Noida Institute of Engineering and Technology, Greater Noida

Input / Output Unit

Unit: 5

Computer Organization & Architecture

B Tech 3rd Sem

Pradeep Kumar

CSE

Department

Content

Input/output

- Peripheral devices
- I/O interface & I/O ports
- Asynchronous data transfer
- Interrupts: interrupt hardware, types of interrupts and exceptions.
- Modes of Data Transfer:
  - Programmed I/O, interrupt initiated I/O
  - Direct Memory Access.,
- I/O channels and processors.
- Serial Communication: Synchronous & asynchronous communication standard communication interfaces.

.

Course Objective

- Study the different ways of communicating with I/O devices and standard I/O interfaces.
- Study of interrupts and exceptions, DMA, I/O channels and processors,Serial Communication

Course Outcome

Understanding the different ways of communicating with I/O devices and standard I/O interfaces.

NIET Greater Noida
GET FUTURE READY

CO-PO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)												
CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
KCS-302.5	2	2	2	1	2	1	1	1	1	2	2	2

CO- PSO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)				
CO.K	**PSO1**	**PSO2**	**PSO3**	**PSO4**
KCS-302.2	2	3	3	1

Prerequisite and Recap

- Basics of Computer
- Control Unit

Peripheral devices

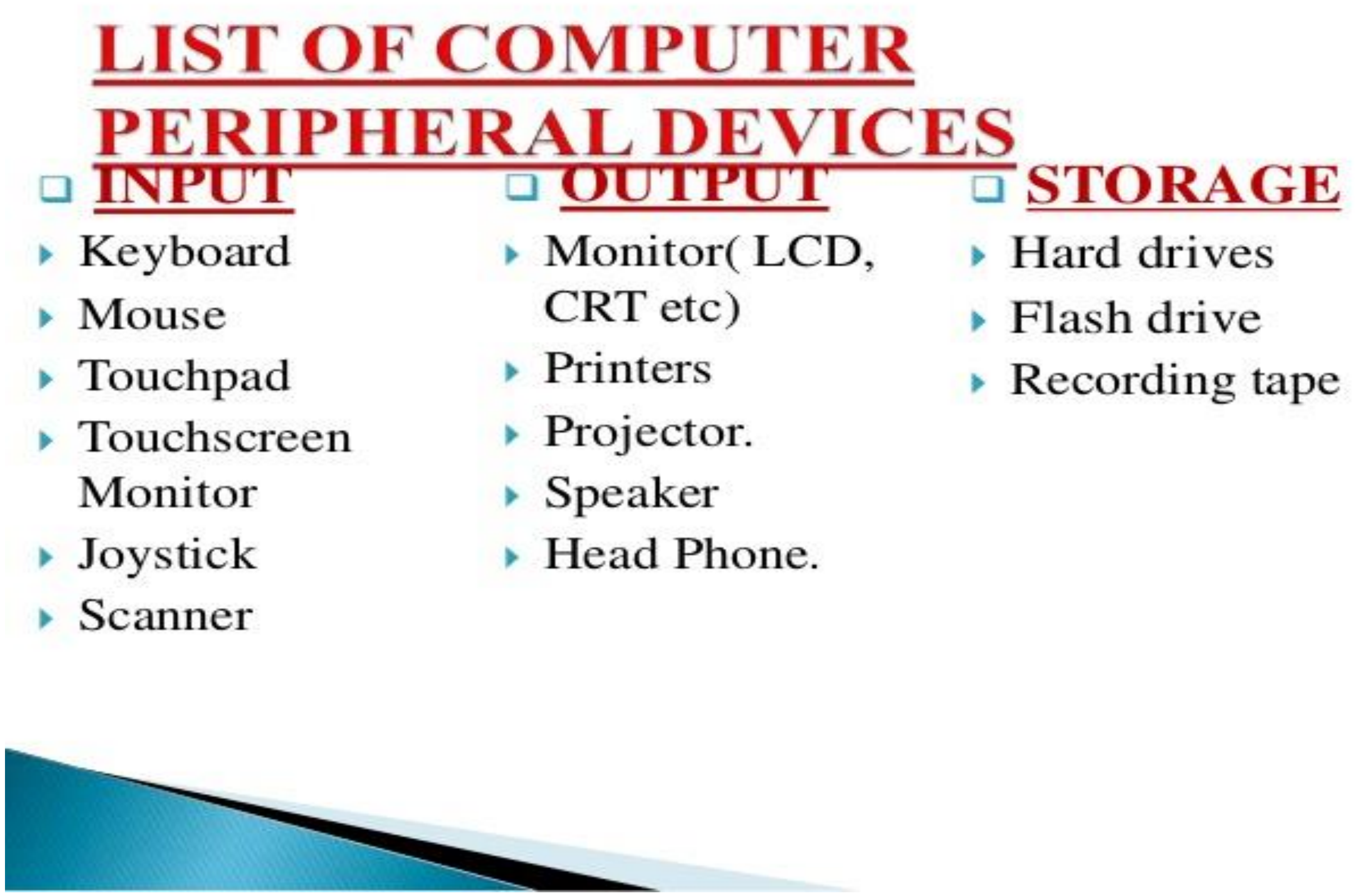
A **peripheral** is a **device** that can be attached to the **computer** processor.

- **Devices** are usually classified as input, output or backing storage **devices**.
  Peripheral devices can be external or internal.

- Examples of external peripherals include mouse, keyboard, printer, monitor, external Zip drive or scanner.

- Examples of internal peripherals include CD-ROM drive, CD-R drive or internal modem.

Peripheral devices

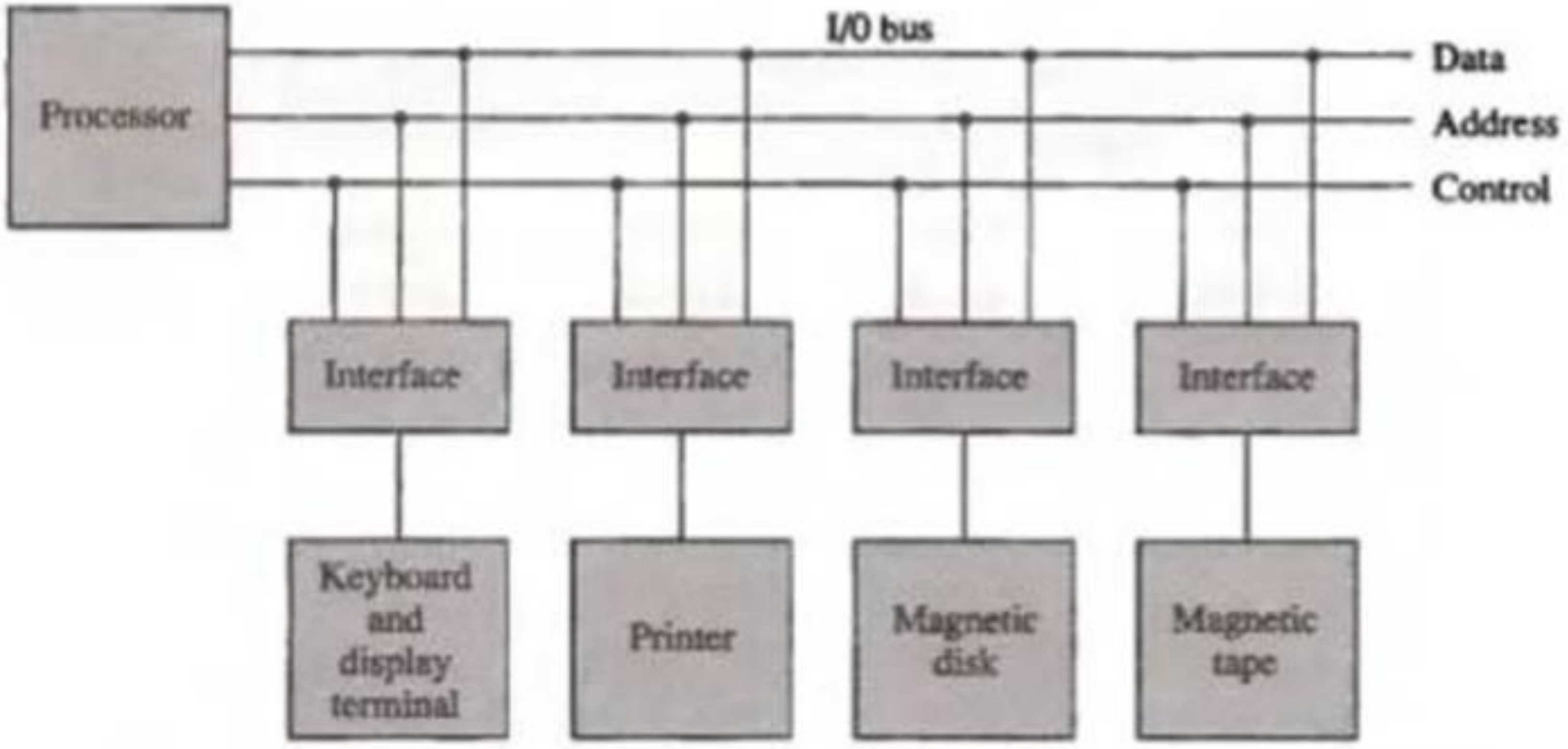


LIST OF COMPUTER PERIPHERAL DEVICES

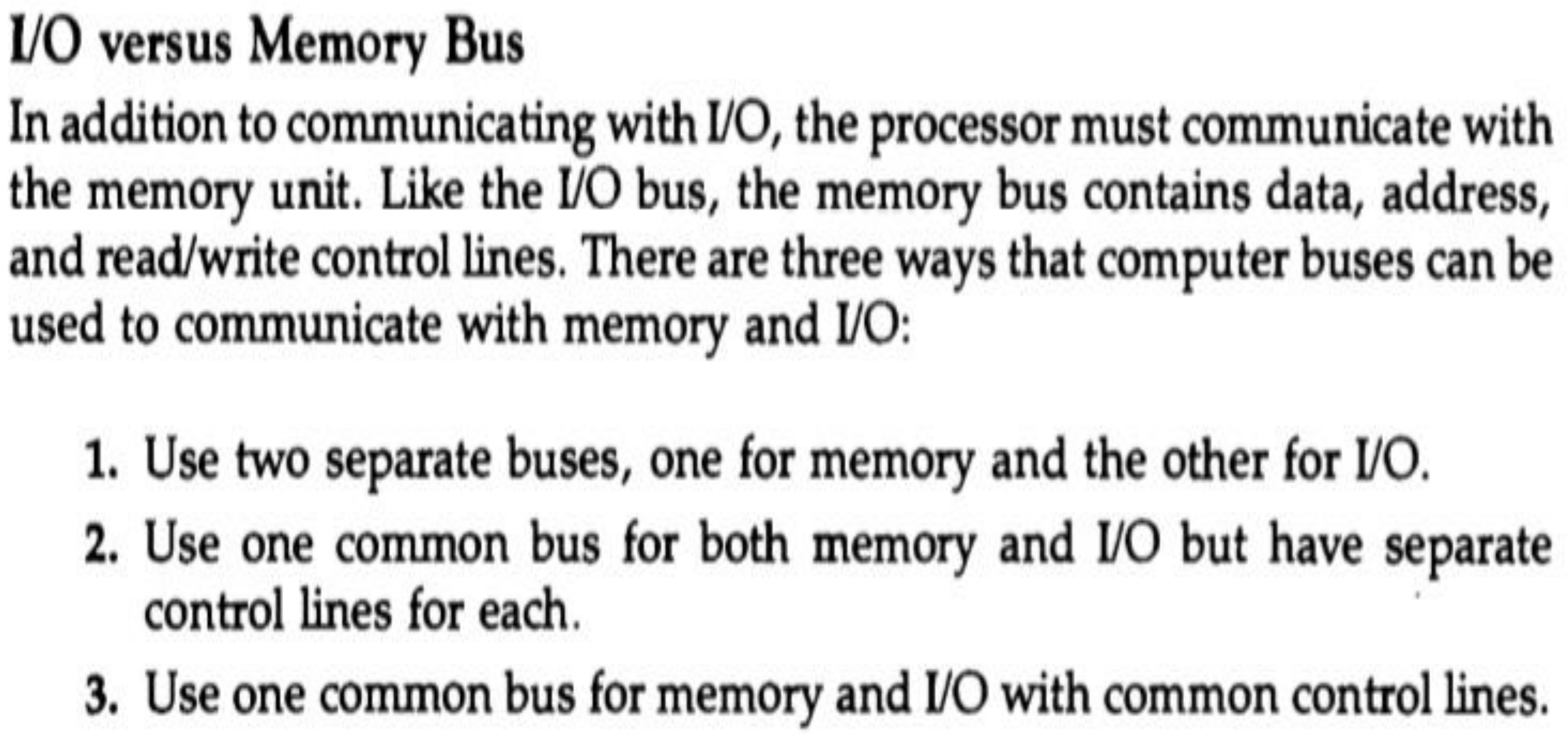
- INPUT
  - Keyboard
  - Mouse
  - Touchpad
  - Touchscreen Monitor
  - Joystick
  - Scanner
- OUTPUT
  - Monitor(LCD, CRT etc)
  - Printers
  - Projector.
  - Speaker
  - Head Phone.
- STORAGE
  - Hard drives
  - Flash drive
  - Recording tape

IO interface

Figure 11-1 Connection of I/O bus to input-output devices.



IO interface

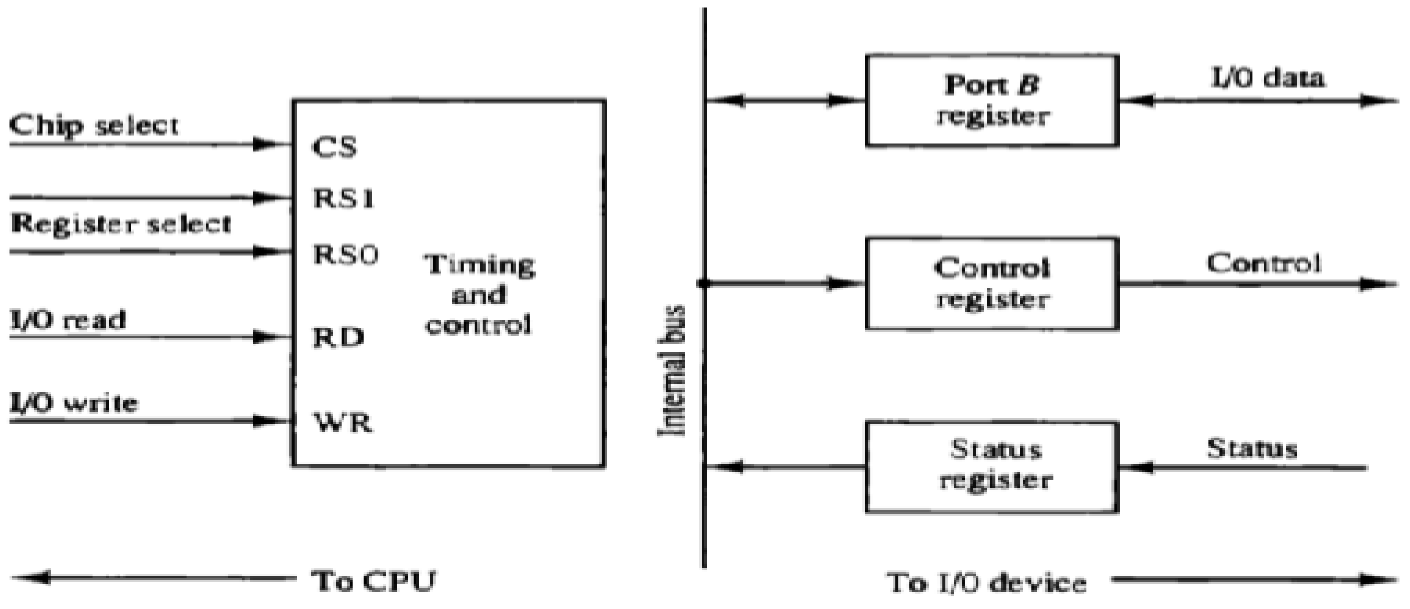


I/O versus Memory Bus

In addition to communicating with I/O, the processor must communicate with the memory unit. Like the I/O bus, the memory bus contains data, address, and read/write control lines. There are three ways that computer buses can be used to communicate with memory and I/O:

1. Use two separate buses, one for memory and the other for I/O.
2. Use one common bus for both memory and I/O but have separate control lines for each.
3. Use one common bus for memory and I/O with common control lines.

IO interface

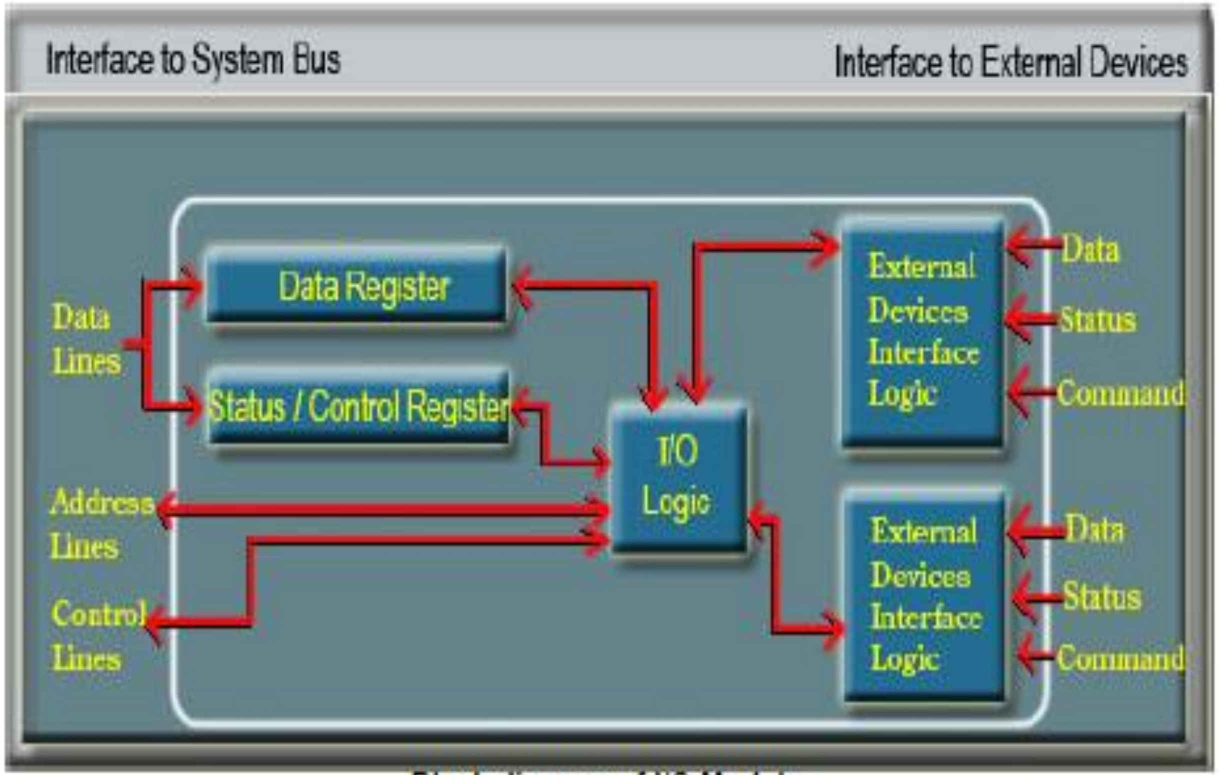


CS	RS1	RS0	Register selected
0	×	×	None: data bus in high–impedance
1	0	0	Port *A* register
1	0	1	Port *B* register
1	1	0	Control register
1	1	1	Status register

Figure 11-2 Example of I/O interface unit.

I/O Module

Block diagram of I/O Module is shown in the figure.



Interface to System Bus
Interface to External Devices
Data Lines
Address Lines
Control Lines
Data Register
Status / Control Register
I/O Logic
External Devices Interface Logic
Data
Status
Command
External Devices Interface Logic
Data
Status
Command

Block diagram of I/O Module.

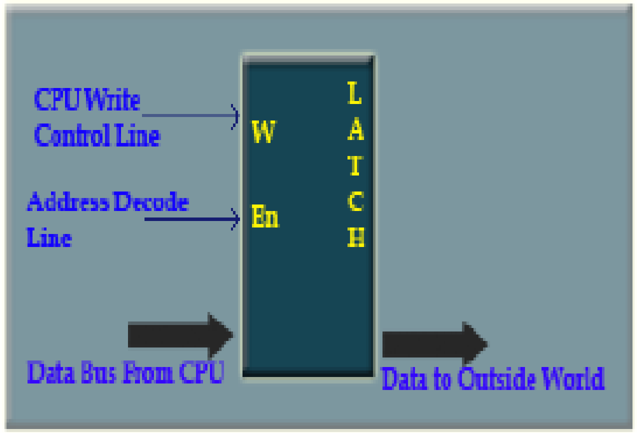
Input / Output Port

Input/Output Port

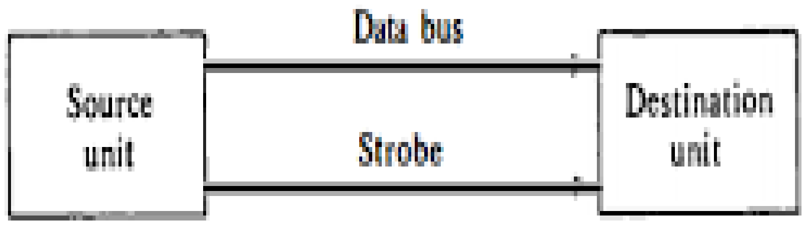
An *I/O port* is a device that looks like a memory cell to the computer but contains connection to the outside world.

An *I/O port* typically uses a *latch*. When the CPU writes to the address associated with the latch, the latch device captures the data and makes it available on a set of wires external to the CPU and memory system.

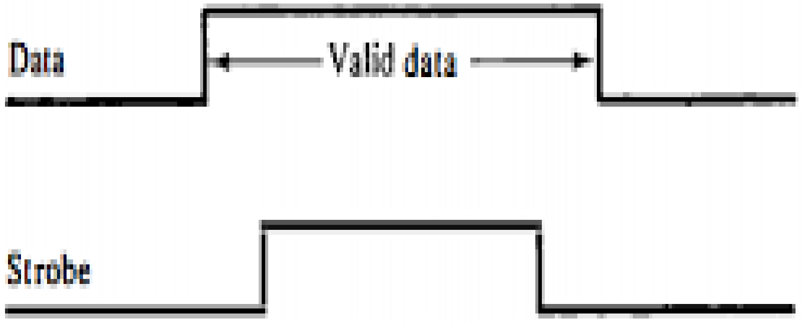
The *I/O ports* can be *read-only*, *write-only*, or *read/write*. The *write-only* port is shown in the figure.



Asynchronous Data Transfer

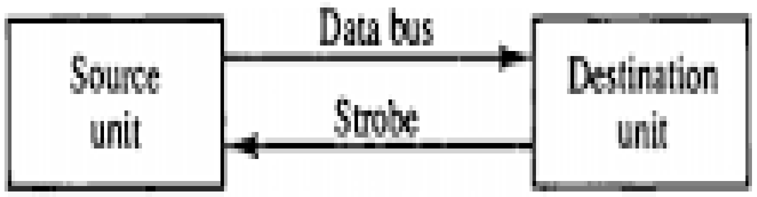


(a) Block diagram

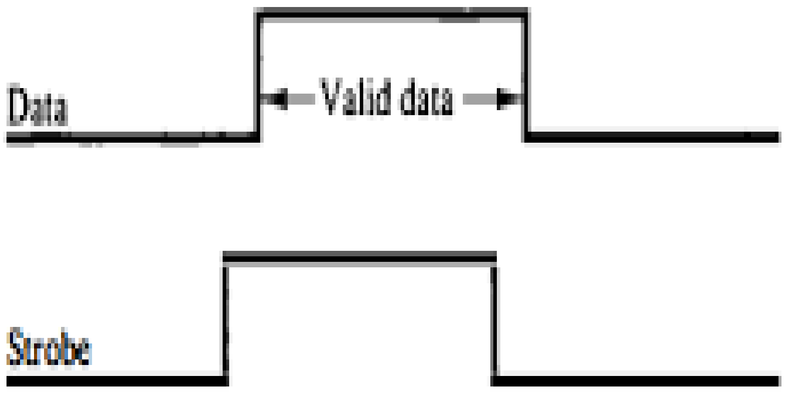


(b) Timing diagram

Figure 3 Source-initiated strobe for data transfer.



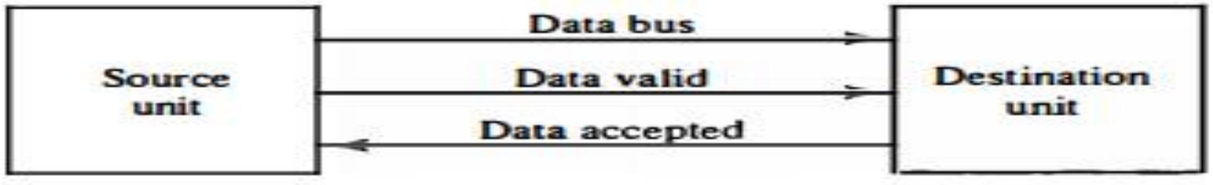
(a) Block diagram



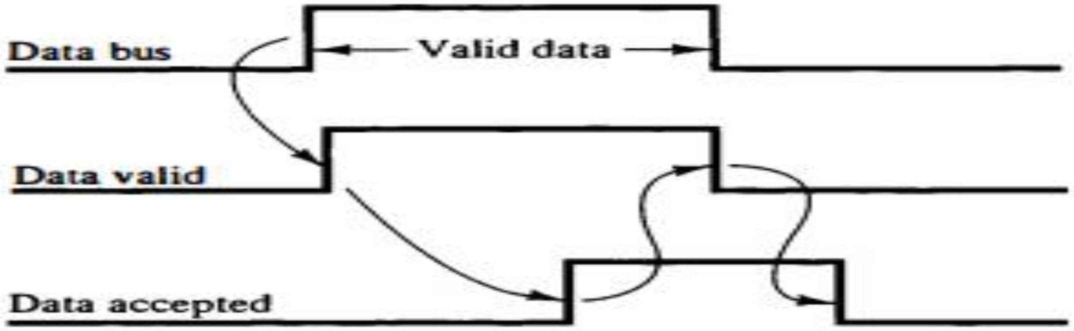
(b) Timing diagram

Figure 4 Destination-initiated strobe for data transfer.

Asynchronous Data Transfer



(a) Block diagram



(b) Timing diagram

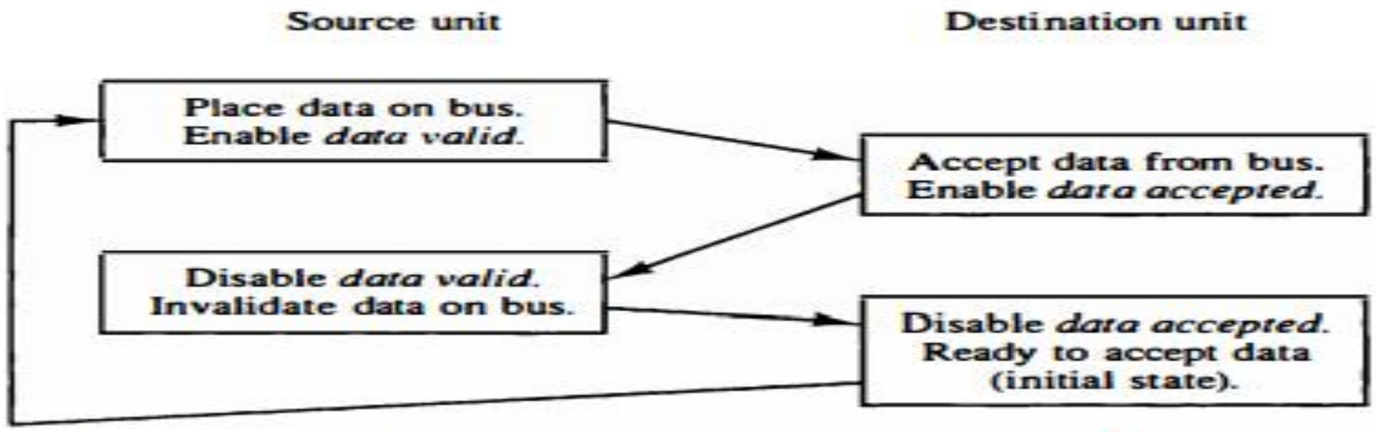
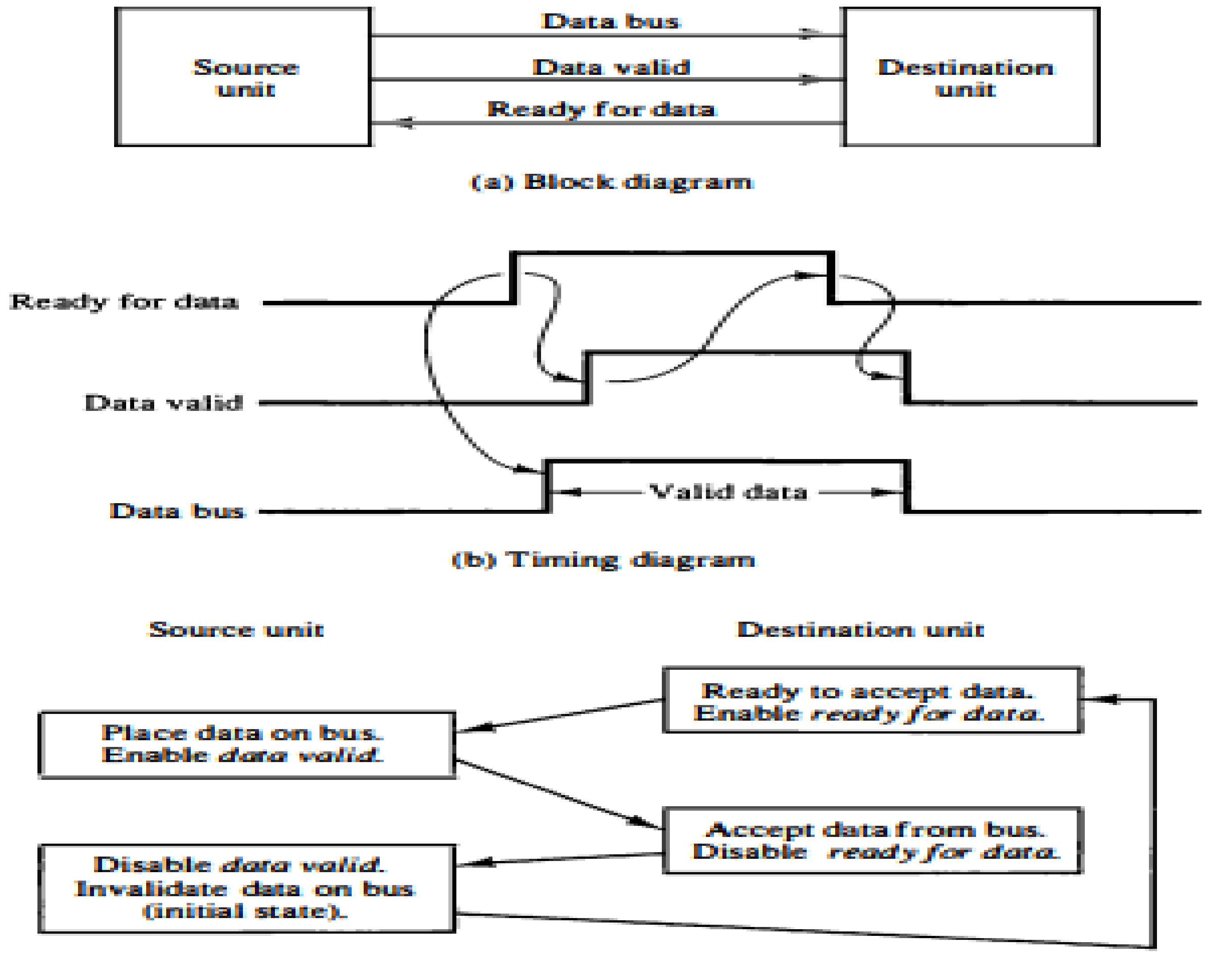


Figure -5 Source-initiated transfer using handshaking.

Asynchronous Data Transfer

Figure 6 Destination-initiated transfer using handshaking.



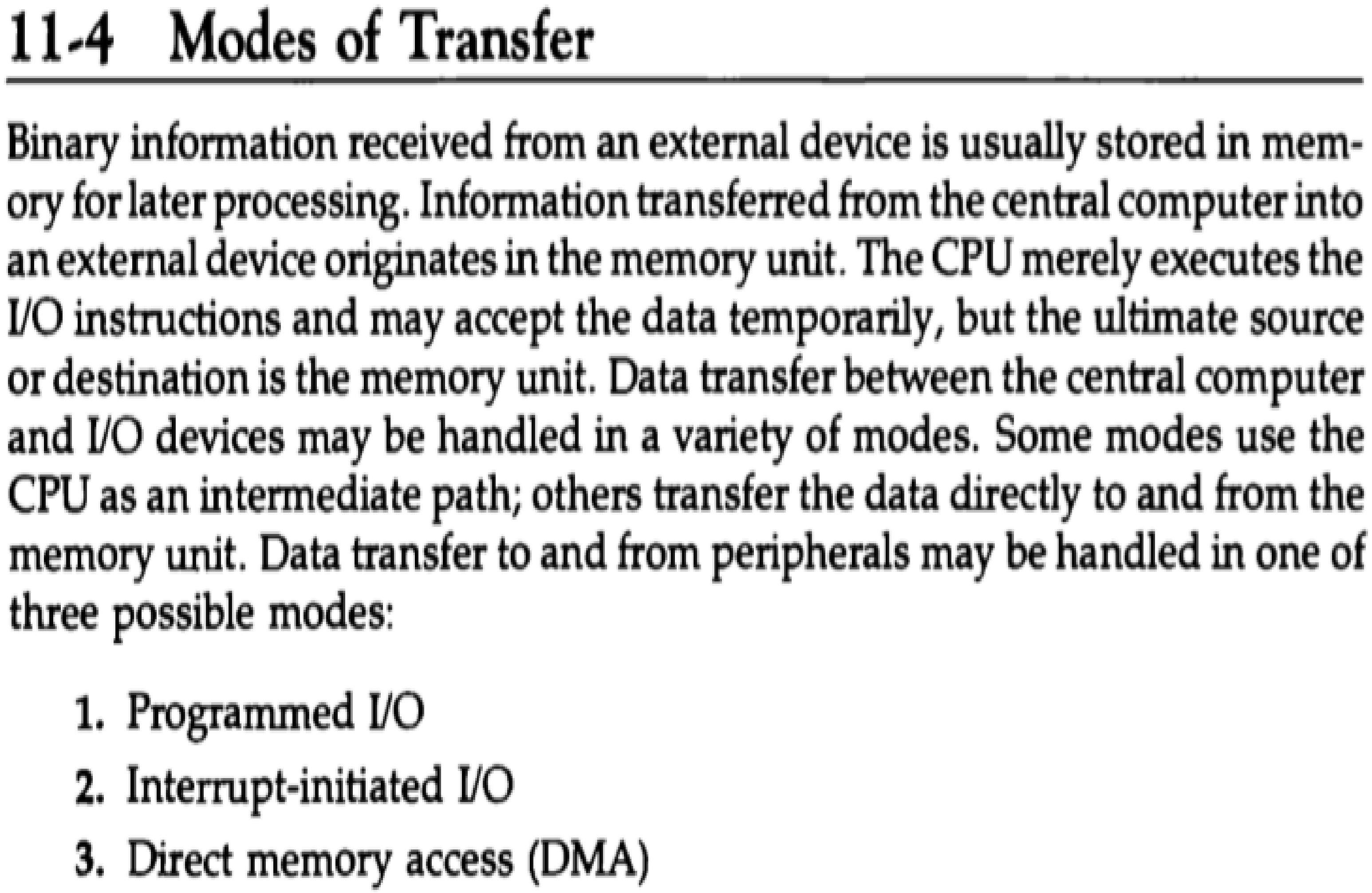
Data bus
Source unit
Data valid
Destination unit
Ready for data
Ready for data
Data valid
Data bus
Valid data
Source unit
Destination unit
Ready to accept data. Enable *ready for data.*
Place data on bus. Enable *data valid.*
Accept data from bus. Disable *ready for data.*
Disable *data valid.* Invalidate data on bus (initial state).

(a) Block diagram

(b) Timing diagram

(c) Sequence of events

Modes of Transfer

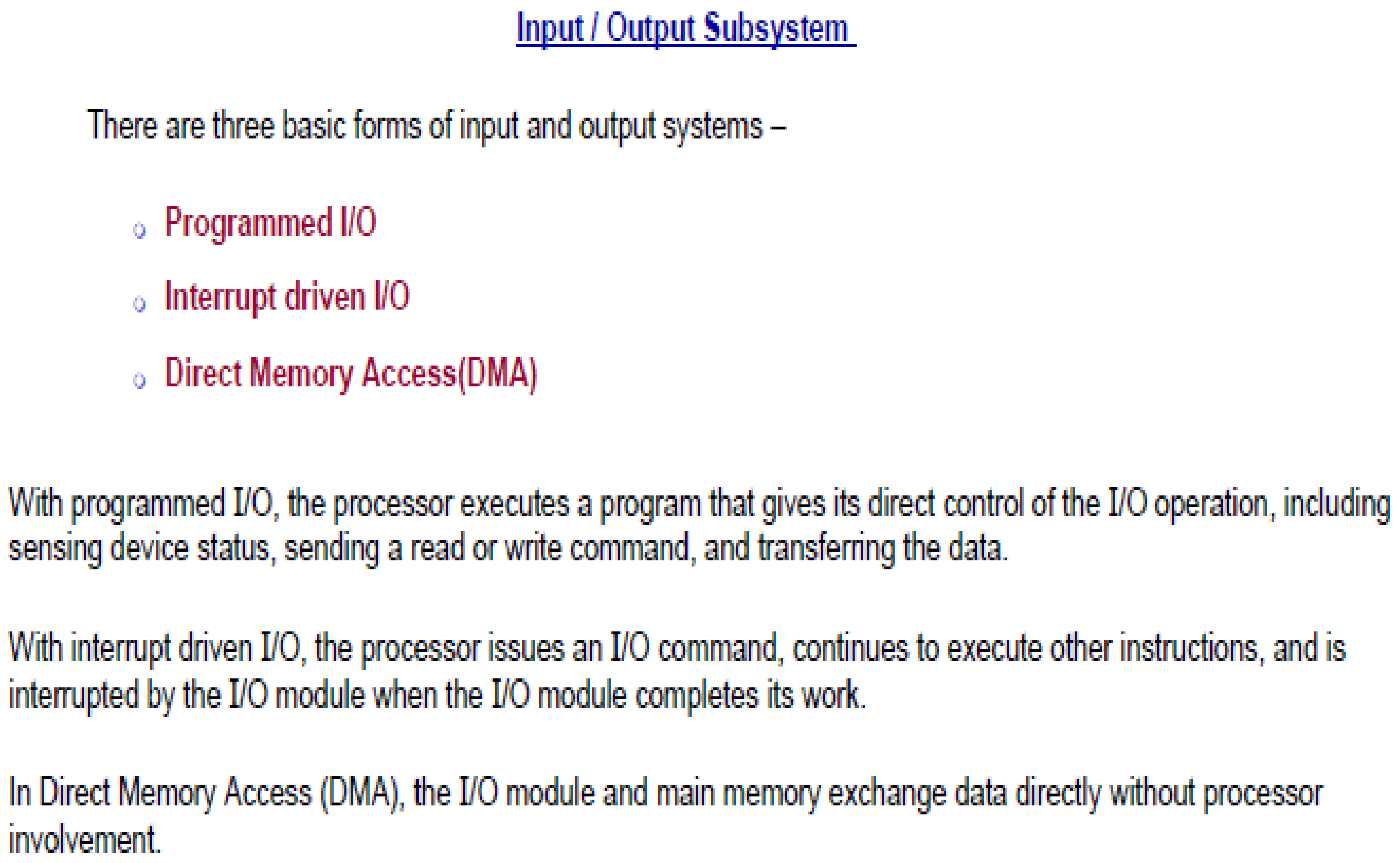


11-4 Modes of Transfer

Binary information received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in the memory unit. The CPU merely executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit. Data transfer between the central computer and I/O devices may be handled in a variety of modes. Some modes use the 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:

1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

Modes of Transfer



Input / Output Subsystem

There are three basic forms of input and output systems –

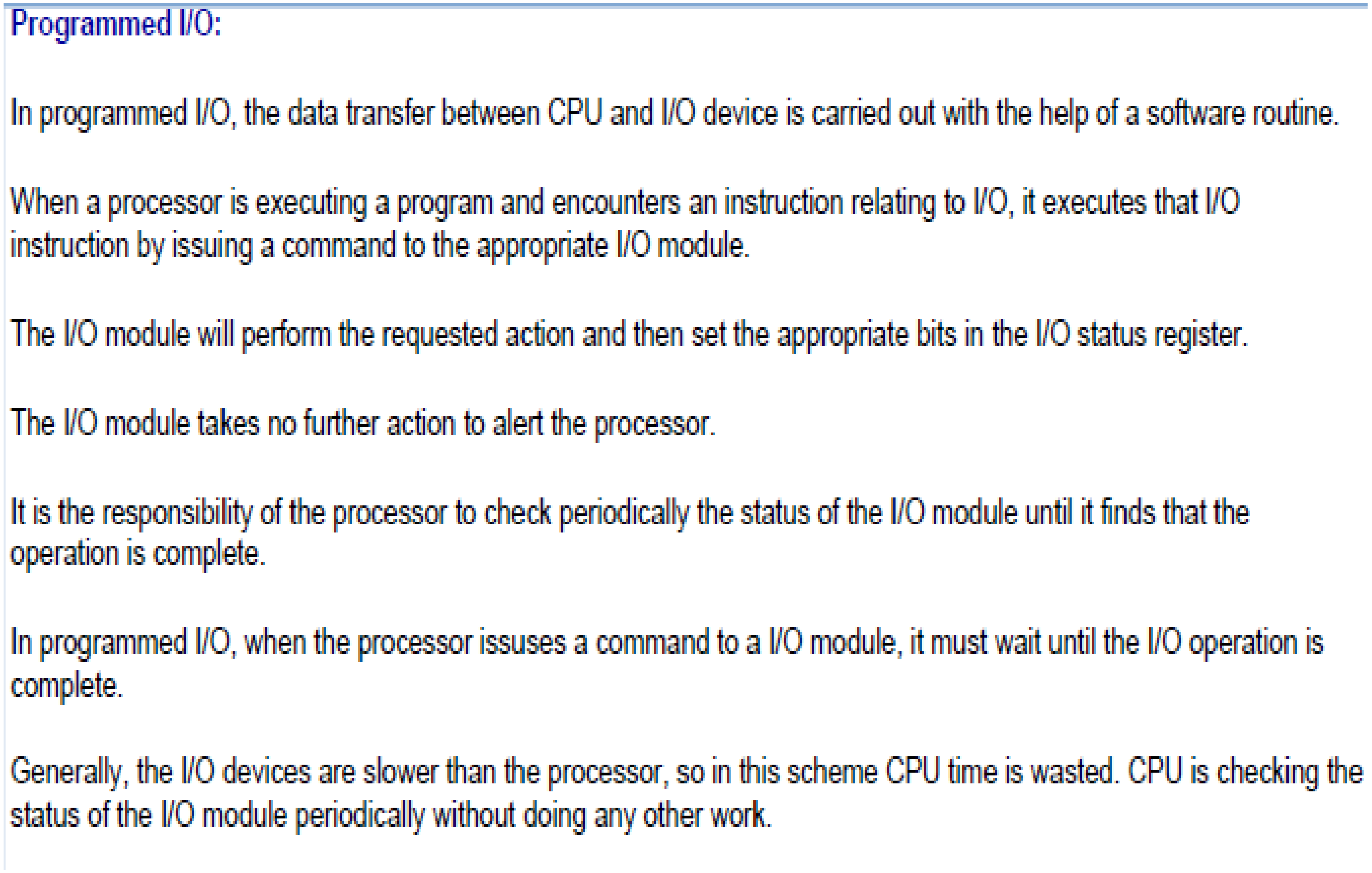
- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access(DMA)

With programmed I/O, the processor executes a program that gives its direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.

With interrupt driven I/O, the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the I/O module completes its work.

In Direct Memory Access (DMA), the I/O module and main memory exchange data directly without processor involvement.

Programmed I/O



Programmed I/O:

In programmed I/O, the data transfer between CPU and I/O device is carried out with the help of a software routine.

When a processor is executing a program and encounters an instruction relating to I/O, it executes that I/O instruction by issuing a command to the appropriate I/O module.

The I/O module will perform the requested action and then set the appropriate bits in the I/O status register.

The I/O module takes no further action to alert the processor.

It is the responsibility of the processor to check periodically the status of the I/O module until it finds that the operation is complete.

In programmed I/O, when the processor issuses a command to a I/O module, it must wait until the I/O operation is complete.

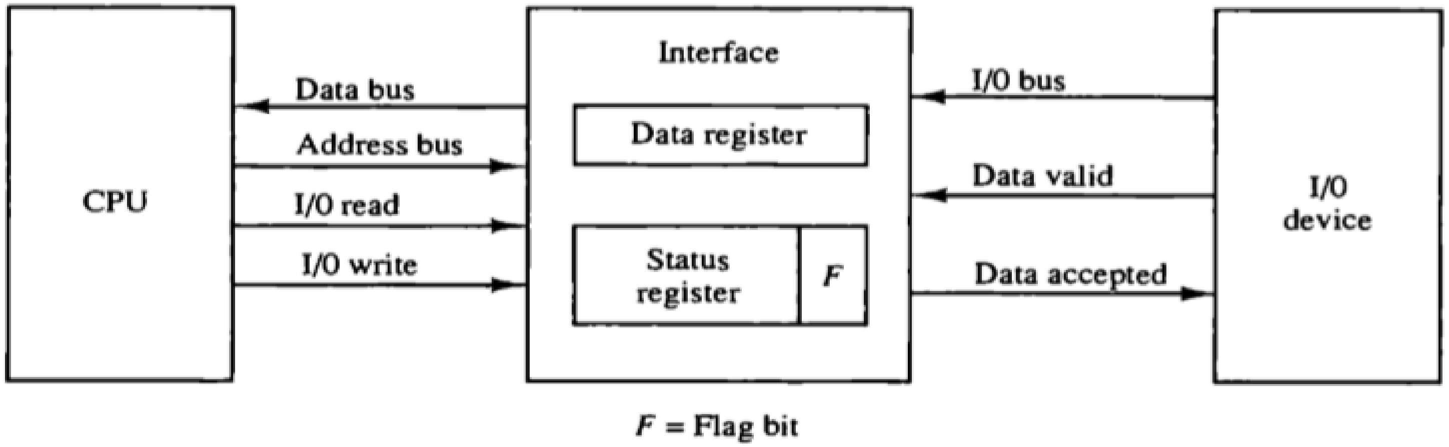
Generally, the I/O devices are slower than the processor, so in this scheme CPU time is wasted. CPU is checking the status of the I/O module periodically without doing any other work.

Programmed I/O

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. A program that

Figure 11-10 Data transfer from I/O device to CPU.



Programmed I/O

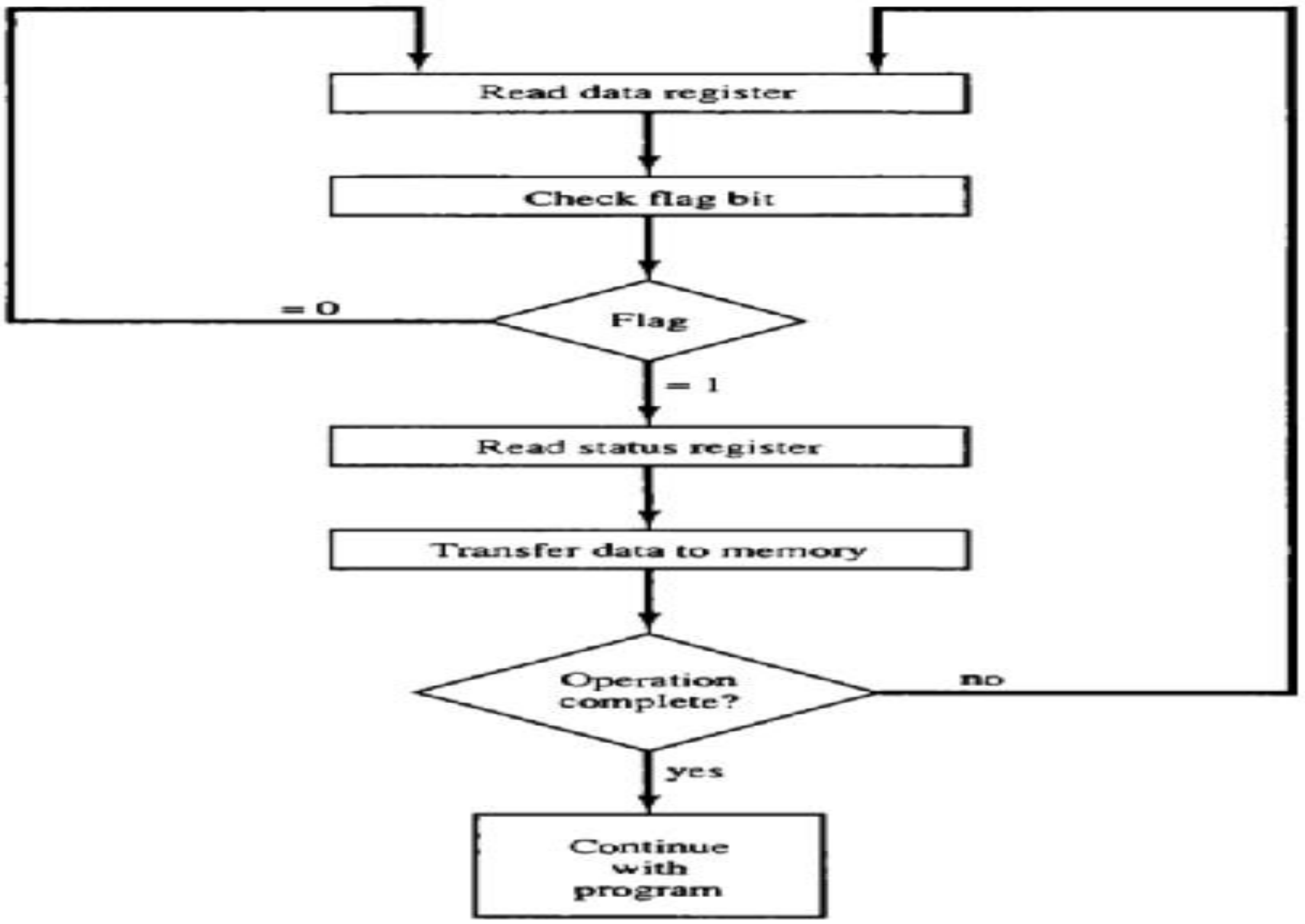


Figure 11-11 Flowchart for CPU program to input data.

Interrupt driven I/O

Interrupt driven 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.

This type of I/O operation, where the CPU constantly tests a part to see if data is available, is polling, that is, the CPU Polls (asks) the port if it has data available or if it is capable of accepting data. Polled I/O is inherently inefficient.

The solution to this problem is to provide an interrupt mechanism. In this approach the processor issues an I/O command to a module and then go on to do some other useful work. The I/O module then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer. Once the data transfer is over, the processor then resumes its former processing.

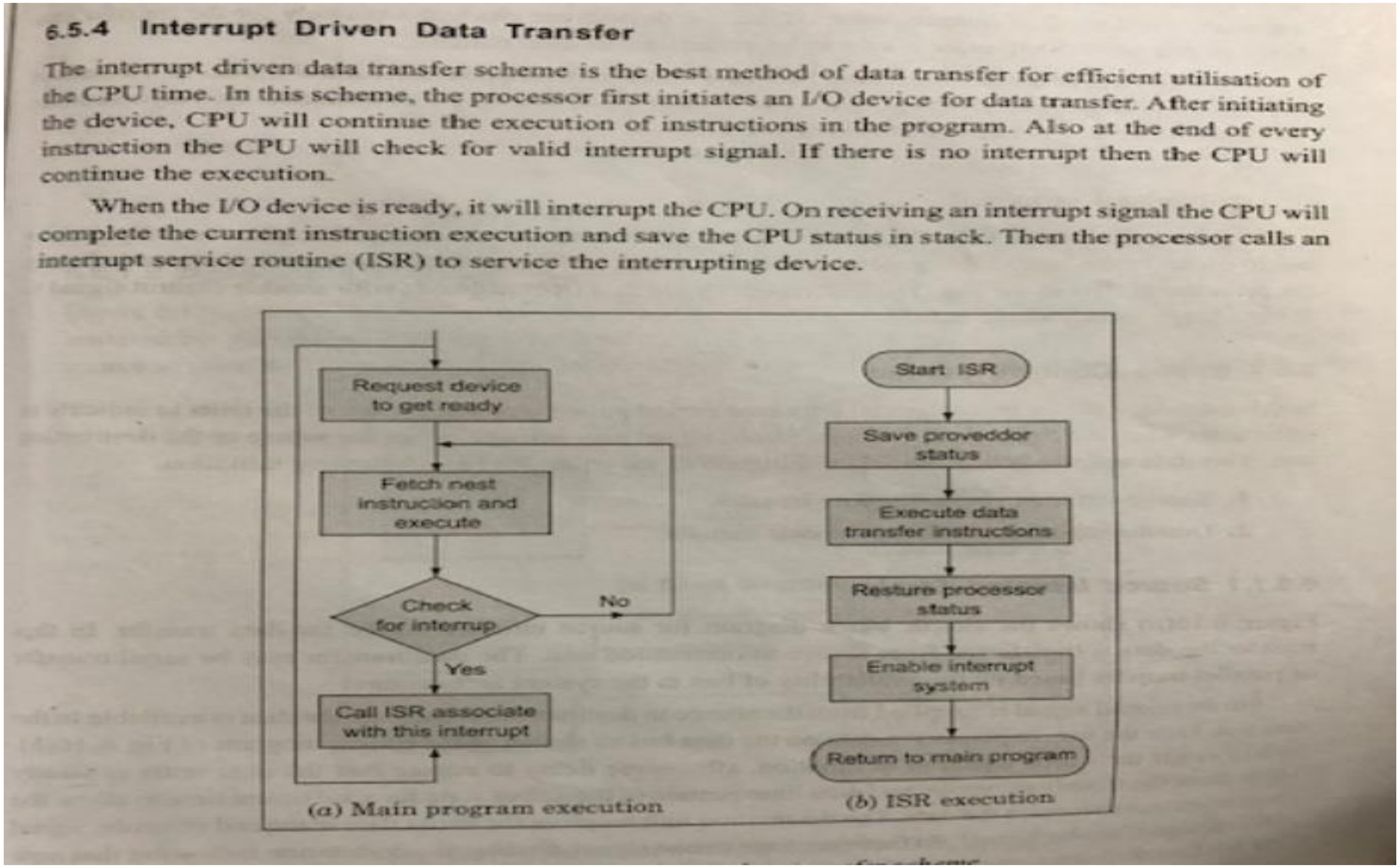
Interrupt driven I/O

Let us consider how it works

A. From the point of view of the I/O module:

- For input, the I/O module services a READ command from the processor.
- The I/O module then proceeds to read data from an associated peripheral device.
- Once the data are in the modules data register, the module issues an interrupt to the processor over a control line.
- The module then waits until its data are requested by the processor.
- When the request is made, the module places its data on the data bus and is then ready for another I/O operation.

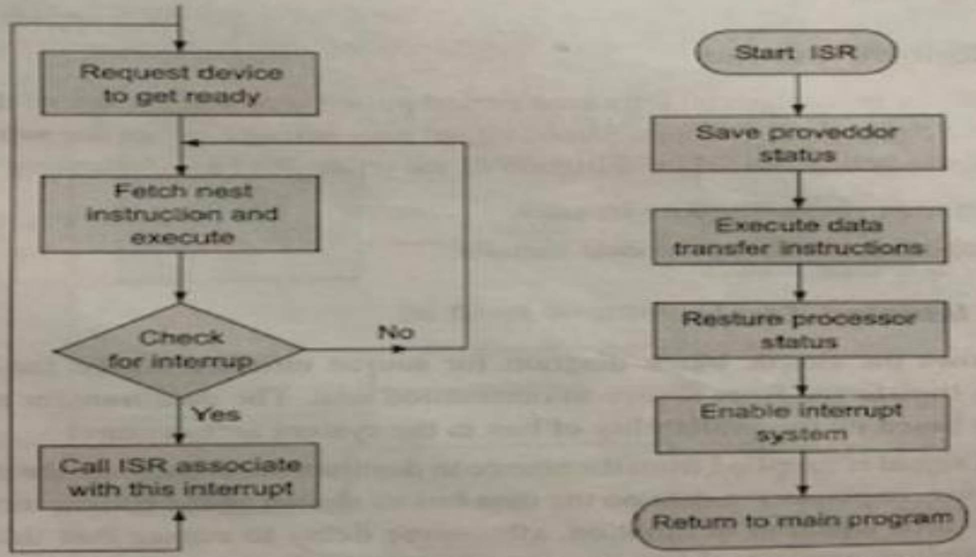
Interrupt driven I/O



6.5.4 Interrupt Driven Data Transfer

The interrupt driven data transfer scheme is the best method of data transfer for efficient utilisation of the CPU time. In this scheme, the processor first initiates an I/O device for data transfer. After initiating the device, CPU will continue the execution of instructions in the program. Also at the end of every instruction the CPU will check for valid interrupt signal. If there is no interrupt then the CPU will continue the execution.

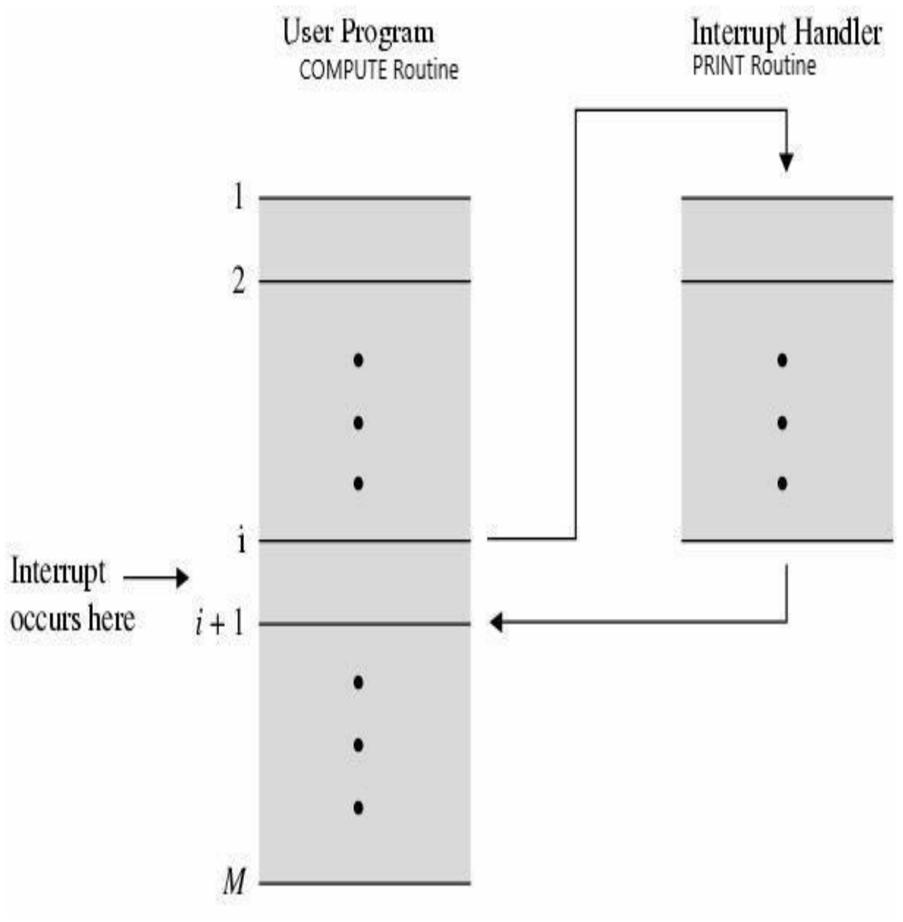
When the I/O device is ready, it will interrupt the CPU. On receiving an interrupt signal the CPU will complete the current instruction execution and save the CPU status in stack. Then the processor calls an interrupt service routine (ISR) to service the interrupting device.

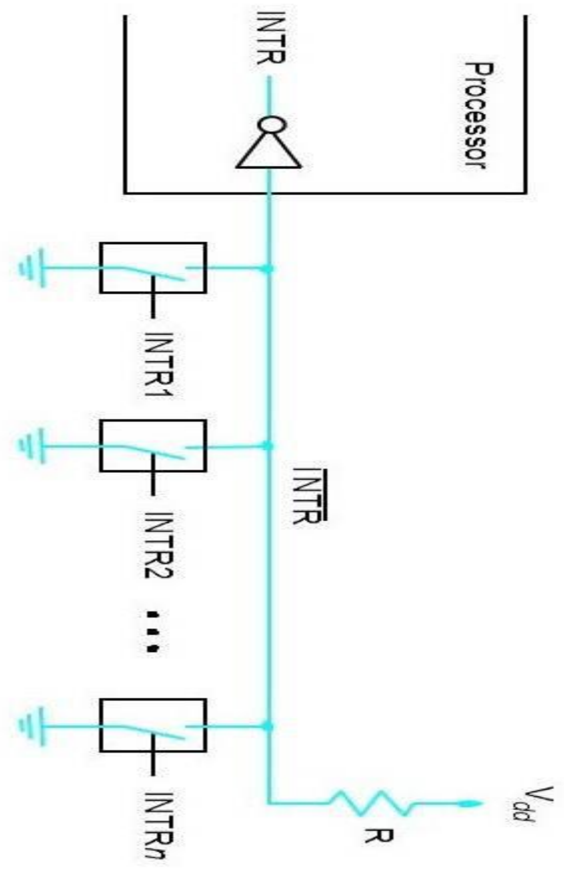


Request device to get ready
Fetch nest instruction and execute
Check for interrup
No
Yes
Call ISR associate with this interrupt
Start ISR
Save proveddor status
Execute data transfer instructions
Resture processor status
Enable interrupt system
Return to main program

(a) Main program execution (b) ISR execution

Interrupt



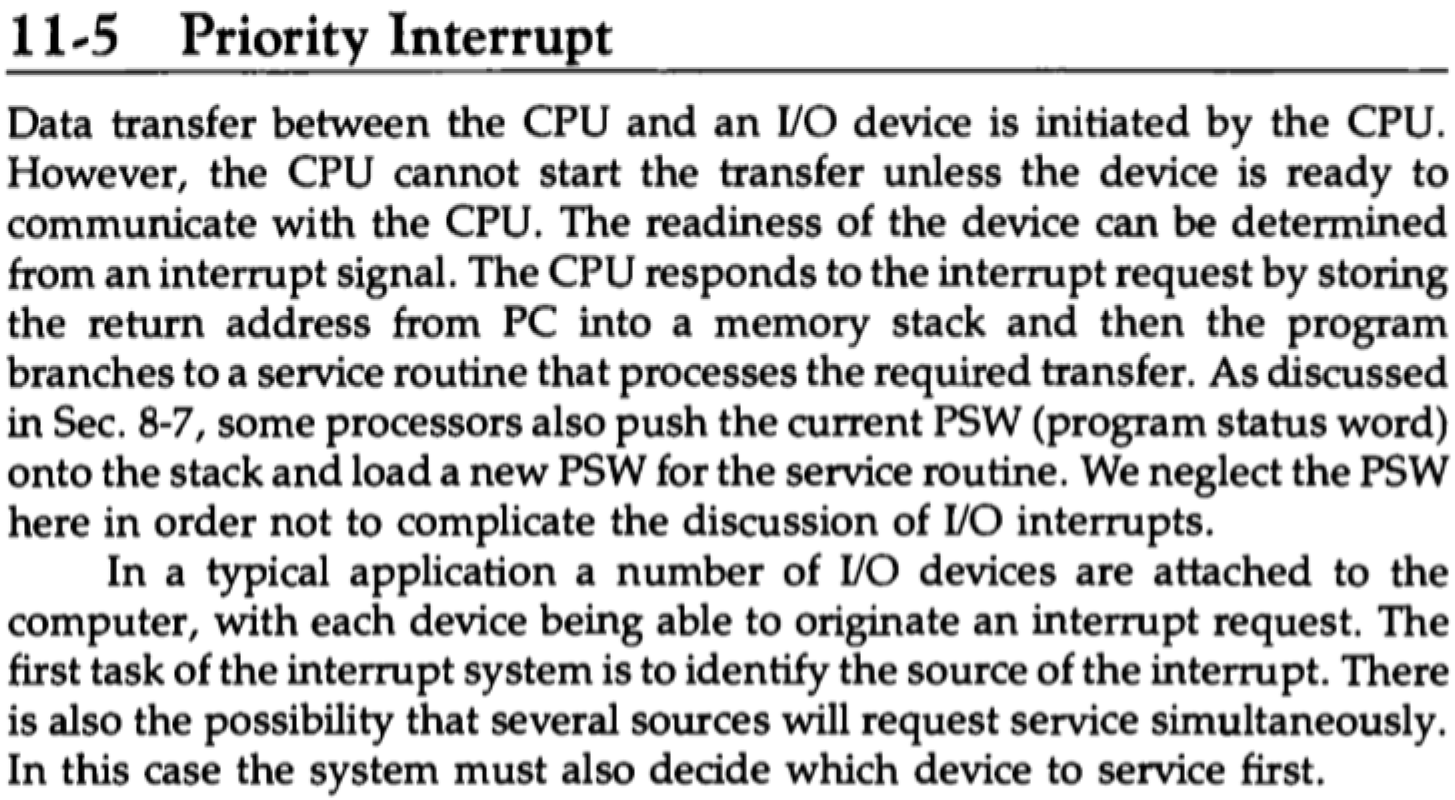


Interrupt service routine

Interrupt Hardware

NIET Greater Noida
GET FUTURE READY

Priority Interrupt



11-5 Priority Interrupt

Data transfer between the CPU and an I/O device is initiated by the CPU. However, the CPU cannot start the transfer unless the device is ready to communicate with the CPU. The readiness of the device can be determined from an interrupt signal. The CPU responds to the interrupt request by storing the return address from PC into a memory stack and then the program branches to a service routine that processes the required transfer. As discussed in Sec. 8-7, some processors also push the current PSW (program status word) onto the stack and load a new PSW for the service routine. We neglect the PSW here in order not to complicate the discussion of I/O interrupts.

In a typical application a number of I/O devices are attached to the computer, with each device being able to originate an interrupt request. The first task of the interrupt system is to identify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system must also decide which device to service first.

Priority Interrupt

Daisy Chain :

In this method for interrupts all I/O modules share a common interrupt request lines. However the interrupt acknowledge line is connected in a daisy chain fashion. When the processor senses an interrupt, it sends out an interrupt acknowledgement.

The interrupt acknowledge signal propagates through a series of I/O module until it gets to a requesting module.

The requesting module typically responds by placing a word on the data lines. This word is referred to as a vector and is either the address of the I/O module or some other unique identification.

In either case, the processor uses the vector as a pointer to the appropriate device service routine. This avoids the need to execute a general interrupt service routine first. This technique is referred to as a *vectored interrupt*.

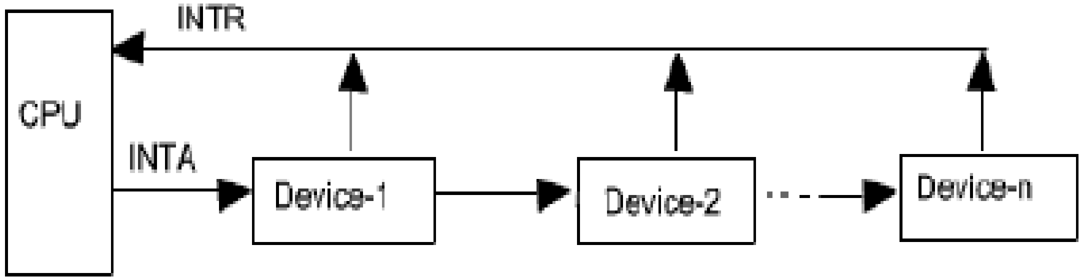


Fig : Daisy chain arrangement

Priority Interrupt

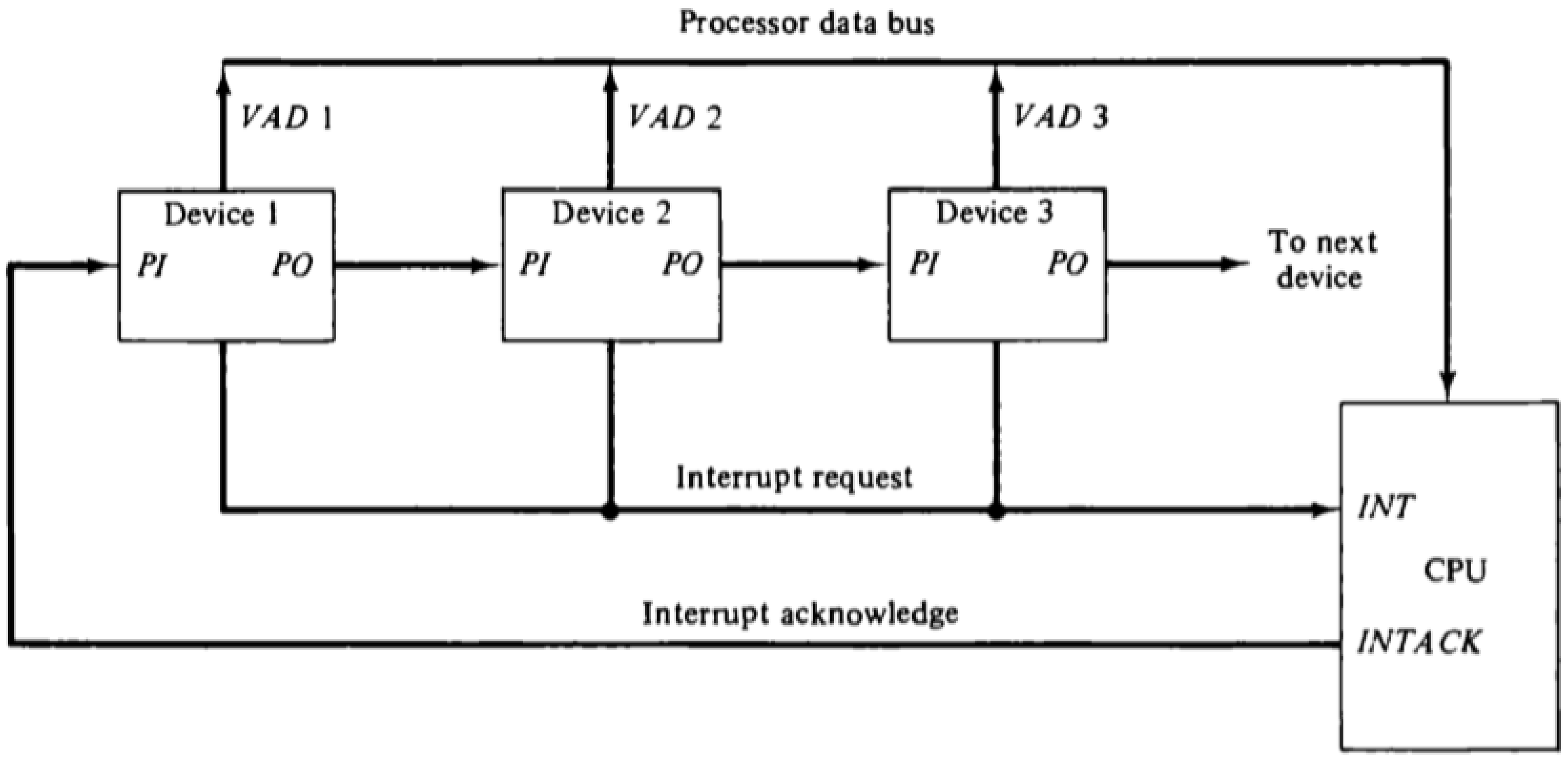
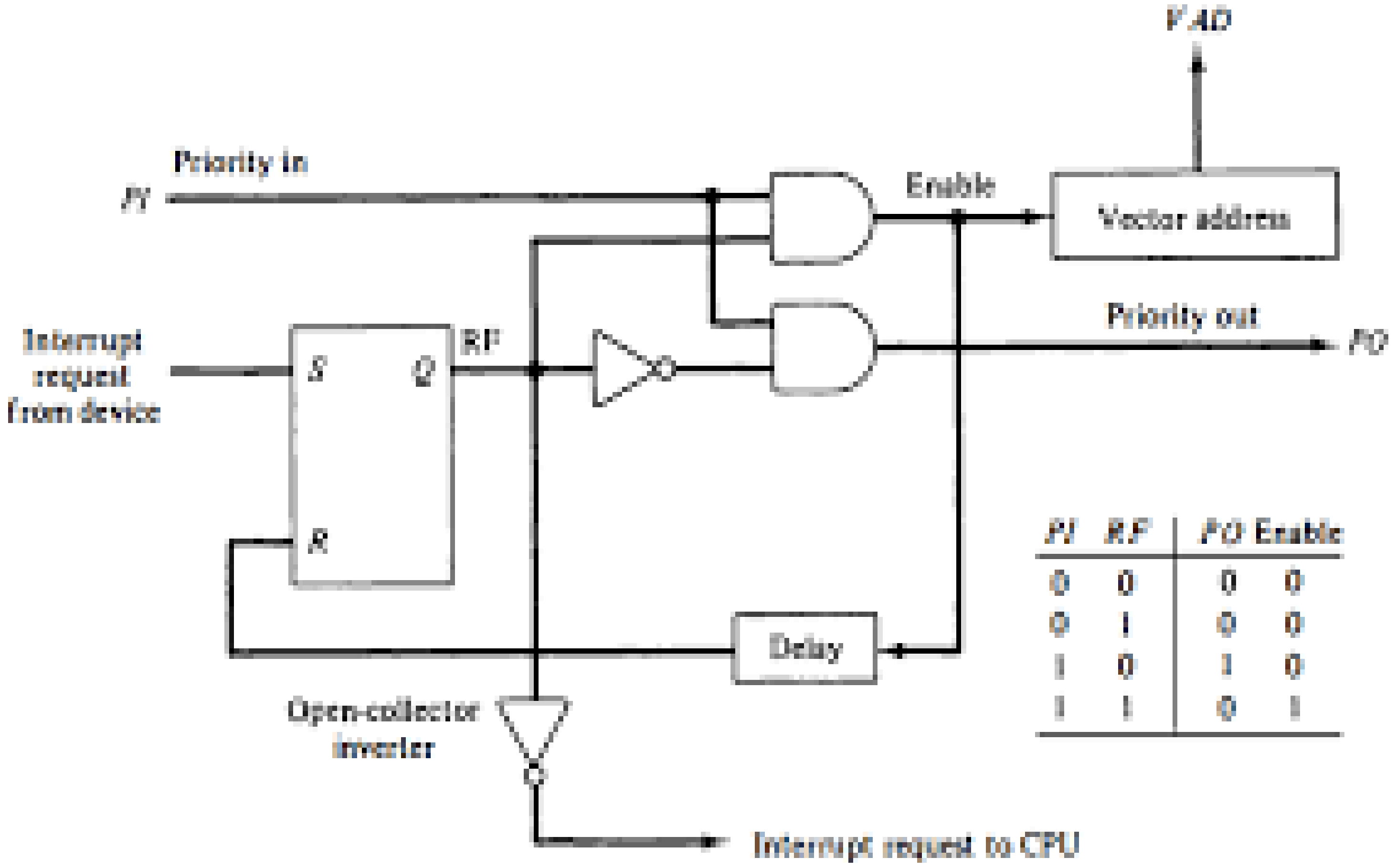


Figure 11-12 Daisy-chain priority interrupt.

Priority Interrupt



PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Figure 13 One stage of the daisy-chain priority arrangement.

Priority Interrupt

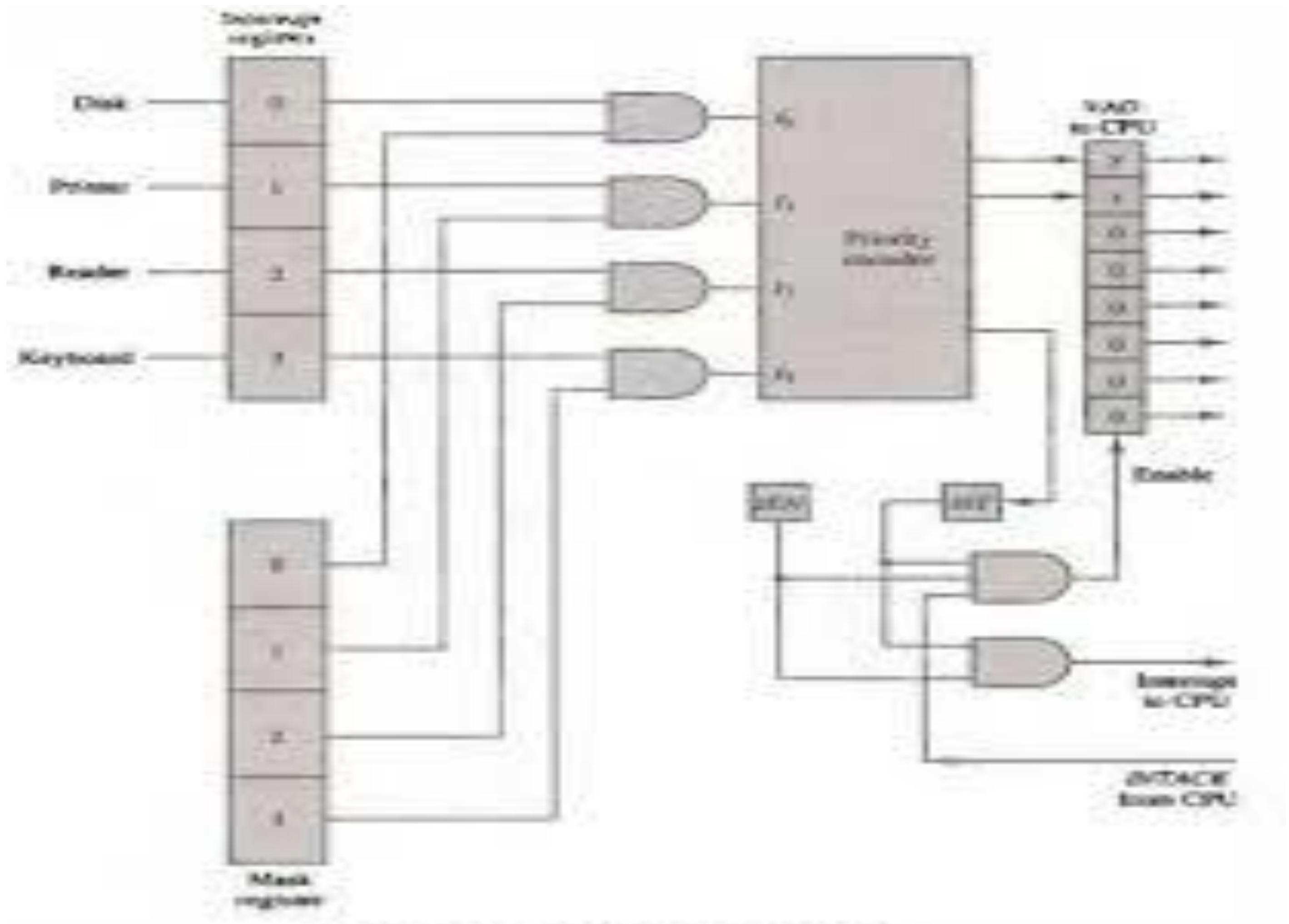
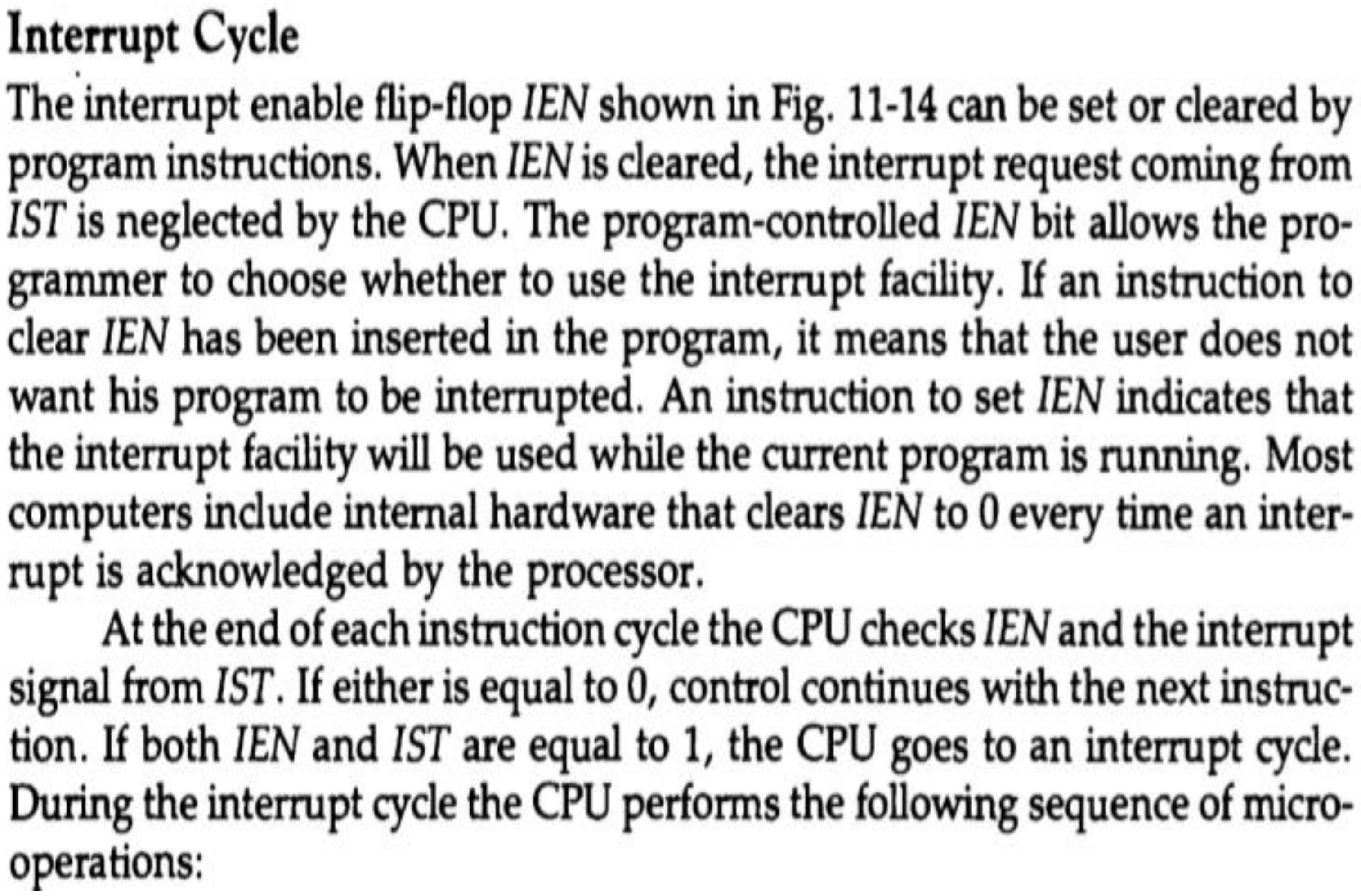


Figure 14 Priority interrupt hardware.

Priority Interrupt



Interrupt Cycle

The interrupt enable flip-flop *IEN* shown in Fig. 11-14 can be set or cleared by program instructions. When *IEN* is cleared, the interrupt request coming from *IST* is neglected by the CPU. The program-controlled *IEN* bit allows the programmer to choose whether to use the interrupt facility. If an instruction to clear *IEN* has been inserted in the program, it means that the user does not want his program to be interrupted. An instruction to set *IEN* indicates that the interrupt facility will be used while the current program is running. Most computers include internal hardware that clears *IEN* to 0 every time an interrupt is acknowledged by the processor.

At the end of each instruction cycle the CPU checks *IEN* and the interrupt signal from *IST*. If either is equal to 0, control continues with the next instruction. If both *IEN* and *IST* are equal to 1, the CPU goes to an interrupt cycle. During the interrupt cycle the CPU performs the following sequence of micro-operations:

Interrupt & Exception

In **computer architecture**, an **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

Interrupt is a signal which has highest priority from hardware or software and the processor should process its signal immediately.

TYPES OF INTERRUPTS

Interrupts have highest priority than other signals, there are many type of interrupts but basic type of interrupts are -

- **Hardware Interrupts:** If the signal for the processor is from external device or hardware is called hardware interrupts. Ex: KB Interrupt, Timer interrupt

Example: From keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts.

Interrupt & Exception

Hardware interrupts can be classified into two types –

Maskable Interrupt: The hardware interrupts which can be disabled or delayed when a higher priority interrupt has occurred to the processor. EX:RST 5.5,RST 6.5,RST 7.5 in 8085

Non Maskable Interrupt: The hardware which cannot be delayed and should process by the processor immediately .EX: RST 4.5 in 8085,power fail interrupts , real time system interrupts

•**Software Interrupts:** Software interrupt can also divided in to two types they are -

Normal Interrupts(TRAP): the interrupts which are caused by the software instructions are called software instructions. Ex: system calls

Exception: unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception. Invalid op-code, invalid memory access, page fault

TRAP

•It is non-maskable and edge and level triggered interrupt.

•TRAP has the highest priority and it is vector interrupt.

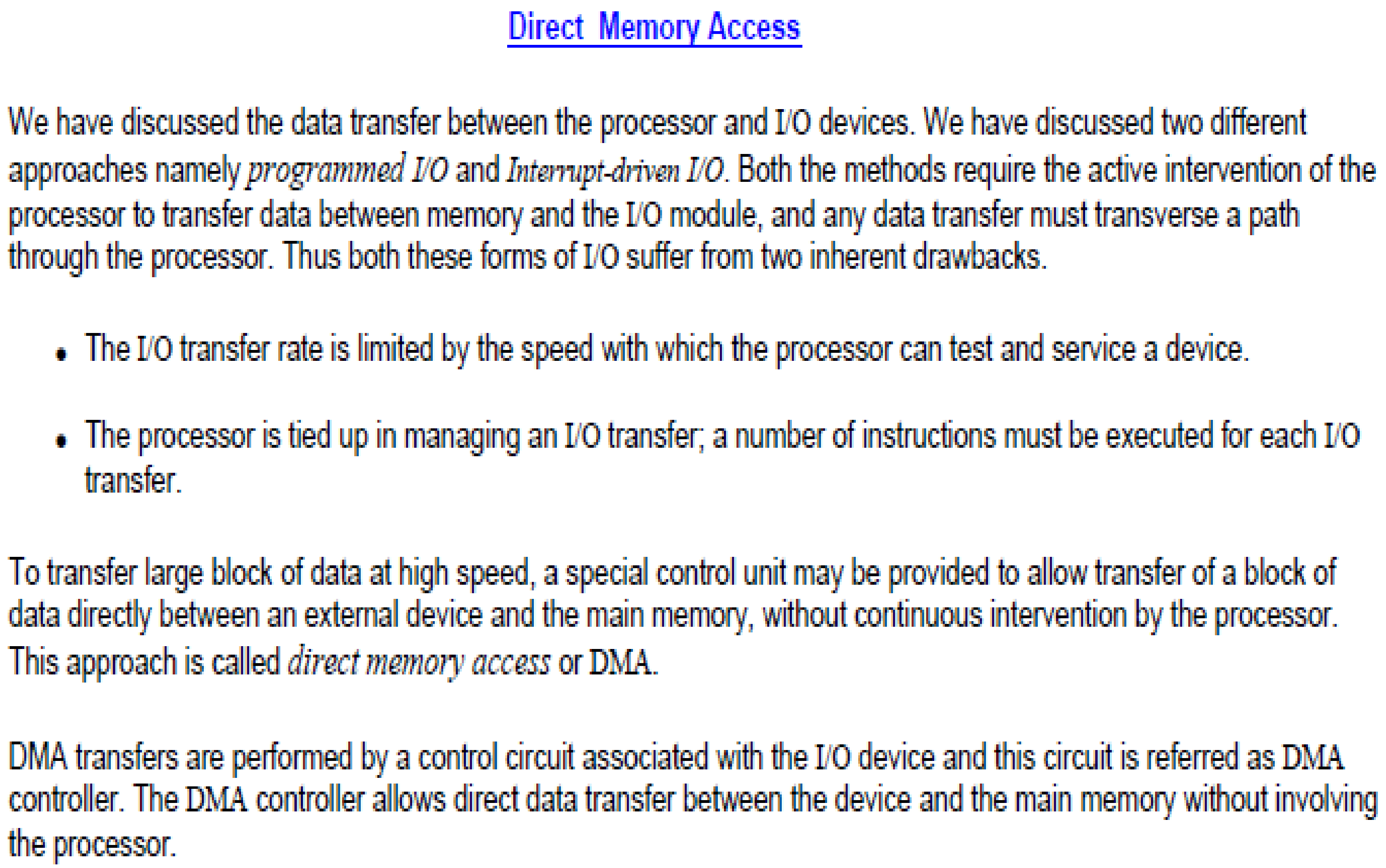
A **trap** is an exception in a user process.

•Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged.

There are two ways to clear TRAP interrupt.

1. By resetting microprocessor (External signal)
2. By giving a high TRAP ACKNOWLEDGE (Internal signal)

DMA



Direct Memory Access

We have discussed the data transfer between the processor and I/O devices. We have discussed two different approaches namely *programmed I/O* and *Interrupt-driven I/O*. Both the methods require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.

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

To transfer large block of data at high speed, a special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This approach is called *direct memory access* or DMA.

DMA transfers are performed by a control circuit associated with the I/O device and this circuit is referred as DMA controller. The DMA controller allows direct data transfer between the device and the main memory without involving the processor.

DMA

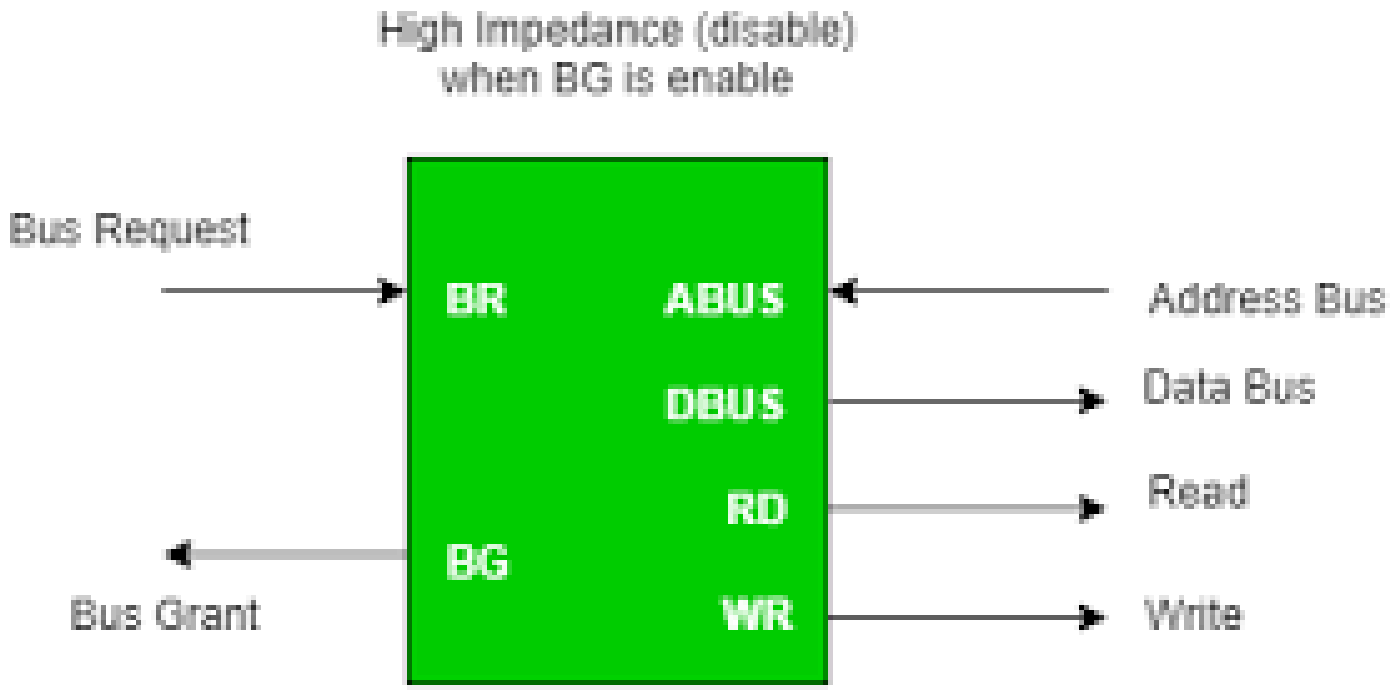
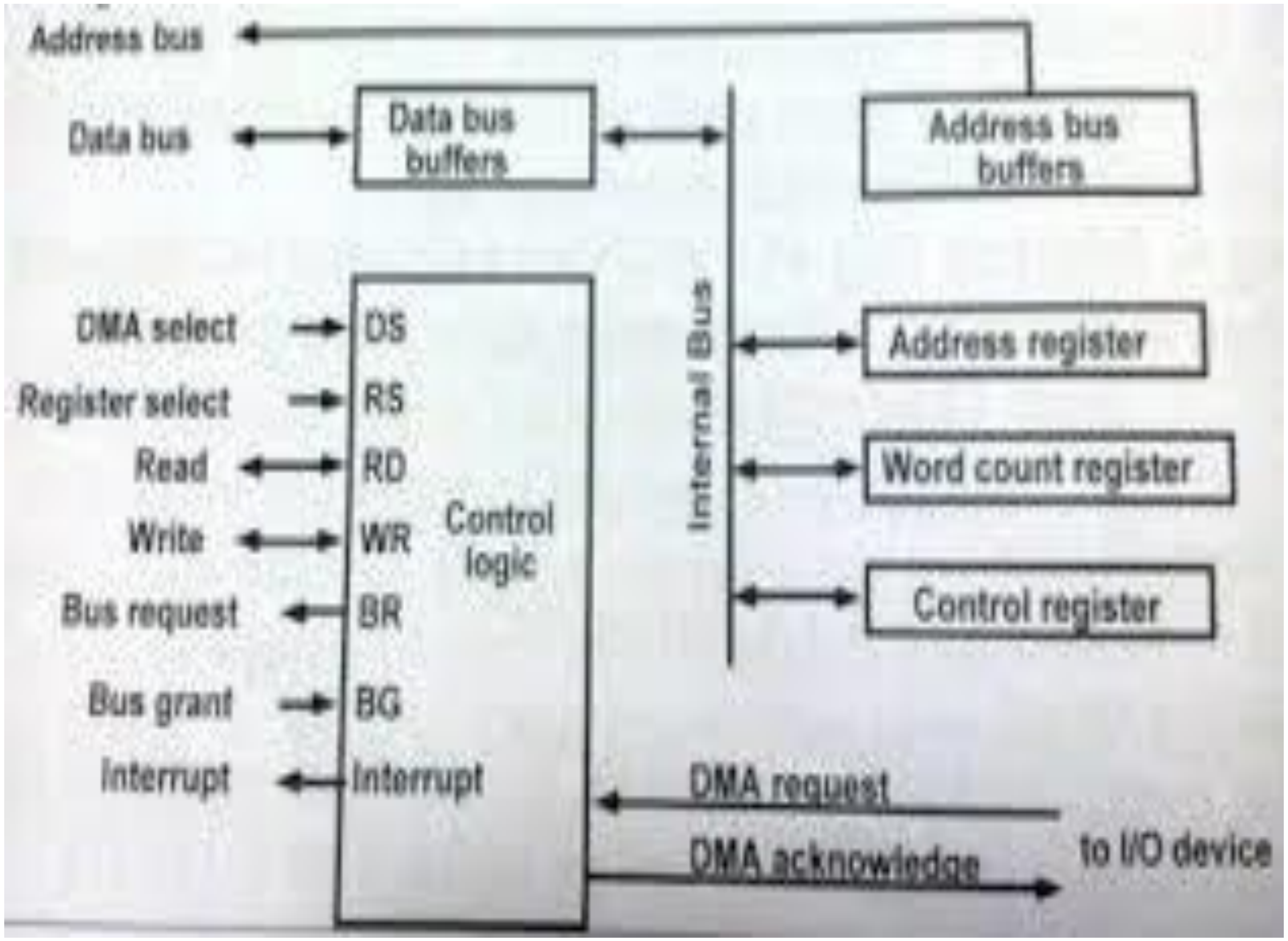


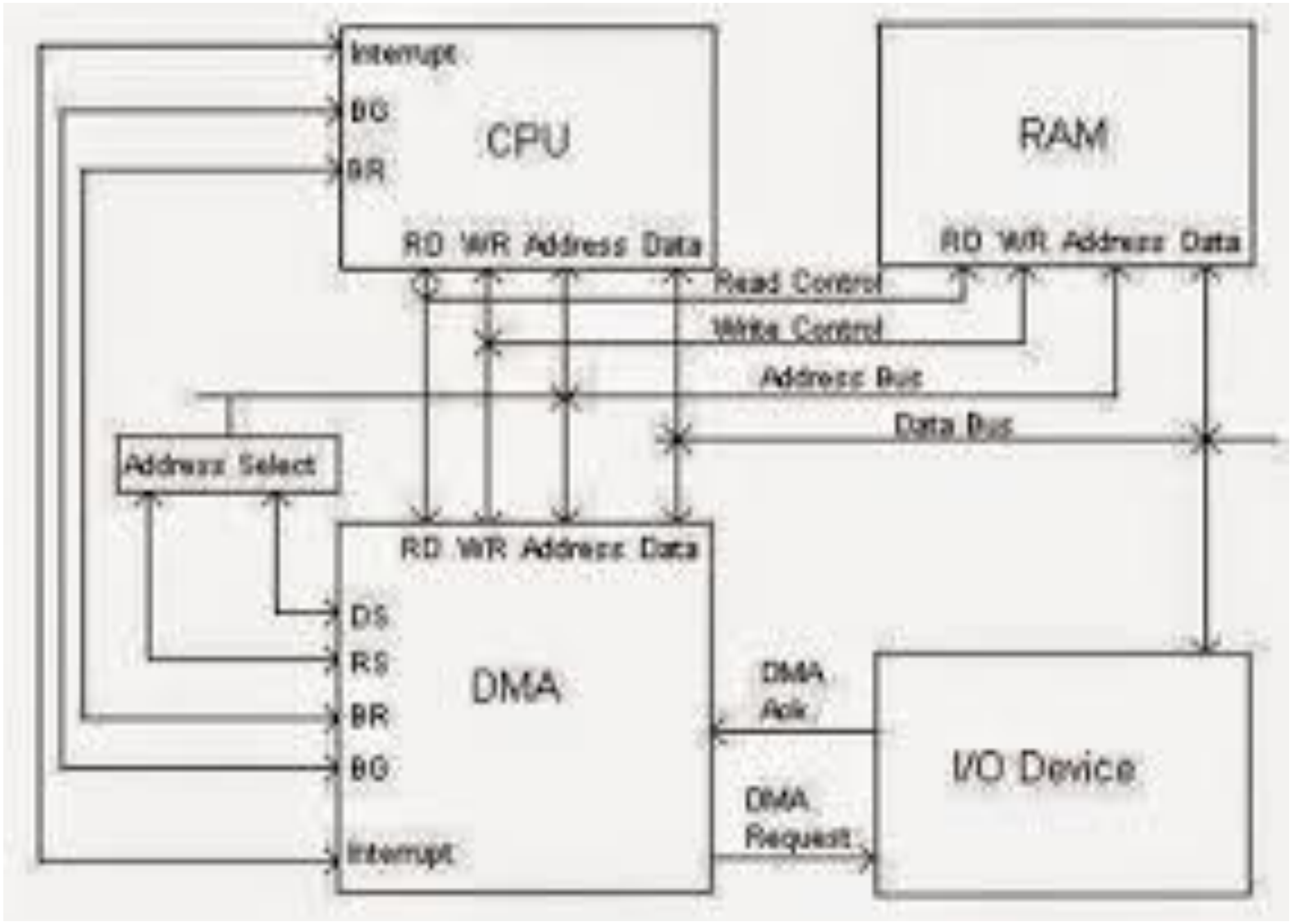
Figure - CPU Bus Signals for DMA Transfer

DMA



Address bus
Data bus
Data bus buffers
Address bus buffers
Internal Bus
DMA select
Register select
Read
Write
Bus request
Bus grant
Interrupt
DS
RS
RD
WR
BR
BG
Interrupt
Control logic
Address register
Word count register
Control register
DMA request
DMA acknowledge
to I/O device

DMA


Interrupt
BG
BR
CPU
RD WR Address Data
RAM
RD WR Address Data
Read Control
Write Control
Address Bus
Data Bus
Address Select
RD WR Address Data
DS
RS
DMA
BR
BG
Interrupt
DMA Ack
DMA Request
I/O Device

DMA

To transfer data between memory and I/O devices, DMA controller takes over the control of the system from the processor and transfer of data take place over the system bus. For this purpose, the DMA controller must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily. The later technique is more common and is referred to as cycle stealing, because the DMA module in effect steals a bus cycle.

The typical block diagram of a DMA controller is shown in the figure.

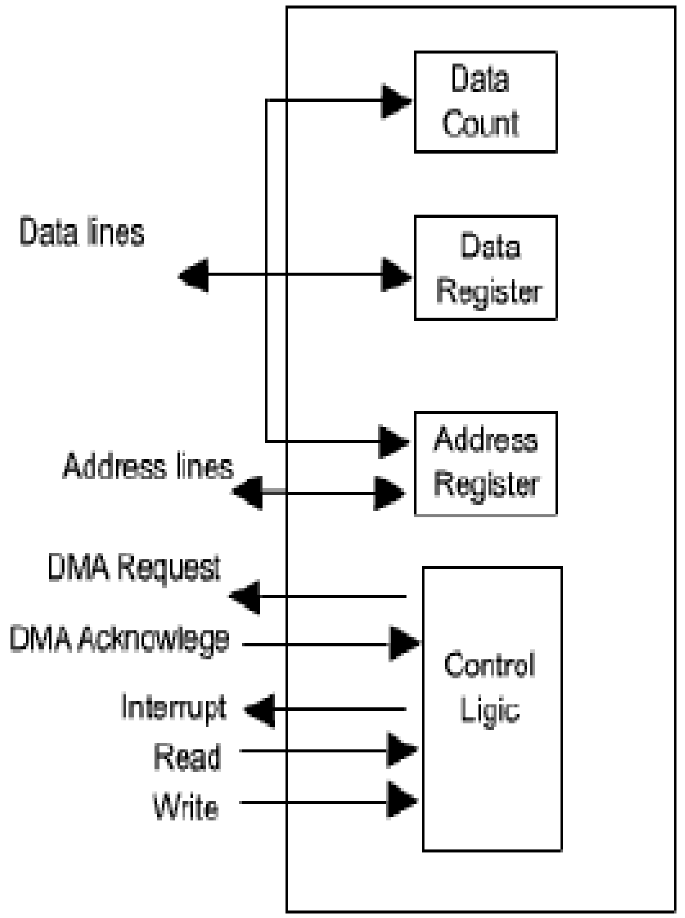


Fig : Typical DMA block diagram

IOP

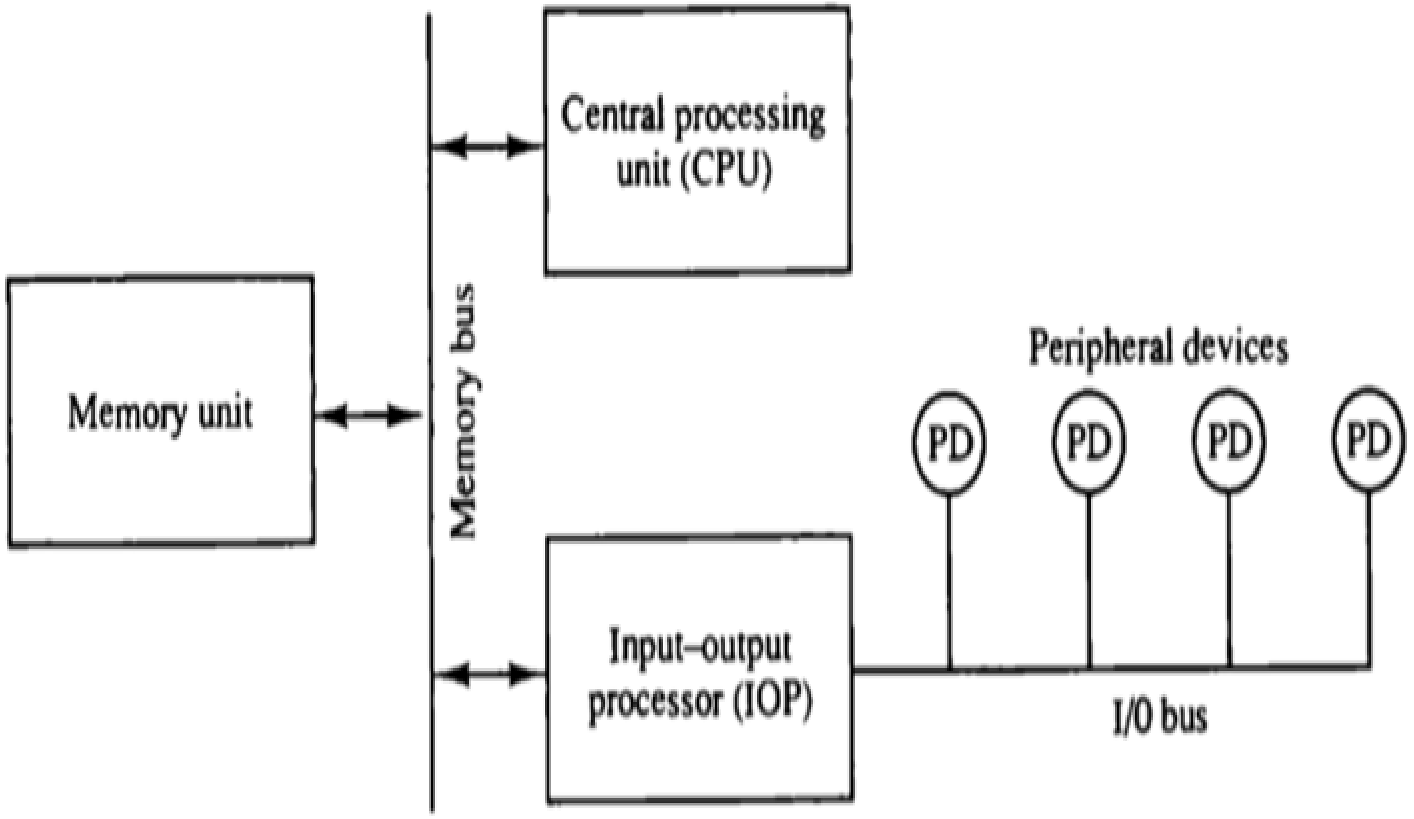
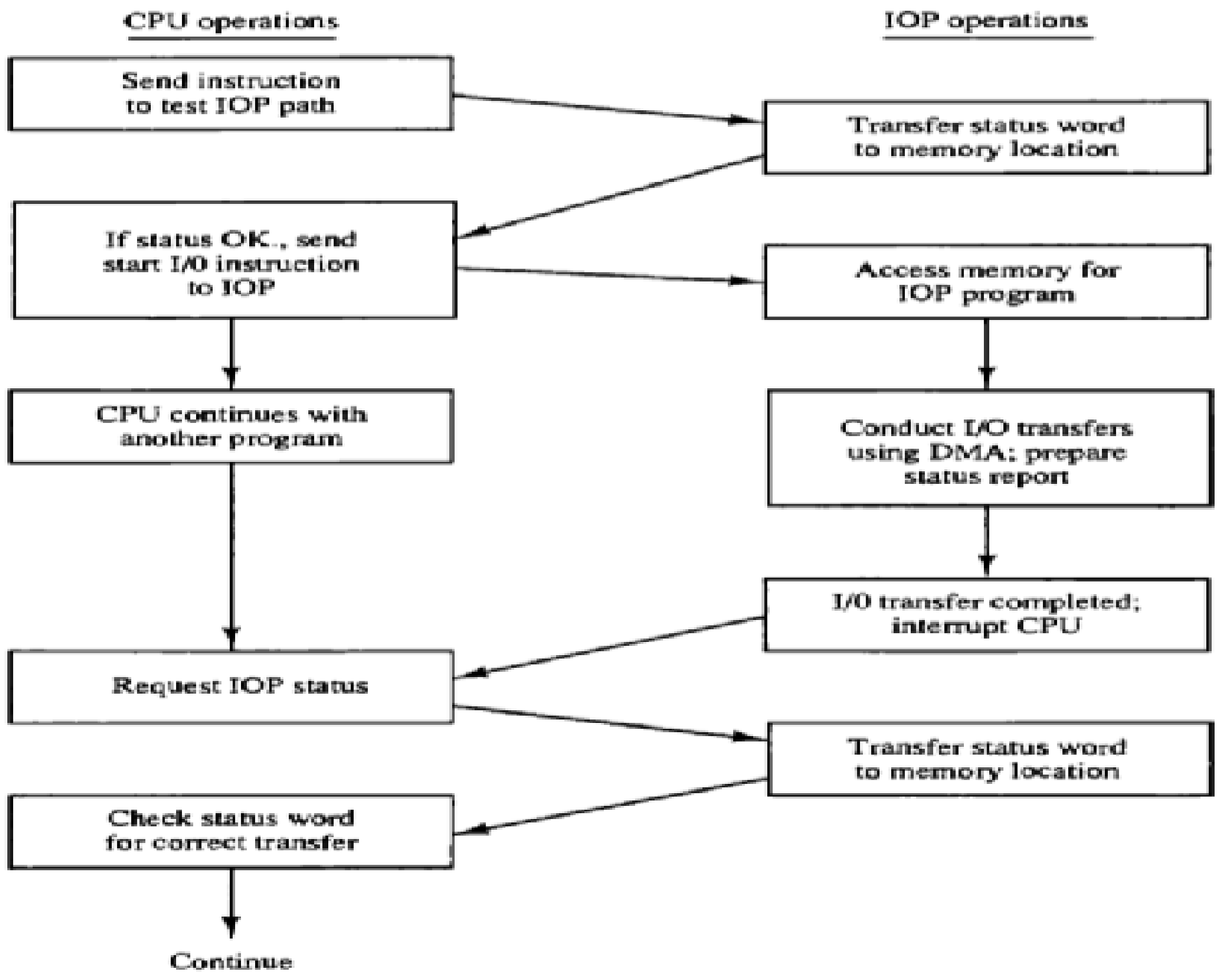


Figure 11-19 Block diagram of a computer with I/O processor.

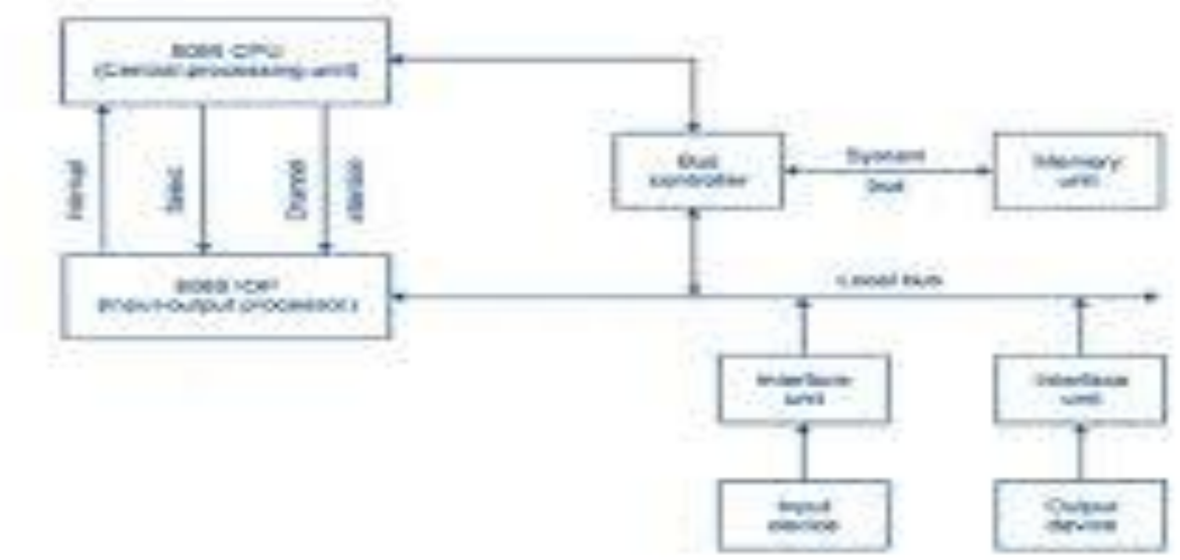
IOP

Figure 11-20 CPU-IOP communication.



CPU operations
IOP operations
Send instruction to test IOP path
Transfer status word to memory location
If status OK., 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

INTEL 8089 IOP



Serial Communication

Serial communication is a **communication** technique used in telecommunications wherein data transfer occurs by transmitting data one bit at a time in a sequential order over a **computer** bus or a **communication** channel.

It is two types-

1. Synchronous communication 2. Asynchronous communication

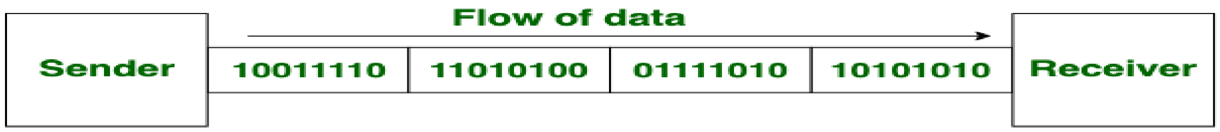
Synchronous communication

In synchronous transmission, data moves in a completely paired approach, in the form of chunks or frames.

Examples of Synchronous Transmission

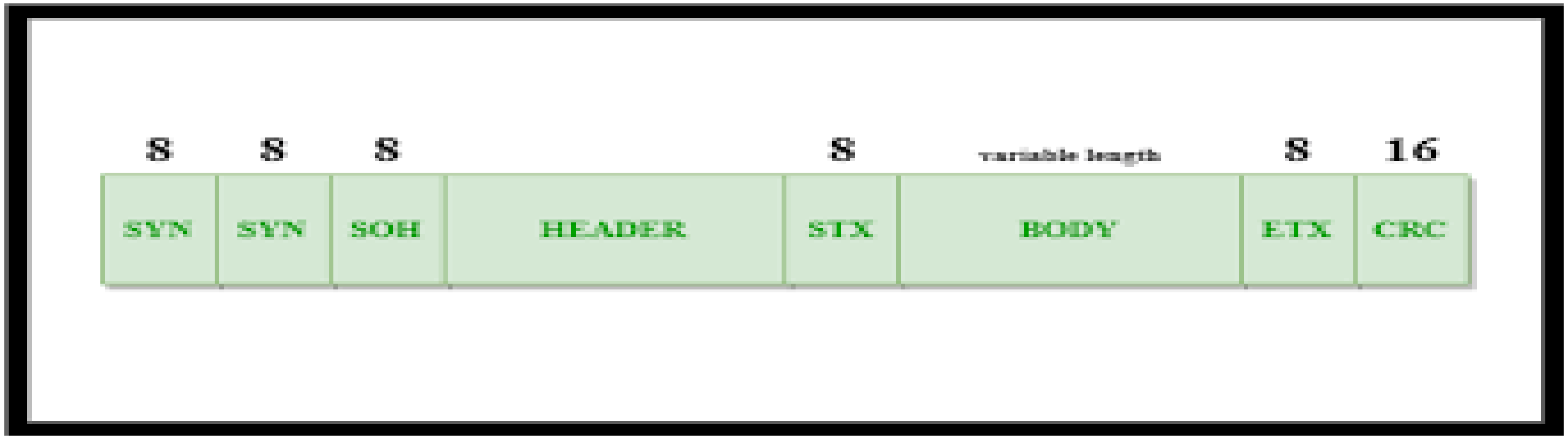
Chat rooms, Video conferencing, Telephonic conversations

Face-to-face interactions

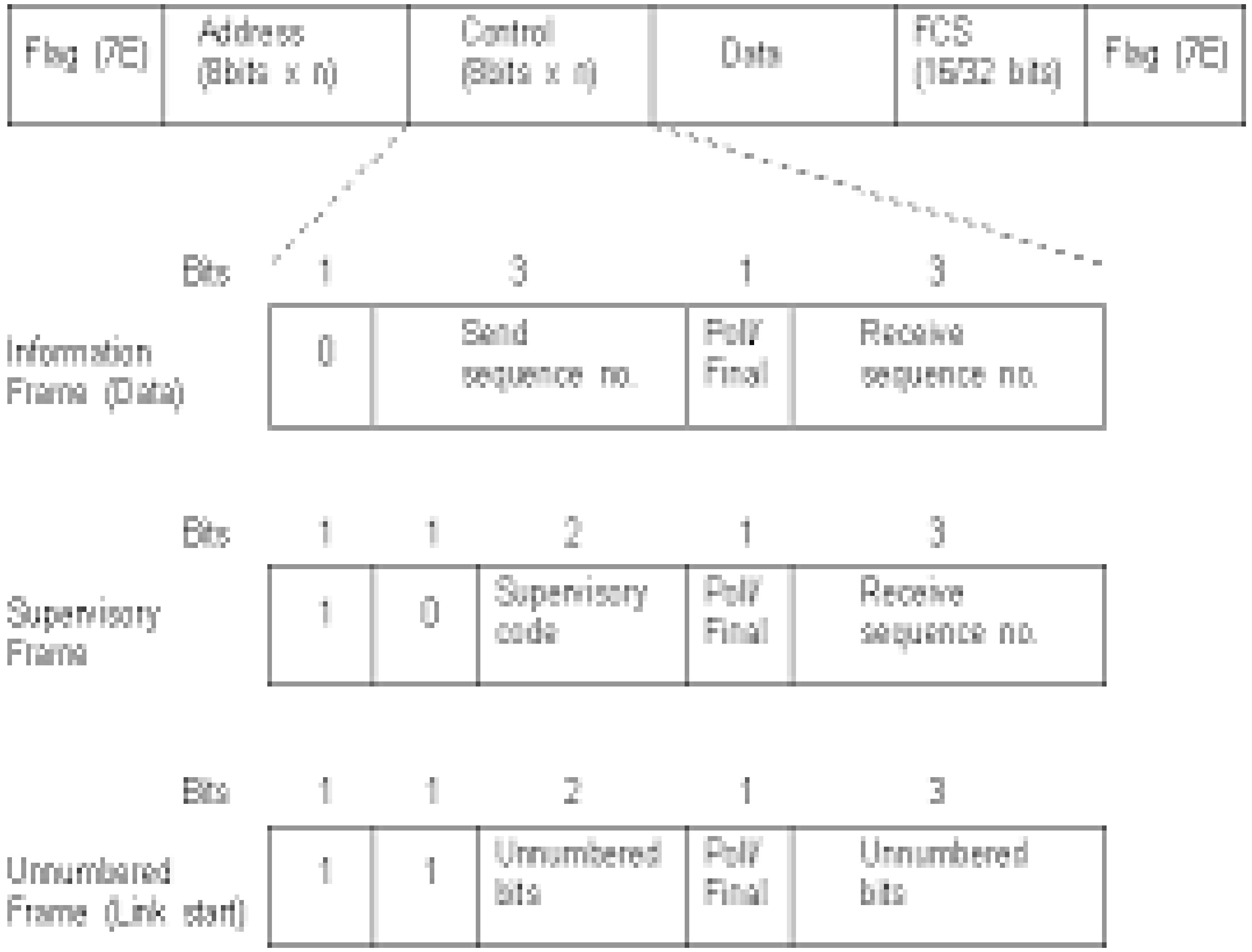


Synchronous Transmission

Character oriented protocol



Bit oriented protocol



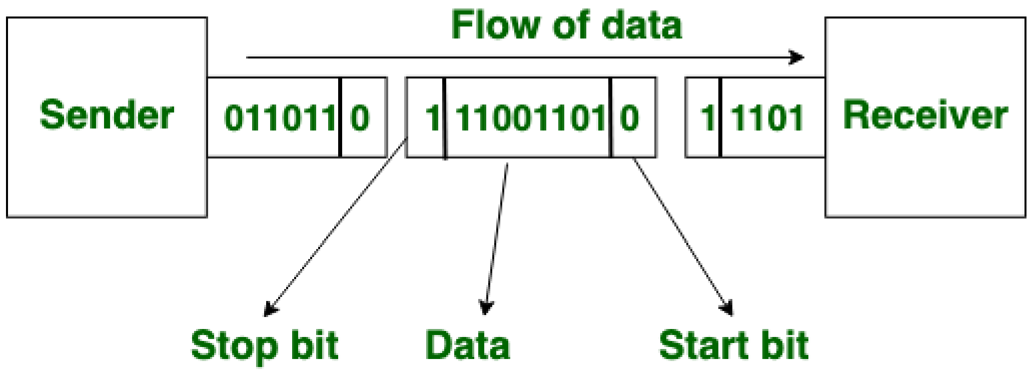
Asynchronous Serial Communication

Asynchronous Transmission

In asynchronous transmission, data moves in a half-paired approach, 1 byte or 1 character at a time. It sends the data in a constant current of bytes.

Examples of Asynchronous Transmission

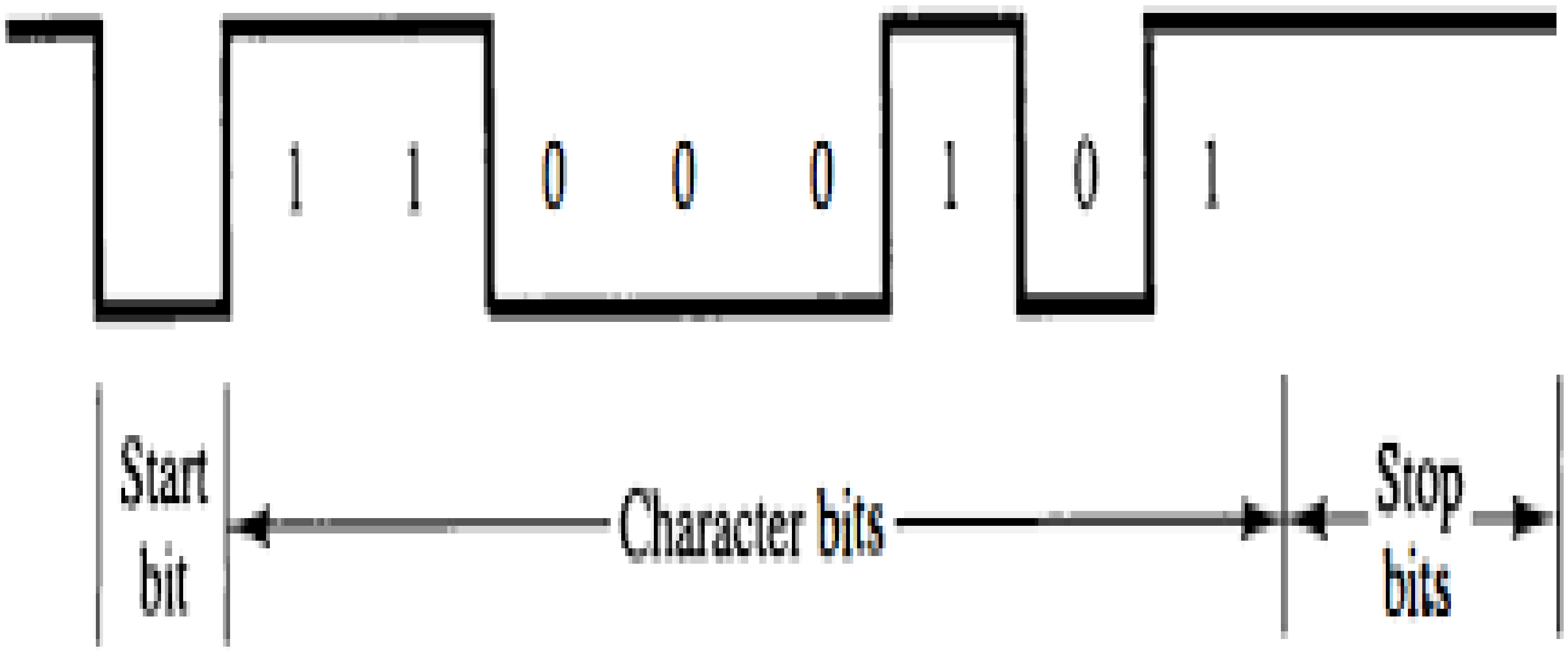
Emails, Forums, Letters, Radios, Televisions



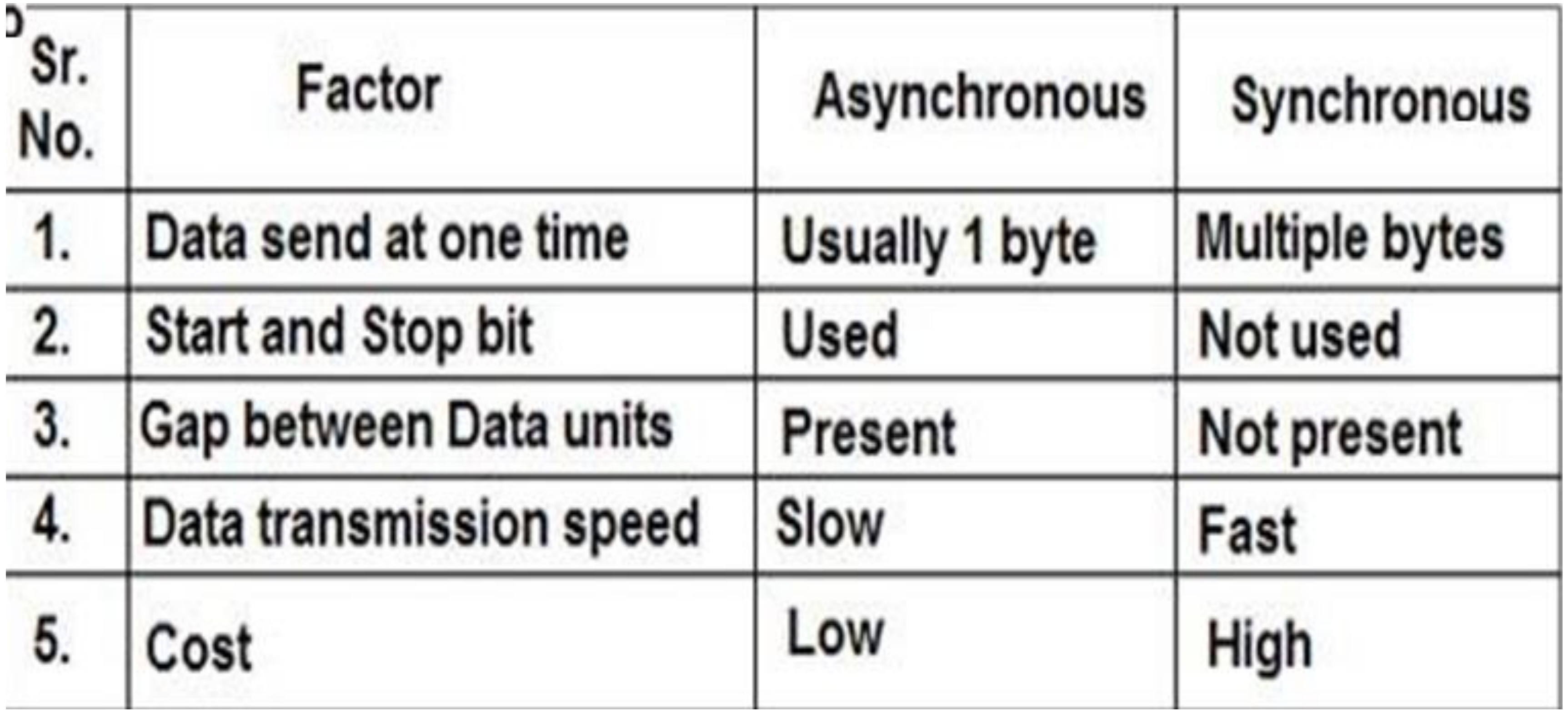
Asynchronous Transmission

Asynchronous Serial Communication

Figure 7 Asynchronous serial transmission.

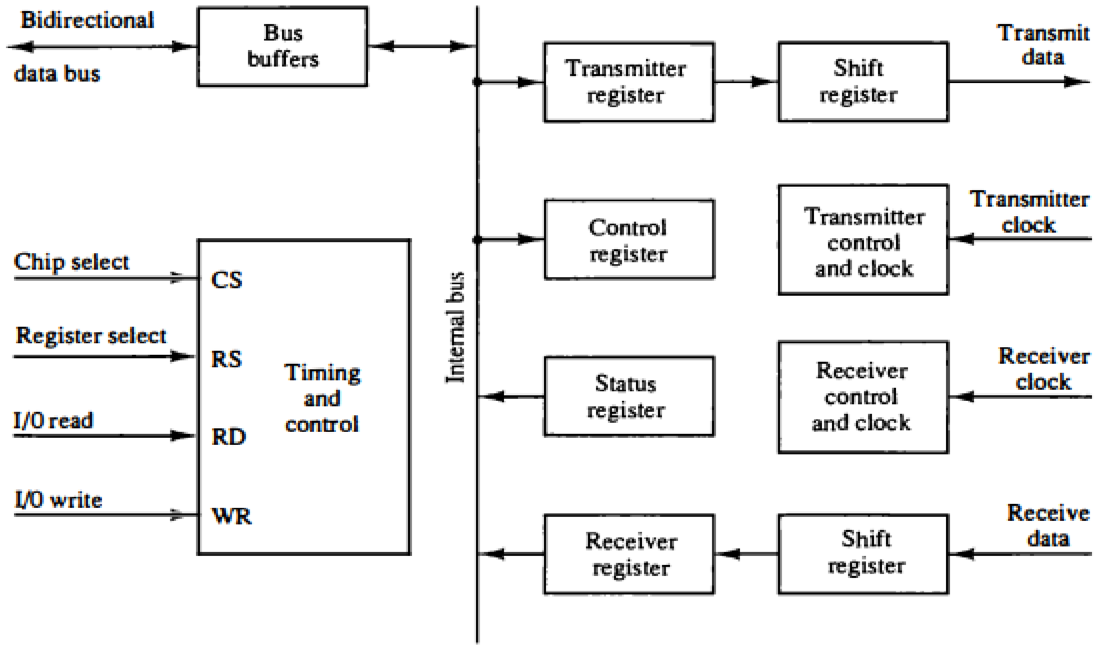


Synchronous Vs. Asynchronous



Sr. No.	Factor	Asynchronous	Synchronous
1.	Data send at one time	Usually 1 byte	Multiple bytes
2.	Start and Stop bit	Used	Not used
3.	Gap between Data units	Present	Not present
4.	Data transmission speed	Slow	Fast
5.	Cost	Low	High

Asynchronous Communication Interface(UART)



CS	RS	Operation	Register selected
0	×	×	None: data bus in high-impedance
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

Figure 11-8 Block diagram of a typical asynchronous communication interface.

Faculty Video Links, You tube Courses Details

You tube/other Video Links

- https://www.youtube.com/watch?v=Vr8wJdC8jCw
- https://www.youtube.com/watch?v=qgwmHnNsd6M
- https://www.youtube.com/watch?v=UnLww6bH6C4

Daily Quiz

- Write down three example of Input & output devices.
- Define I/O Ports.
- Write down full form of DMA.
- What do mean by Exception & TRAP ?
- Write down various types of interrupts

Weekly Assignment

- Discuss the various type of Interrupts and Exception in computer organization with suitable examples.

- Write short notes on-

  a) I/O Interface b) I/O channels & Processor

- Explain Modes of data transfer : Programmed I/O, Interrupt initiated I/O.

- Explain the working of DMA controller with help of suitable diagrams.

- Discuss the Synchronous and Asynchronous communication with suitable examples.

MCQ

1 To avoid loading during read operation, the device used is
a) latch b) flipflop c) buffer d) tristate buffer

2. The device that enables the microprocessor to read data from the external devices is, a) printer b) joystick c) display d) reader

3. The controller is connected to the ____
a) Processor BUS b) System BUS c) External BUS

4. The DMA transfers are performed by a control circuit called as

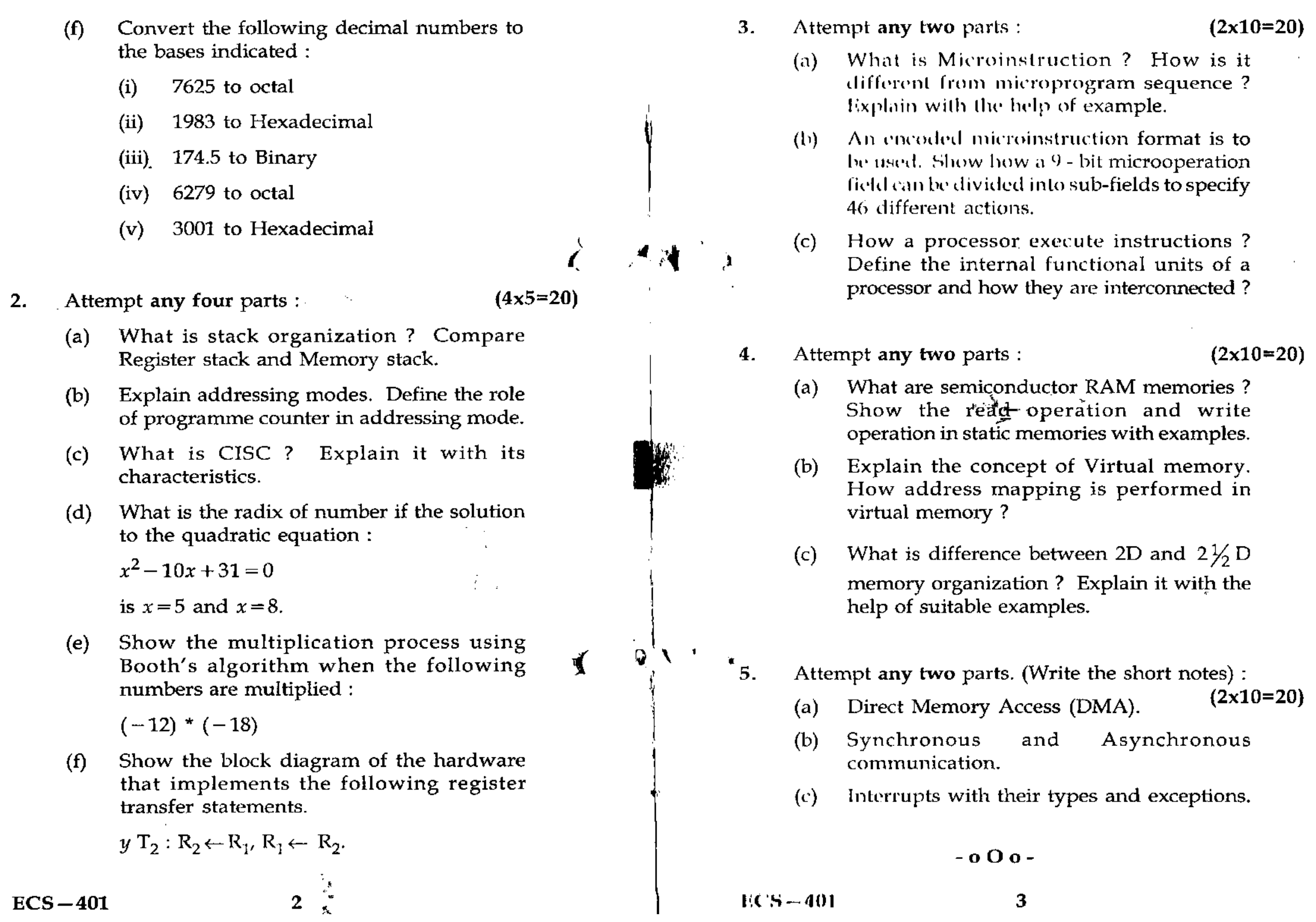
5. The DMA controller has _______ registers
a) 4 b) 2 c) 3 d) 1

6. The INTR interrupt may be
a) Maskable b) nonmaskable

7. The INTR interrupt may be masked using the flag
a) direction flag b) overflow flag c) interrupt flag d) sign flag

Solution 1 d. 2b. 3 b. 4 DMA controller . 5 c 6 a 7 c

Old Question Papers



(f) Convert the following decimal numbers to the bases indicated :

(i) 7625 to octal

(ii) 1983 to Hexadecimal

(iii) 174.5 to Binary

(iv) 6279 to octal

(v) 3001 to Hexadecimal

2. Attempt any four parts : (4x5=20)

(a) What is stack organization ? Compare Register stack and Memory stack.

(b) Explain addressing modes. Define the role of programme counter in addressing mode.

(c) What is CISC ? Explain it with its characteristics.

(d) What is the radix of number if the solution to the quadratic equation :

$x^2 - 10x + 31 = 0$

is $x = 5$ and $x = 8$.

(e) Show the multiplication process using Booth's algorithm when the following numbers are multiplied :

$(-12) * (-18)$

(f) Show the block diagram of the hardware that implements the following register transfer statements.

$y\ T_2 : R_2 \leftarrow R_1,\ R_1 \leftarrow R_2.$

ECS – 401 2

3. Attempt any two parts : (2x10=20)

(a) What is Microinstruction ? How is it different from microprogram sequence ? Explain with the help of example.

(b) An encoded microinstruction format is to be used. Show how a 9 - bit microoperation field can be divided into sub-fields to specify 46 different actions.

(c) How a processor execute instructions ? Define the internal functional units of a processor and how they are interconnected ?

4. Attempt any two parts : (2x10=20)

(a) What are semiconductor RAM memories ? Show the read operation and write operation in static memories with examples.

(b) Explain the concept of Virtual memory. How address mapping is performed in virtual memory ?

(c) What is difference between 2D and $2\frac{1}{2}$ D memory organization ? Explain it with the help of suitable examples.

5. Attempt any two parts. (Write the short notes) : (2x10=20)

(a) Direct Memory Access (DMA).

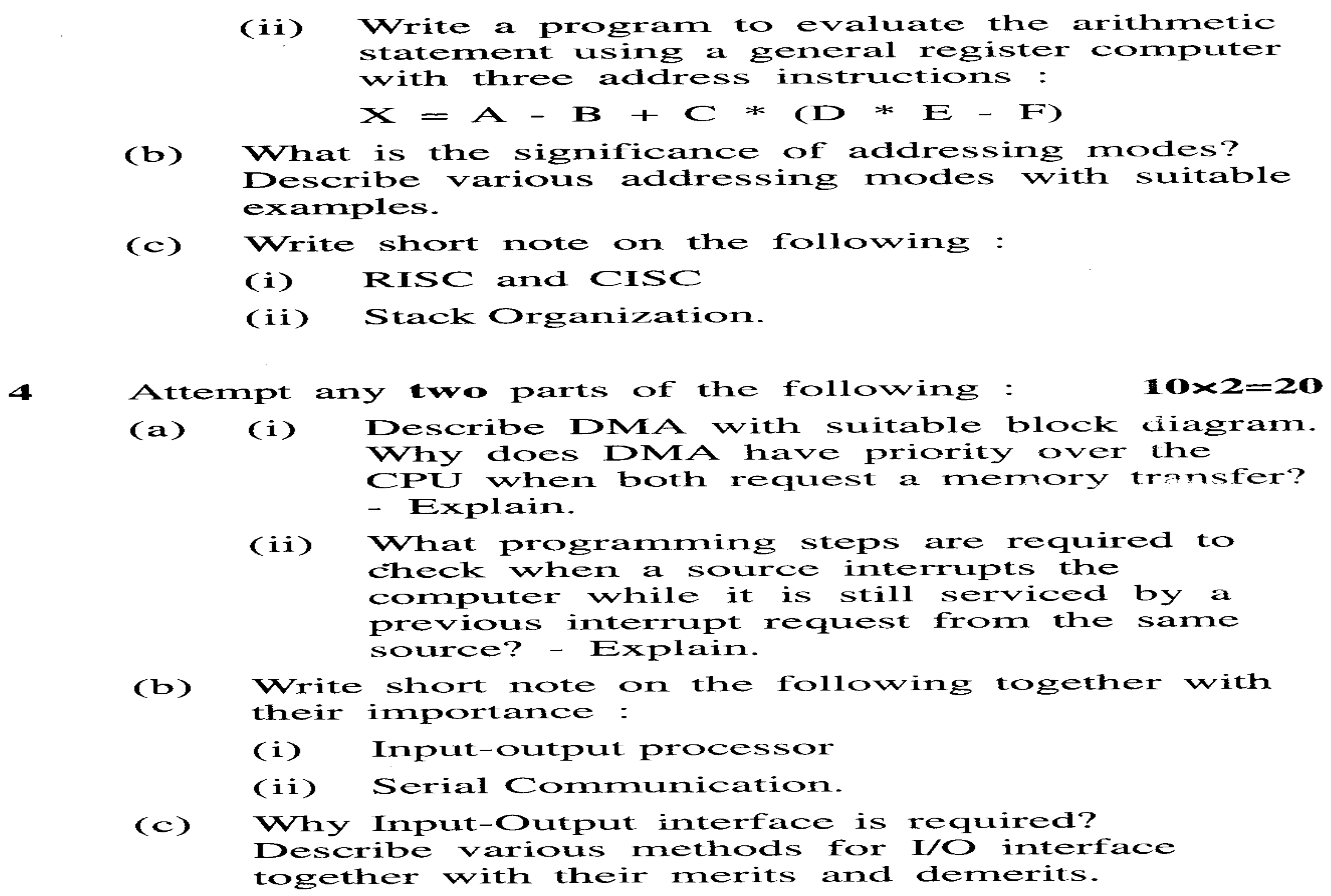
(b) Synchronous and Asynchronous communication.

(c) Interrupts with their types and exceptions.

- o O o -

ECS – 401 3

Old Question Papers



(ii) Write a program to evaluate the arithmetic statement using a general register computer with three address instructions :

X = A - B + C * (D * E - F)

(b) What is the significance of addressing modes? Describe various addressing modes with suitable examples.

(c) Write short note on the following :

(i) RISC and CISC

(ii) Stack Organization.

4 Attempt any **two** parts of the following : **10×2=20**

(a) (i) Describe DMA with suitable block diagram. Why does DMA have priority over the CPU when both request a memory transfer? - Explain.

(ii) What programming steps are required to check when a source interrupts the computer while it is still serviced by a previous interrupt request from the same source? - Explain.

(b) Write short note on the following together with their importance :

(i) Input-output processor

(ii) Serial Communication.

(c) Why Input-Output interface is required? Describe various methods for I/O interface together with their merits and demerits.

Expected Questions for University Exam

- ❑Explain the working of DMA controller with help of suitable diagrams.
- ❑Write short notes on-
  I/O Ports b) I/O Interface c) I/O channels & Processor
- ❑Discuss the Synchronous and Asynchronous communication with suitable examples.
- ❑Discuss the various type of Interrupts and Exception in computer organization with suitable examples.
- ❑Explain Modes of data transfer : Programmed I/O, Interrupt initiated I/O.

Summary

In previous slides we discuss in details

Input/output Unit:

- Peripheral devices
- I/O interface & I/O ports
- Interrupts: interrupt hardware, types of interrupts and exceptions.
- Modes of Data Transfer:
  - Programmed I/O, interrupt initiated I/O
  - Direct Memory Access.,
- I/O channels and processors.
- Serial Communication: Synchronous & asynchronous communication , standard communication interfaces