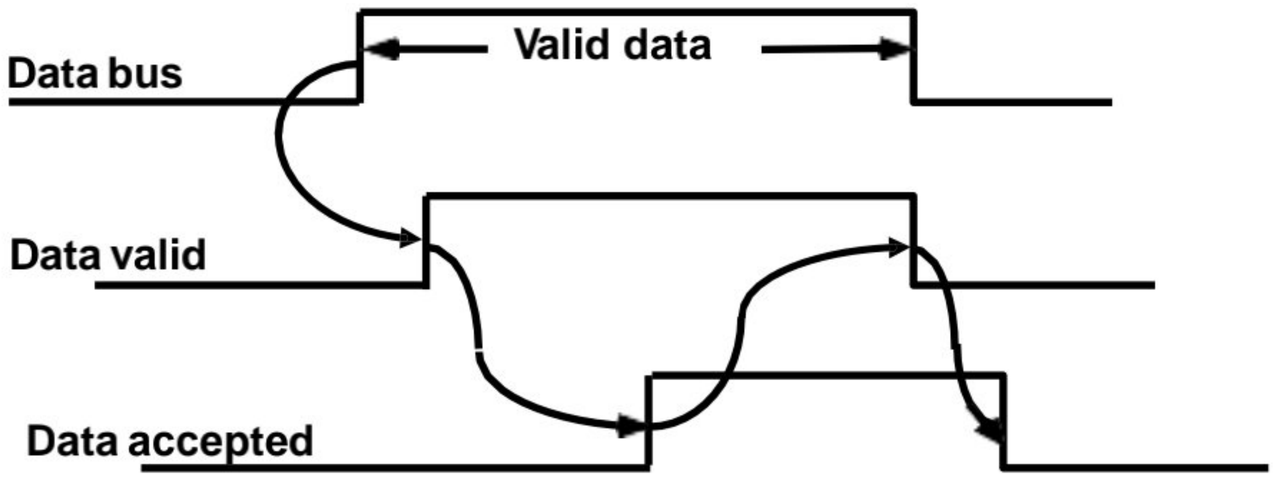
To solve this problem, the ***HANDSHAKE*** **method** introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

SOURCE-INITIATED TRANSFER USING HANDSHAKE

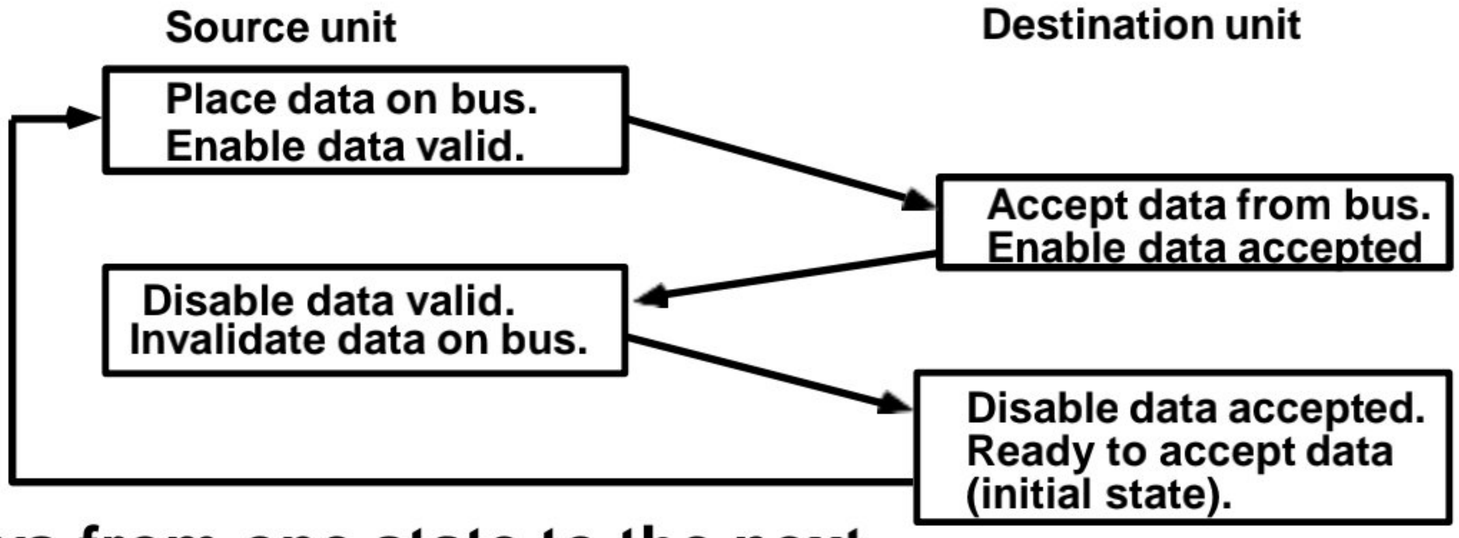
Block Diagram



Timing Diagram



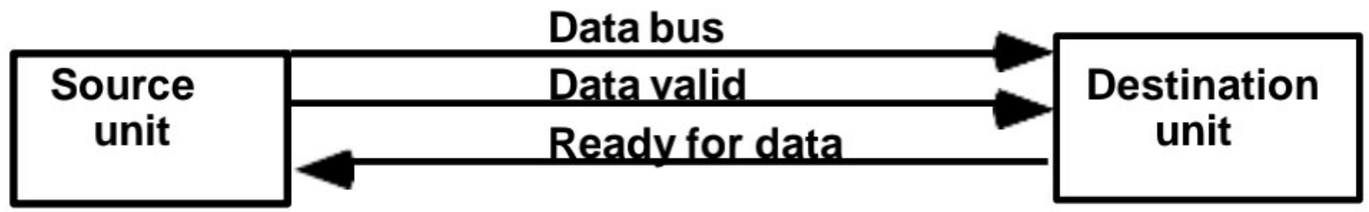
Sequence of Events



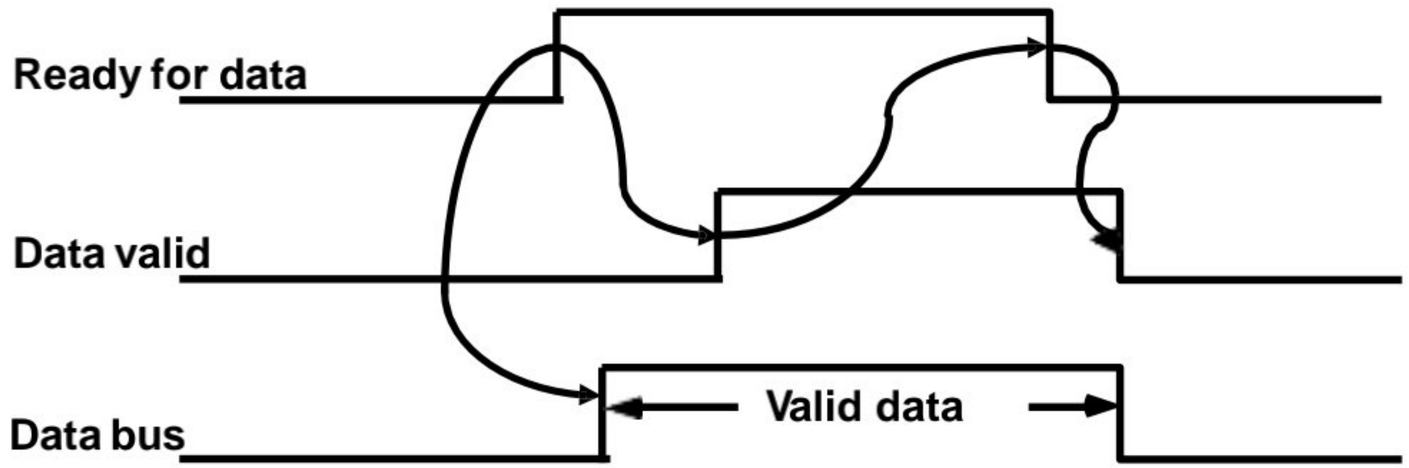
* Allows arbitrary delays from one state to the next
* Permits each unit to respond at its own data transfer rate
* The rate of transfer is determined by the slower unit

DESTINATION-INITIATED TRANSFER USING HANDSHAKE

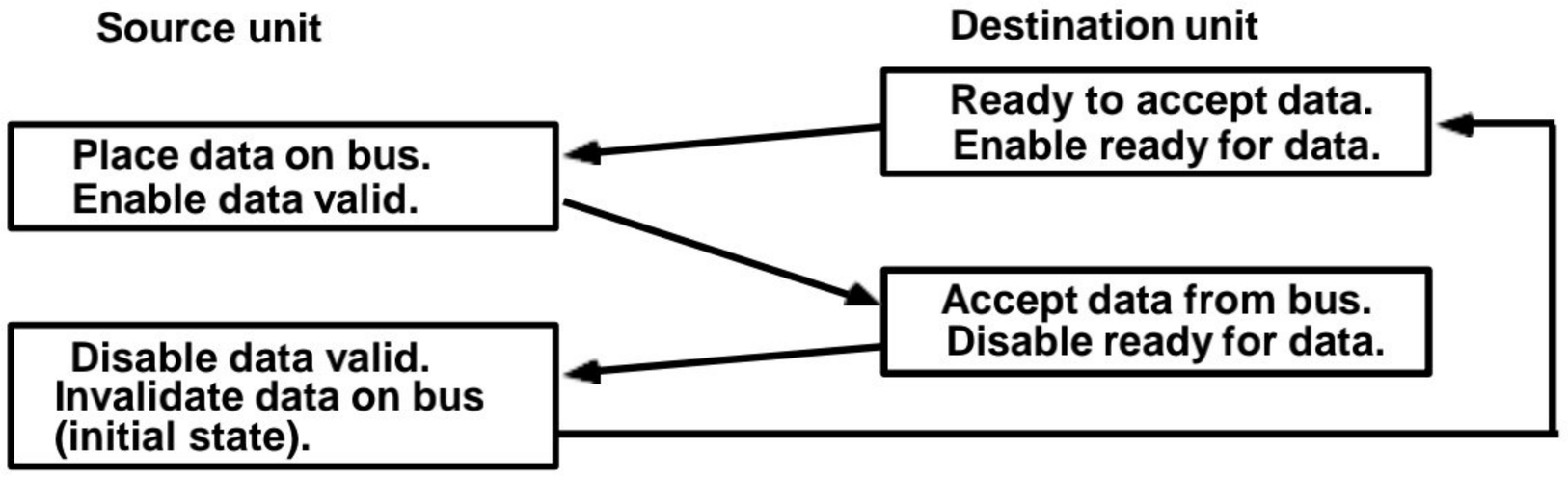
Block Diagram



Timing Diagram



Sequence of Events



* Handshaking provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units
* If one unit is faulty, data transfer will not be completed
  -> Can be detected by means of a *timeout* mechanism

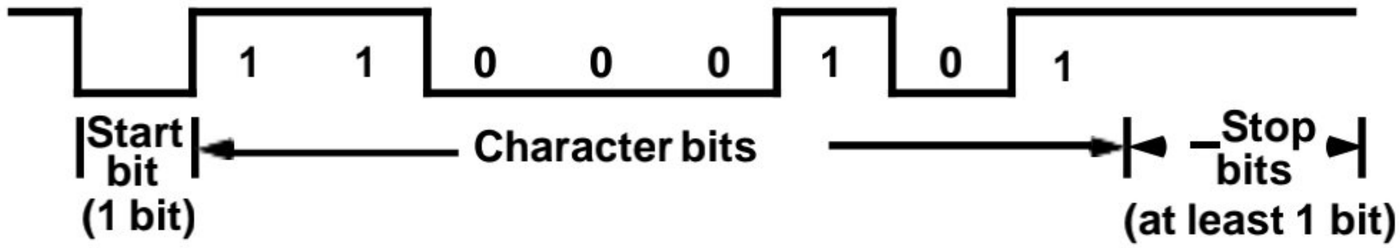
ASYNCHRONOUS SERIAL TRANSFER

Four Different Types of Transfer

Asynchronous serial transfer
Synchronous serial transfer
Asynchronous parallel transfer
Synchronous parallel transfer

Asynchronous Serial Transfer

- **Employs special bits which are inserted at both ends of the character code**
- **Each character consists of three parts; Start bit; Data bits; Stop bits.**



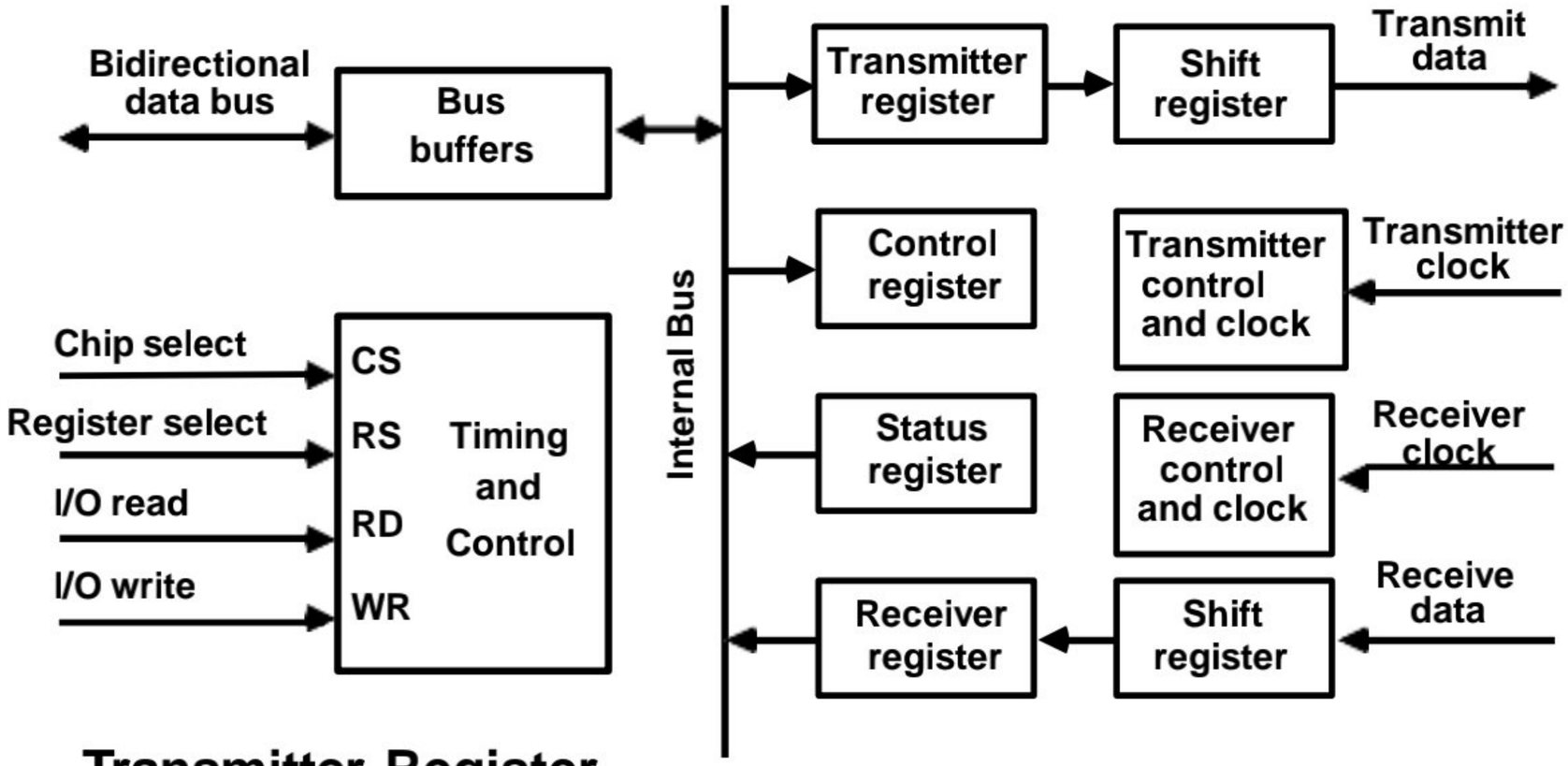
A character can be detected by the receiver from the knowledge of 4 rules;

- **When data are not being sent, the line is kept in the 1-state (idle state)**
- **The initiation of a character transmission is detected by a *Start Bit* , which is always a 0**
- **The character bits always follow the *Start Bit***
- **After the last character , a *Stop Bit* is detected when the line returns to the 1-state for at least 1 bit time**

The receiver knows in advance the transfer rate of the bits and the number of information bits to expect

UNIVERSAL ASYNCHRONOUS RECEIVER-TRANSMITTER
- UART -

A typical asynchronous communication interface available as an IC



CS	RS	Oper.	Register selected
0	x	x	None
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

Transmitter Register
- Accepts a data byte(from CPU) through the data bus
- Transferred to a shift register for serial transmission

Receiver
- Receives serial information into another shift register
- Complete data byte is sent to the receiver register

Status Register Bits
- Used for I/O flags and for recording errors

Control Register Bits
- Define baud rate, no. of bits in each character, whether to generate and check parity, and no. of stop bits

Modes of transfer (11.4)

- Data transfer between the central computer and the I/O devices may be handled in a variety of modes.

- The modes of transfer are:

4. Programmed I/O.
5. Interrupt-initiated I/O
6. Direct memory access (DMA)

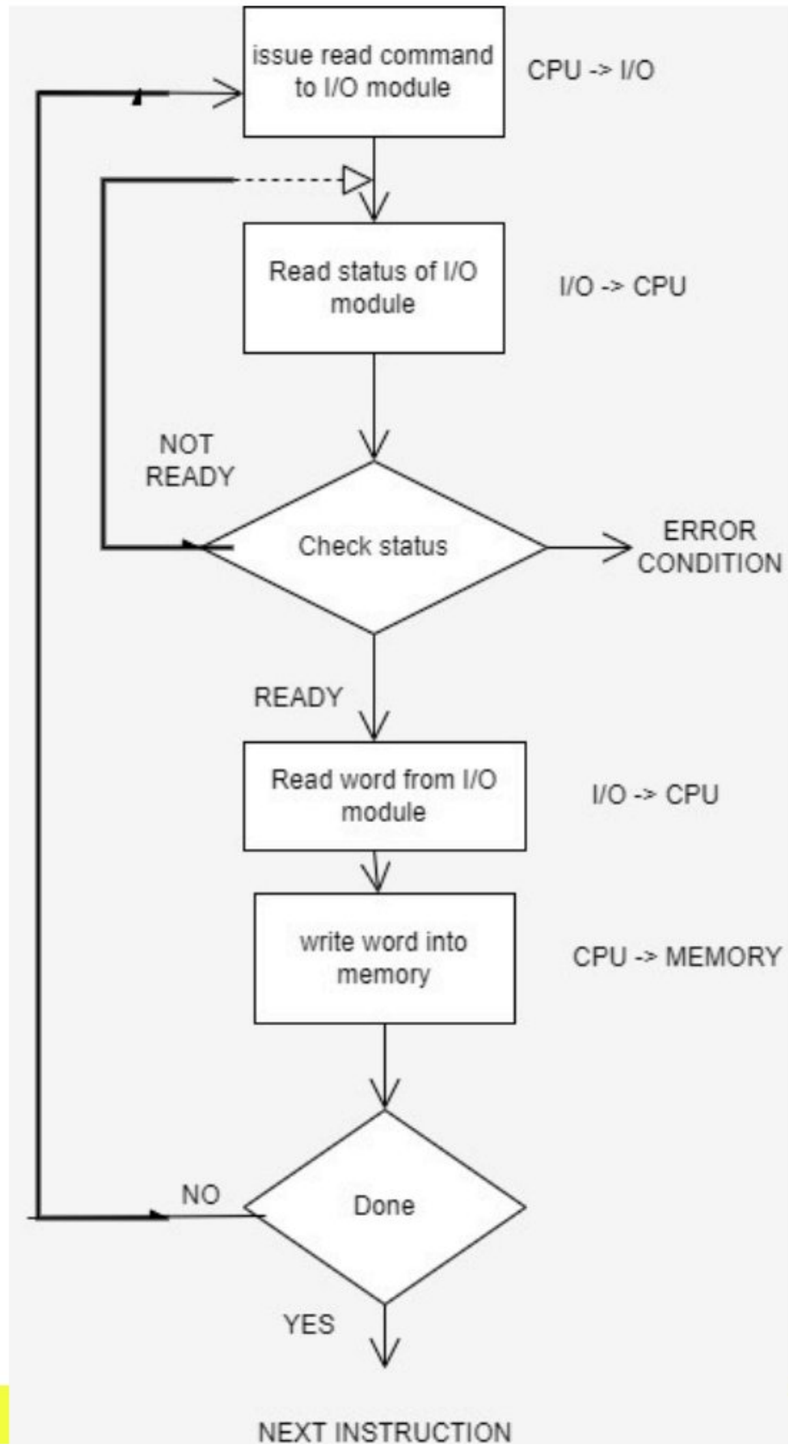
Programmed I/O

- Programmed I/O operations are the result of I/O instructions written in the computer program.
- Each data item transfer is initiated by an instruction in the program.
- Usually, the transfer is to and from CPU register and peripheral.
- Other instructions are needed to transfer the data to and from CPU and memory.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU.
- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.
- It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.
- In this method, the 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.

Programmed I/O

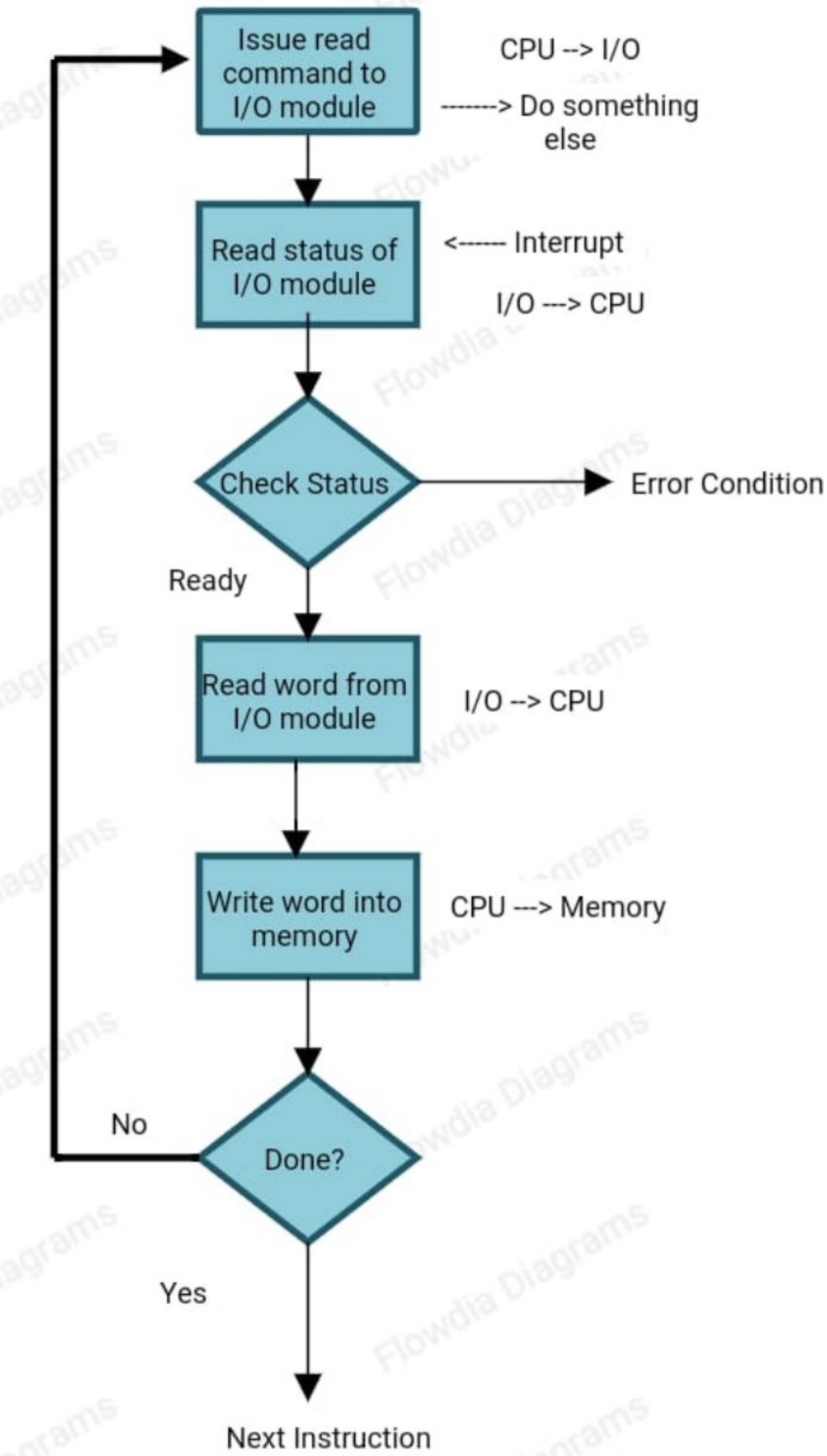
- CPU while executing a program encounters an I/O instruction
- CPU issues I/O command to I/O module
- I/O module performs the requested action & set status registers
- CPU is responsible to check status registers periodically to see if I/O operation is complete. **SO**
- No Interrupt to alert the processor

Interrupts By: Irfanullah Uni



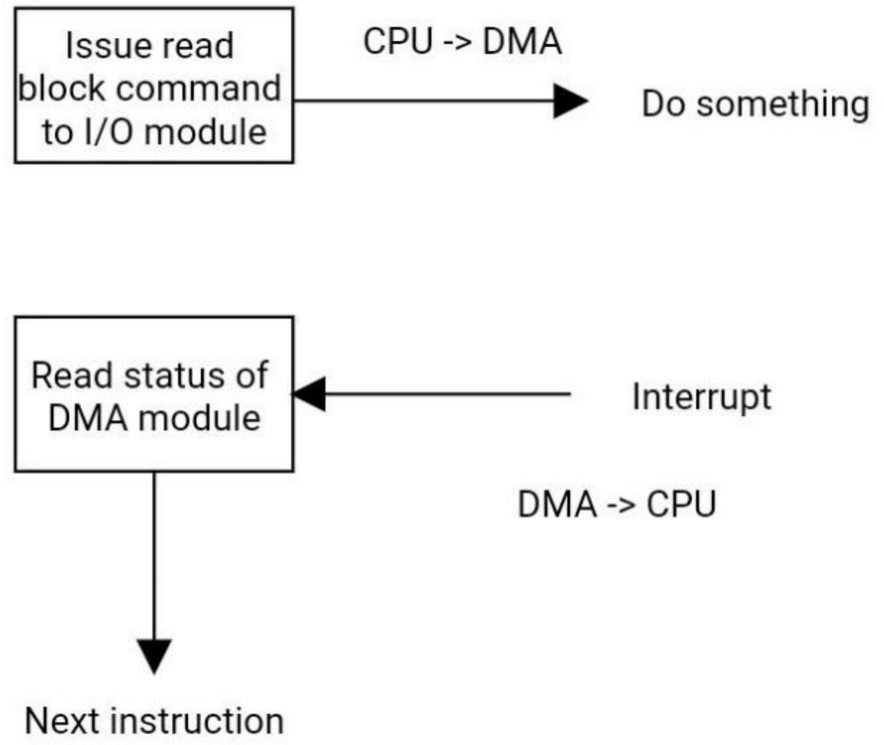
Interrupt Initiated IO

- In the programmed I/O method, the 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 interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.
- In the mean-time the CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device.
- When the interface determines that the device is ready for the data transfer, it generates an interrupt request to the computer.
- Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.


Issue read command to I/O module
CPU --> I/O
-------> Do something else
Read status of I/O module
<------ Interrupt
I/O ---> CPU
Check Status
Error Condition
Ready
Read word from I/O module
I/O --> CPU
Write word into memory
CPU ---> Memory
No
Done?
Yes
Next Instruction

Direct Memory Access (DMA)

- I/O exchanges occur directly with memory
  - Requires DMA module on system bus
  - Capable of mimicking CPU and taking over control of system from CPU
  - DMA will use bus when
    - Processor does not require it OR
    - Must force processor to suspend operation temporarily– called cycle stealing
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer



Interrupts By: Irfanullah University Of Peshawar, F 17

DMA (Direct Memory Access – 11.6)

- Direct memory access is an I/O technique used for high speed data transfer.
- In DMA, the interface transfers data into and out of the memory unit through the memory bus.
- In DMA, the CPU releases the control of the buses to a device called a DMA controller.
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.
- 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, the DMA transfers the data directly into memory.
- The CPU merely delays its memory access operation to allow the direct memory I/O transfer.

Direct Memory Access (DMA)

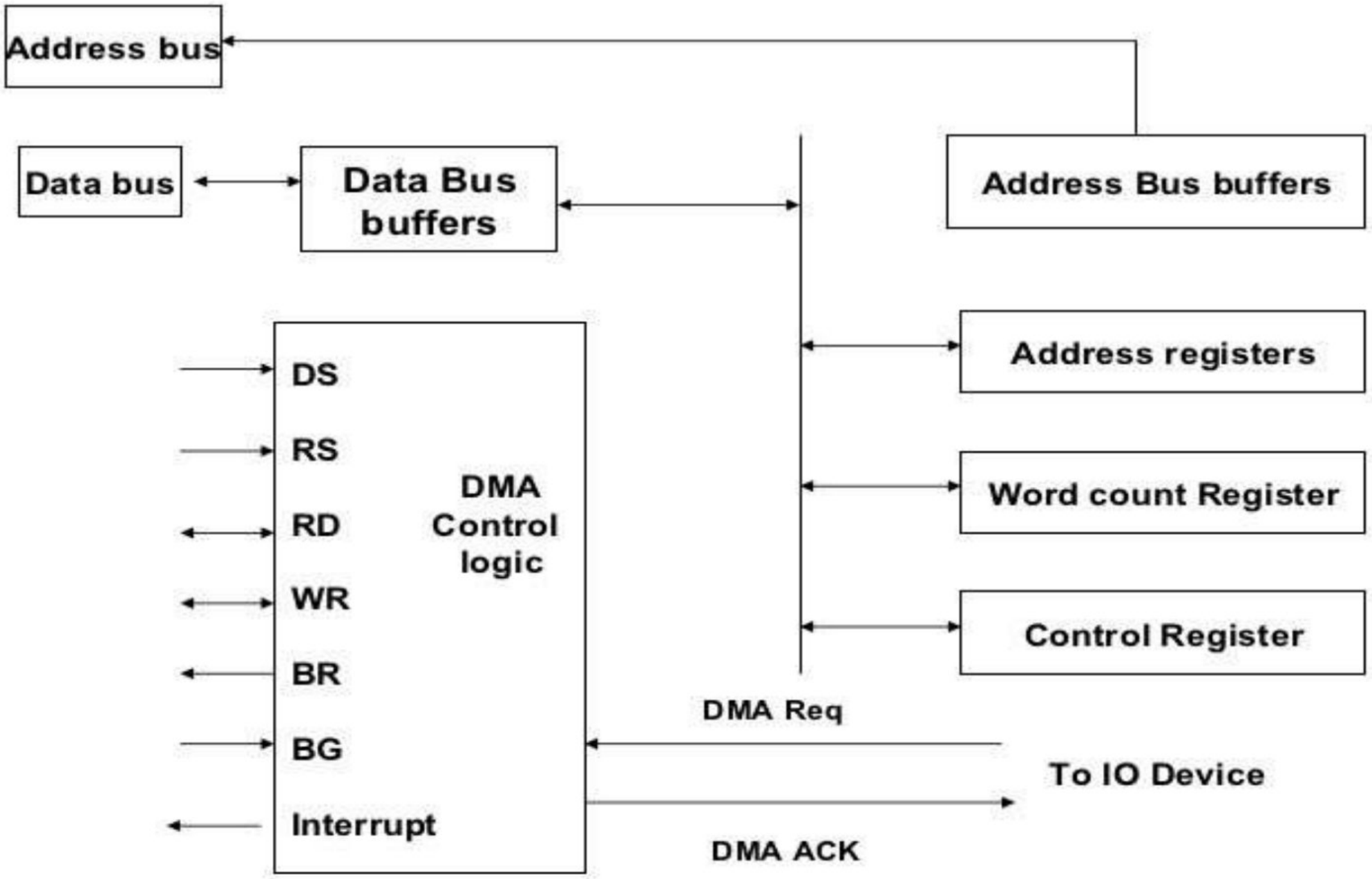
Cycle Stealing

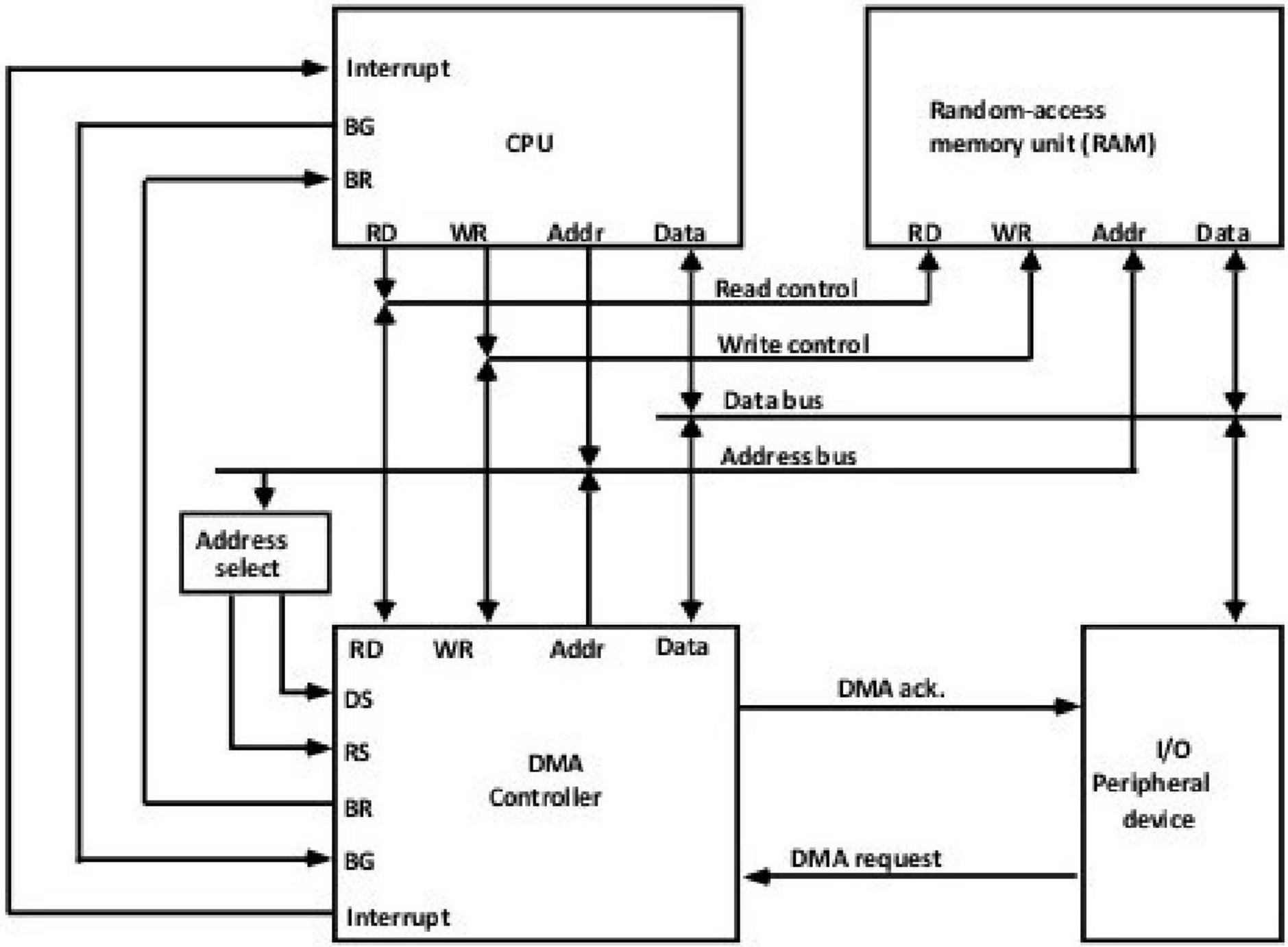
- DMA Controller acquires control of bus
- Transfers a single byte (or word)
- Releases the bus
- The CPU is slowed down due to bus contention

Burst Mode

- DMA Controller acquires control of bus
- Transfers all the data
- Releases the bus
- The CPU operation is temporarily suspended

Block diagram of DMA Controller





Direct Memory Access(DMA)

BURST TRANSFER / CYCLE STEALING

When DMA takes control of the bus system, it communicate directly with The memory. The transfer can be made in several ways.

BURST TRANSFER

In DMA burst transfer, a block sequence consisting of a number of memory word is transferred in a continuous burst. This mode is needed for fast devices.

CYCLE STEALING

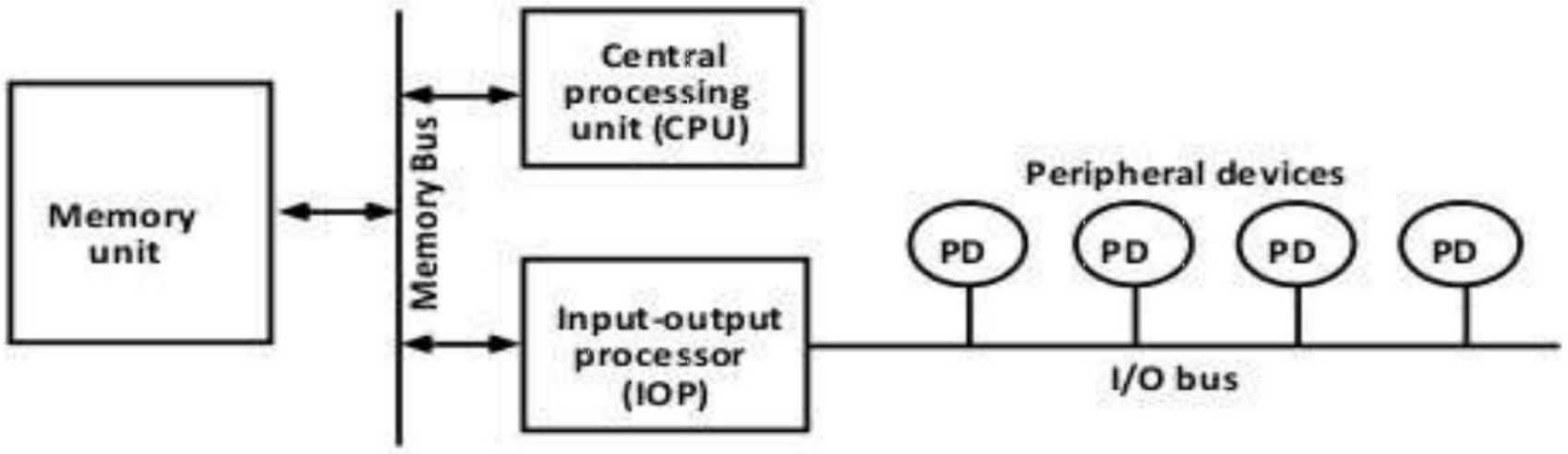
An alternative technique called Cycle Stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU.

- 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
- DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

I/O Processor - Channel

Channel

- **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 an channel I/O class instruction and once initiated, channel operates independently of the CPU**



I/O Processor

- Many computers combines the interface logic with the requirements for direct memory access into one unit and call it an I/O processor. The IOP can handle many peripherals through a DMA and interrupt facility. The computer is divided into three separate modules in such a system.

✓ Memory unit

✓ CPU

✓ IOP

- CPU is the master while the IOP is a slave processor. The CPU performs the tasks of initiating all operations.
- The operations include

✓ Starting an I/O transfer

✓ Testing I/O status conditions needed for making decisions on various I/O activities.