Input / Output Unit

Unit: 5

Computer Organization & Architecture

B Tech (DS)

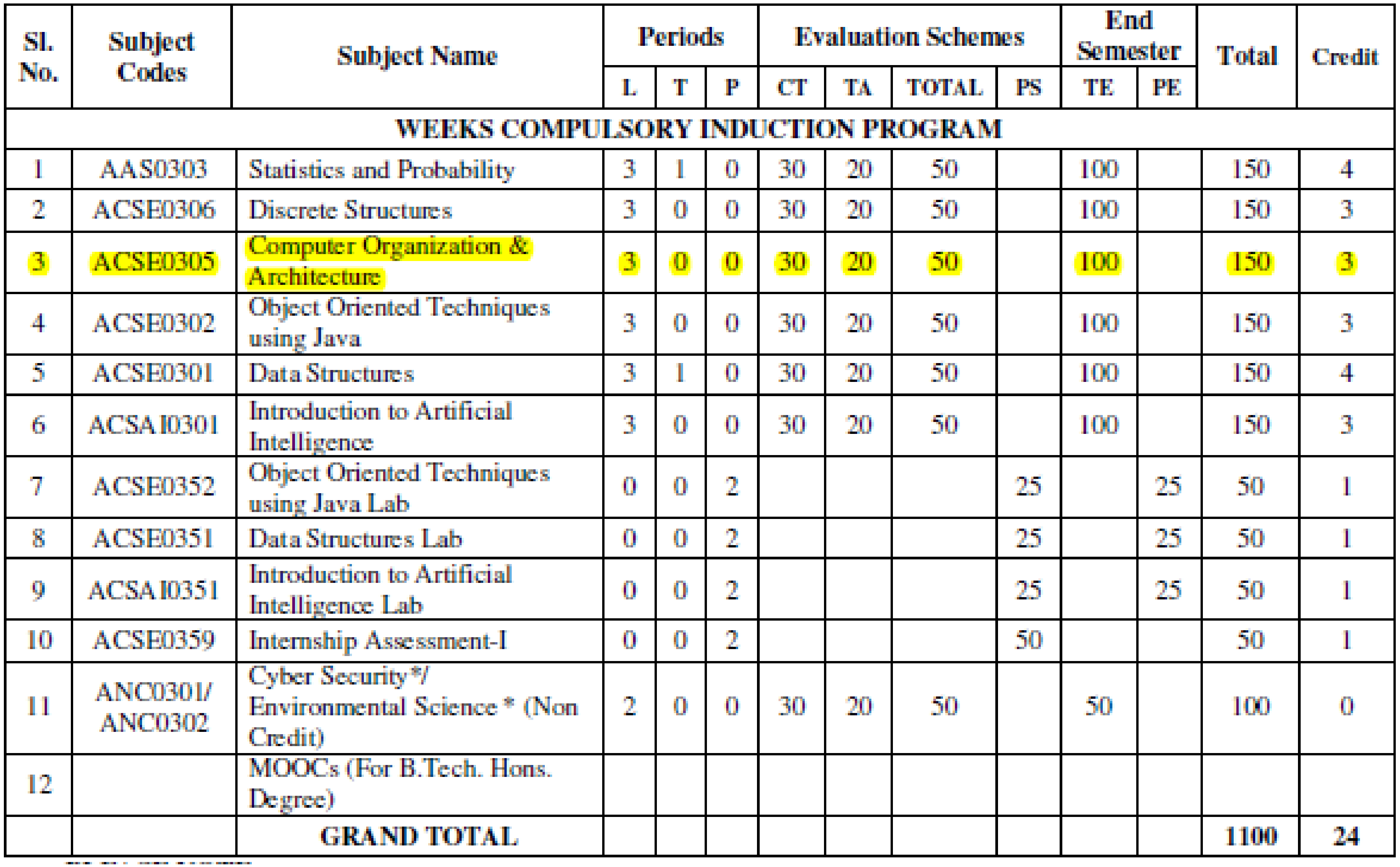
3rd Sem

Ms. Khushboo

Asst. Professor

ECE Department

Evaluation scheme



Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
WEEKS COMPULSORY INDUCTION PROGRAM													
1	AAS0303	Statistics and Probability	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0305	Computer Organization & Architecture	3	0	0	30	20	50		100		150	3
4	ACSE0302	Object Oriented Techniques using Java	3	0	0	30	20	50		100		150	3
5	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
6	ACSAI0301	Introduction to Artificial Intelligence	3	0	0	30	20	50		100		150	3
7	ACSE0352	Object Oriented Techniques using Java Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACSAI0351	Introduction to Artificial Intelligence Lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301/ ANC0302	Cyber Security*/ Environmental Science * (Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

Subject Syllabus

Course Contents / Syllabus		
UNIT-I	**Introduction**	8 Hours
Computer Organization and Architecture, Functional units of digital system and their interconnections, buses, bus architecture, types of buses and bus arbitration and it's types. Register, bus and memory transfer. Process or organization, general registers organization, stack organization and addressing modes.		
UNIT-II	**ALU Unit**	8 Hours
Arithmetic and logic unit: Lookahead carries adders. Multiplication: Signed operand multiplication, Booth's algorithm and array multiplier. Division and logic operations. Floating point arithmetic operation, Arithmetic & logic unit design. IEEE Standard for Floating Point Numbers.		
UNIT-III	**Control Unit**	8 hours
Control Unit: Instruction types, formats, instruction cycles and sub cycles (fetch and execute etc.), microoperations, execution of a complete instruction. Program Control, Reduced Instruction Set Computer, Complex Instruction Set Computer, Pipelining. Hardwire and microprogrammed control, Concept of horizontal and vertical microprogramming, Flynn's classification.		

Subject Syllabus

Course Contents / Syllabus		
UNIT-IV	**Memory Unit**	8 hours
Memory: Basic concept and hierarchy, semiconductor RAM memories, 2D & 2 1/2D memory organization. ROM memories. Cache memories: concept and design issues & performance, address mapping and replacement Auxiliary memories: magnetic disk, magnetic tape and optical disks Virtual memory: concept implementation, Memory Latency, Memory Bandwidth, Memory Seek Time.		
UNIT-V	**Input/Output**	8 hours
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 and Direct Memory Access. ,I/O channels and processors. Serial Communication: Synchronous & asynchronous communication.		

Branch wise Applications

Computer Science:

Understanding of Computer Organization and Architecture is required for:

- Performance analysis of practical software
- Parallel Software and its execution
- High performance databases
- Modern Compilers and Code optimization
- High performance game programming

Other applications

- **Bio-informatics, Data science using python, Web programming**

For high performance computing, we require COA background.

Course Objective

- Discuss the basic concepts and structure of computers.
- Understand concepts of register transfer logic and arithmetic operations.
- Explain different types of addressing modes and memory organization.
- Understand the concepts of memory system and Learn the different types of memories to store data.
- Explain the various types of interrupts and modes of data transfer.

Course Outcomes

Course outcomes : After completion of this course students will be able to		
CO 1	Understand the basic structure and operation of a digital computer system	K1, K2
CO 2	Analyze the design of arithmetic & logic unit and understand the fixed point and floating-point arithmetic operations.	K1, K4
CO 3	Implement control unit techniques and the concept of Pipelining	K3
CO 4	Understand the hierarchical memory system, cache memories and virtual memory.	K2
CO 5	Understand different ways of communicating with I/O devices and standard I/O interfaces.	K2

Program Outcomes

- **Program Outcomes** are narrow statements that describe what the students are expected to know and would be able to do upon the graduation.
- These relate to the skills, knowledge, and behavior that students acquire through the programmed.

1. Engineering knowledge
2. Problem analysis
3. Design/development of solutions
4. Conduct investigations of complex problems
5. Modern tool usage
6. The engineer and society
7. Environment and sustainability
8. Ethics
9. Individual and team work
10. Communication
11. Project management and finance
12. Life-long learning

CO-PO Mapping

		COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)										
CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12
CO1	3	2	1	1	1	1	1	-	1	1	1	2
CO2	2	2	2	2	1	1	-	1	1	1	1	2
CO3	3	2	2	1	2	2	1	1	2	2	1	2
CO4	3	2	2	2	2	1	1	-	1	1	1	2
CO5	2	2	2	1	2	-	1	-	1	2	2	2
Average	2.6	2	1.8	1.4	1.6	1	0.8	0.4	1.2	1.4	1.2	2

End Semester Question Paper Template

Question Paper
emplate -100 Mark

Prerequisite and Recap

- **Basic knowledge of Digital Logic Design**
- **ALU Unit**
- **Control Unit**
- **Memory unit**

Brief Introduction about Subject

The Computer Organization and architecture is one of the most important and comprehensive subject that includes many foundational concepts and knowledge used in design of a computer system. This subject provides in-depth knowledge of internal working, structuring, and implementation of a computer system.

The COA important topics include all the fundamental concepts such as computer system functional units , processor micro architecture , program instructions, instruction formats , addressing modes , instruction pipelining, memory organization , instruction cycle, interrupts and other important related topics.

Video link: https://www.youtube.com/watch?v=q6oiRtKTpX4

Unit Content

- 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 communication
  - ➢Asynchronous communication

Unit Objective

Explain the various types of interrupts and modes of data transfer.

Introduction to Topic 1

Name of Topic	Objective of Topic	Mapping with CO
Peripheral Devices, I/O ports and I/O interface	Students will be able to understand about different peripheral devices and how they are connected to processor and Memory system through IO port and Interface.	CO 5

Peripheral devices

- **Input or output devices attached to the computer are also called *peripherals.***
- There are three types of peripherals such as input, output, and input–output peripherals.
- These peripherals may be analog or digital and serial or parallel.
- Among the most common peripherals are keyboards, display units, and printers.
- Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.

Peripheral devices

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

Monitor and keyboard-

•Video monitors are the most commonly used peripherals. They consist
•of a keyboard as the input device and a display unit as the output device.
•There are different types of video monitors, but the most popular use a cathode ray tube (CRT). The CRT contains an electronic gun that sends an electronic beam to a phosphorescent screen in front of the tube.
•The beam can be deflected horizontally and vertically. To produce a pattern on the screen, a grid inside the CRT receives a variable voltage that causes the beam to hit the screen and make it glow at selected spots.
•Horizontal and vertical signals deflect the beam and make it sweep across the tube, causing the visual pattern to appear on the screen.

Peripheral devices

- A characteristic feature of display devices is a cursor that marks the position in the screen where the next character will be inserted.
- The cursor can be moved to any position in the screen, to a single character, the beginning of a word, or to any line.
- Edit keys add or delete information based on the cursor position.
- The display terminal can operate in a single character mode where all characters entered on the screen through the keyboard are transmitted to the computer simultaneously.
- In the block mode, the edited text is first stored in a local memory inside the terminal. The text is transferred to the computer as a block of data.

Peripheral devices

Printer -

•Printers provide a permanent record on paper of computer output data or text.

•There are three basic types of character printers: daisywheel, dot matrix, and laser printers.

•The daisywheel printer contains a wheel with the characters placed along the circumference.

•To print a character, the wheel rotates to the proper position and an energized magnet then presses the letter against the ribbon.

•The dot matrix printer contains a set of dots along the printing mechanism.

•The laser printer uses a rotating photographic drum that is used to imprint the character images. The pattern is then transferred onto paper in the same manner as a copying machine

Input-Output Interface

- **Input–output interface provides a method for transferring information between internal storage and external I/O devices.**

- The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.
- The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

2. **The data transfer rate of peripherals is usually slower than the transfer rate of the CPU**, and consequently, a synchronization mechanism may be needed.

Input-Output Interface

3. **Data codes and formats in peripherals differ** from the word format in the CPU and memory.

4. **The operating modes of peripherals are different** from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

Input-Output Interface

Interface -

•To resolve the differences that exist between the central computer and each peripheral computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers.

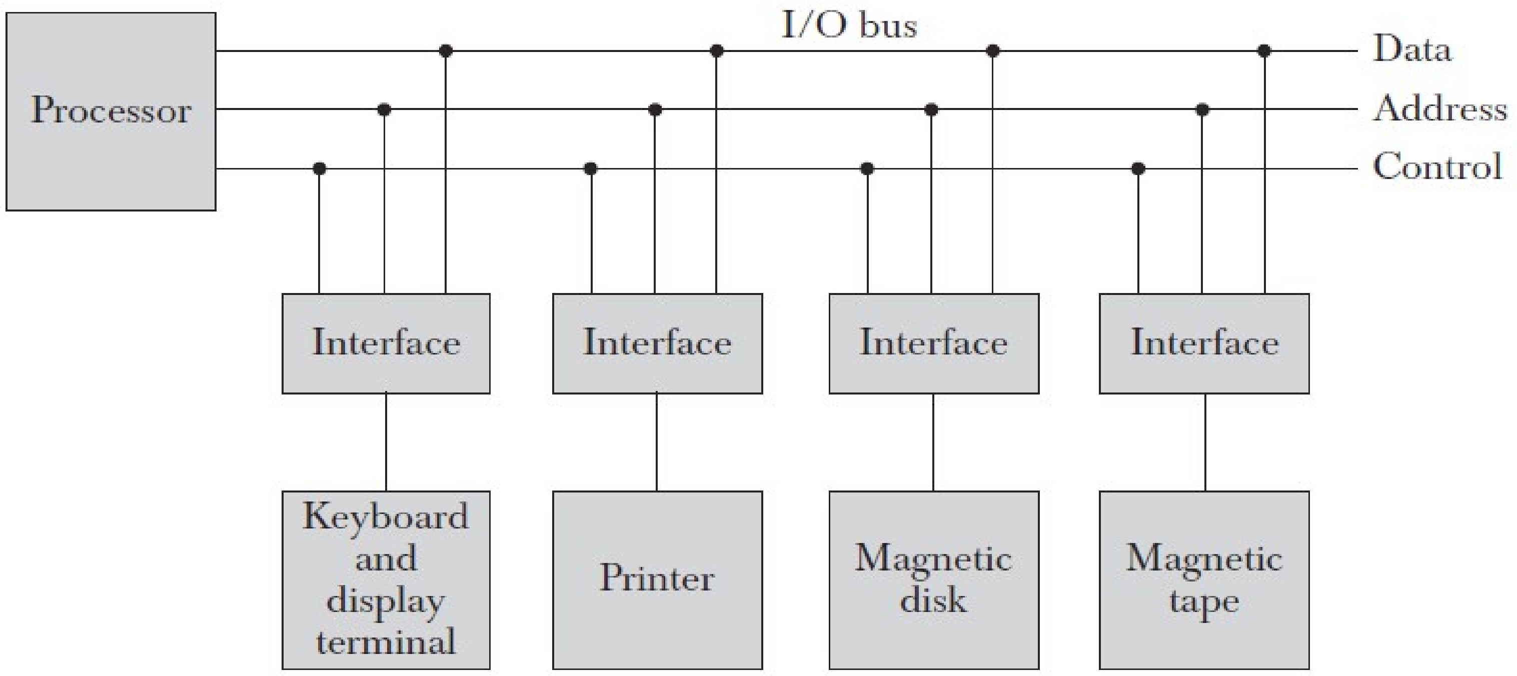
•**These components are called *interface* units because they interface between the processor bus and the peripheral device.**

•The word “Interface” is a general term for the point of contact between two parts of a system.

Input-Output Interface - I/O Bus and Interface Modules

- A typical communication link between the processor and several peripherals is shown in Fig. 11-1.

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



Input-Output Interface

- The I/O bus from the processor is attached to all peripheral interfaces.
- To communicate with a particular device, the processor places a device address on the address lines.
- Each interface attached to the I/O bus contains an address decoder that monitors the address lines.
- When the interface detects its own address, it activates the path between the bus lines and the device that it controls.
- All peripherals whose address does not correspond to the address in the bus are disabled by their interface.

Input-Output Interface

I/O command

•It is a function code and is **an instruction that is executed in the interface and its attached peripheral unit.**

•The interpretation of the command depends on the peripheral that the processor is addressing.

•There are four types of commands that an interface may receive- Control command, Status ,Data output, and data input.

1.Control Command -

A *control command* is issued to activate the peripheral and to inform it what to do.

Each peripheral receives its own distinguished sequence of control commands, depending on its mode of operation.

Input-Output Interface

2. Status-

A *status command* is used to test various status conditions in the interface and the peripheral.

- For example, the computer may wish to check the status of the peripheral before a transfer is initiated.
- During the transfer, one or more errors may occur which are detected by the interface.
- These **errors are designated by setting bits in a status register** that the processor can read at certain intervals.

Input-Output Interface

3. Data Output

A *data output command* causes the interface to respond by transferring data from the bus into one of its registers.

- Consider an example with a tape unit. The computer starts the tape moving by issuing a control command.
- The processor then monitors the status of the tape by means of a status command.
- When the tape is in the correct position, the processor issues a data output command.
- The interface responds to the address and command and transfers the information from the data lines in the bus to its buffer register.
- The **interface then communicates with the tape controller and sends the data to be stored on tape**.

Input-Output Interface

4. Data Input

- **The *data input command* is the opposite of the data output.**
- In this case the interface receives an item of data from the peripheral and places it in its buffer register.
- The processor checks if data are available by means of a status command and then issues a data input command.
- The **interface places the data on the data lines**, where they are accepted by the processor.

Input-Output Interface - I/O versus Memory Bus

I/O versus Memory Bus

•**In addition to communicating with I/O, the processor must communicate with the memory unit.**

•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.

Input-Output Interface - I/O versus Memory Bus

1. IOP

- **In this, the computer has independent sets of data, address, and control buses, one for accessing memory and the other for I/O.**
- This is done in computers that provide a separate I/O processor (IOP) in addition to the central processing unit (CPU).
- The memory communicates with both the CPU and the IOP through a memory bus.
- The IOP communicates also with the input and output devices through a separate I/O bus with its own address, data and control lines.
- **The purpose of the IOP is to provide an independent pathway for the transfer of information** between external devices and internal memory.
- The I/O processor is **sometimes called a data channel.**

Input-Output Interface - I/O versus Memory Bus

2. Isolated I/O

- **In this computers use one common bus to transfer information between memory or I/O and the CPU.**
- The distinction between a memory transfer and I/O transfer is made through **separate read and write lines**.
- The *I/O read* and *I/O write* control lines are enabled during an I/O transfer. The *memory read* and *memory write* control lines are enabled during a memory transfer.
- The **isolated I/O method isolates memory and I/O interface addresses so that memory address values are not affected by interface address assignment since each has its own address space.**

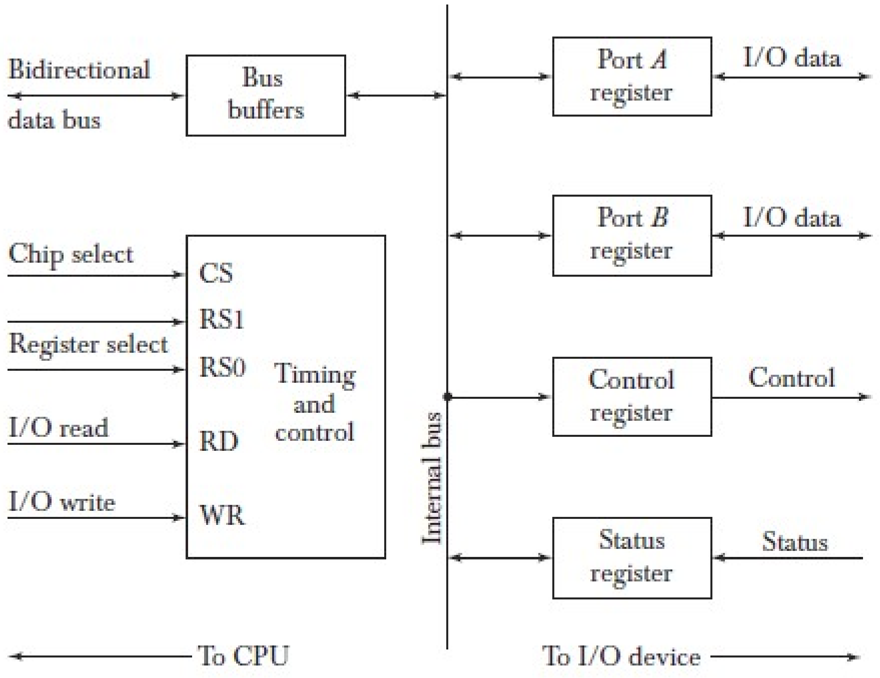
Input-Output Interface - I/O versus Memory Bus

3. Memory mapped I/O

In this computers use same address space for both memory and I/O.

- This is the case in computers that **employ only one set of read and write signals** and do not distinguish between memory and I/O addresses.
- The computer treats an interface register as being part of the memory system.
- **The assigned addresses for interface registers cannot be used for memory words, which reduces the memory address range available**.
- Computers with memory-mapped I/O can use memory-type instructions to access I/O data.
- It allows the computer to use the same instructions for either input–output transfers or for memory transfers.
- **The advantage is that the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers.**

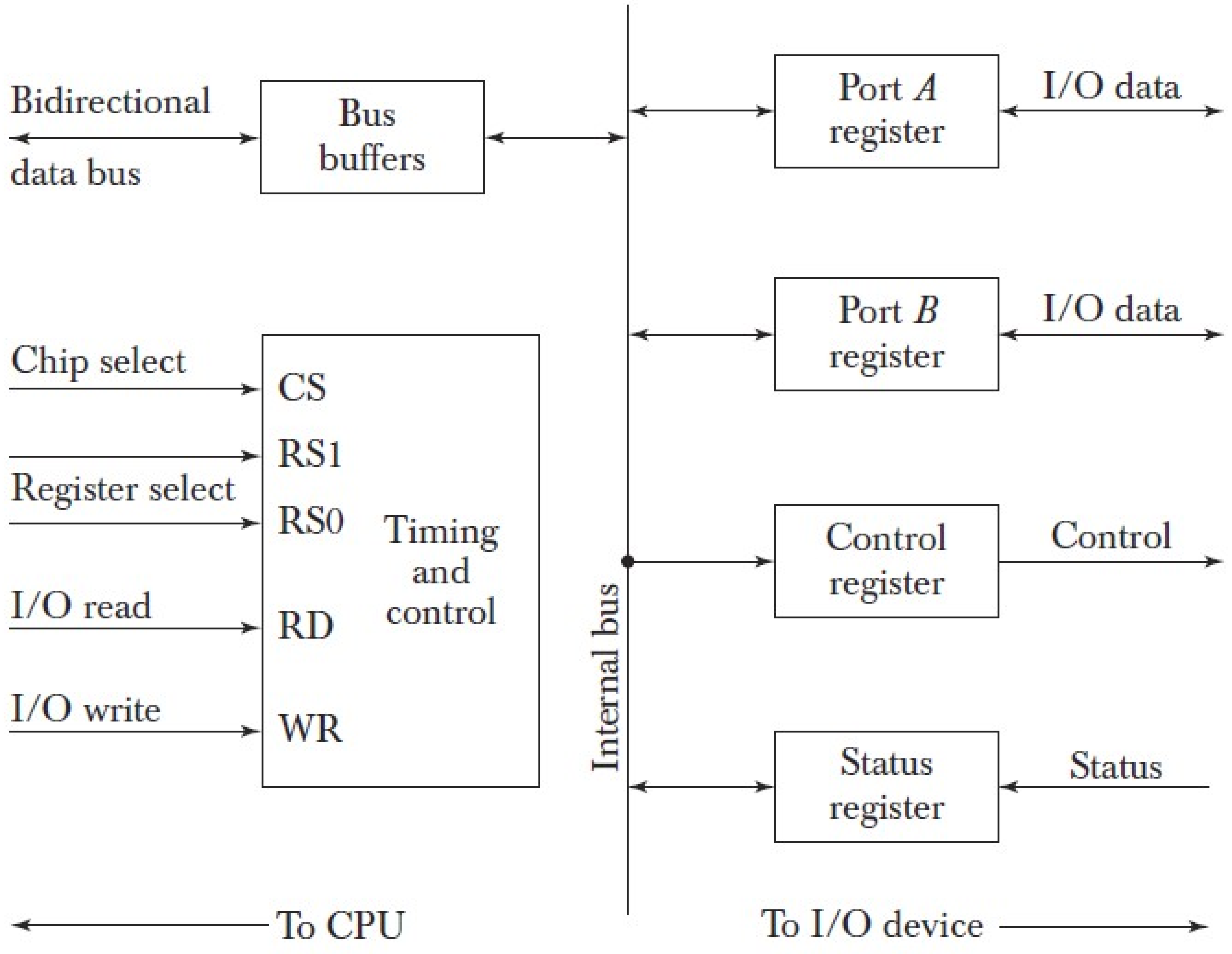
Example of I/O Interface Unit



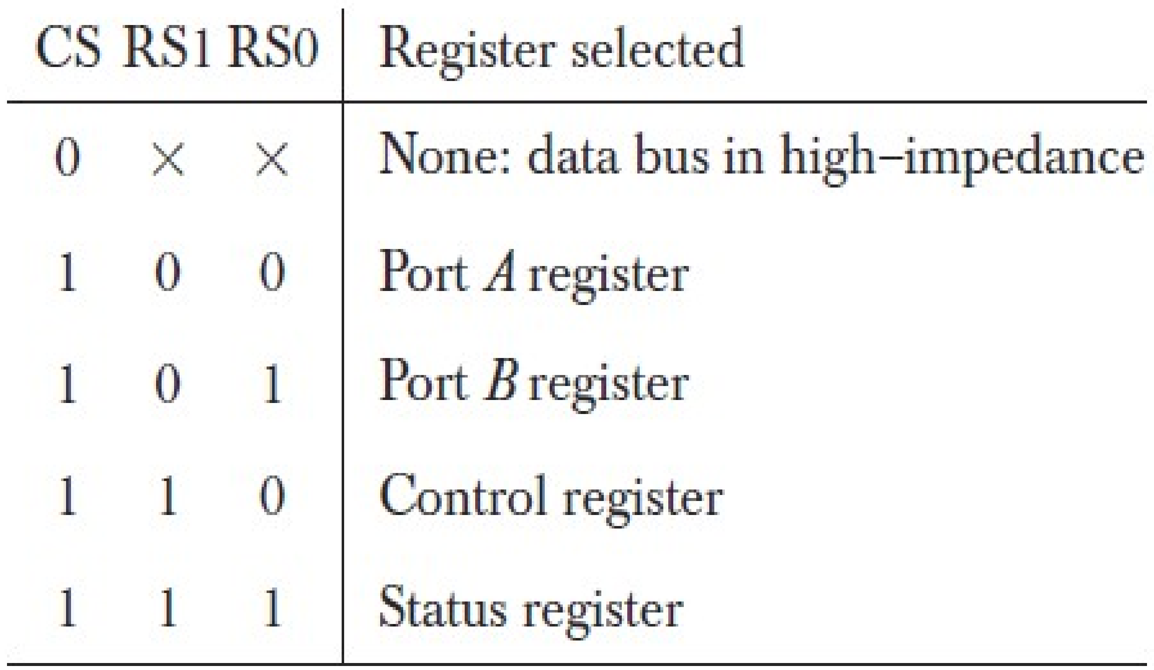
CS	RS1	RS0	Register selected
0	×	×	None: data bus in high-impedance
1	0	0	Port *A* register
1	0	1	Port *B* register
1	1	0	Control register
1	1	1	Status register

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

Example of I/O Interface Unit



Example of I/O 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.

Data Transfer

- The transfer of data between two units may be done in –

1. Parallel or
2. Serial

1. Parallel data transmission

- In this each bit of the message has its own path and the total message is transmitted at the same time.
- This means that an *n*-bit message must be transmitted through *n* separate conductor paths.
- **Parallel transmission is faster but requires many wires.**
- It is used for short distances and where speed is important.

Data Transfer

2. Serial data transmission

- In this each bit in the message is sent in sequence one at a time.
- This method requires the use of one pair of conductors or one conductor and a common ground.
- **Serial transmission is slower but is less expensive since it requires only one pair of conductors.**
- Serial transmission can be synchronous or asynchronous.

Serial Transfer

Serial transmission can be synchronous or asynchronous.

1.Synchronous Transmission -

•In synchronous transmission, the two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.

2.Asynchronous Transmission -

•In asynchronous, the internal timing in each unit is independent from the other or in this each register uses its own private clock.

•In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.

Asynchronous Data Transfer

- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.

- In asynchronous, the internal timing in each unit is independent from the other or in this each register uses its own private clock.

- Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

- There are two ways- **Strobe and Handshaking**

Asynchronous Data Transfer

1. **Strobe**

- **A *strobe* pulse is supplied by one of the units to indicate to the other unit when the transfer has to occur.**

2. **Handshaking**

- In handshaking, **each data item being transferred is accompanied 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**.

- This type of agreement between two independent units is referred to as *handshaking.*

Asynchronous Data Transfer

Timing Diagram-

•It is customary to specify the asynchronous transfer between two independent units by means of a timing diagram that **shows the timing relationship that must exist between the control signals and the data in the buses.**

•**The sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination unit.**

Strobe Control

Source –initiated strobe for data transfer

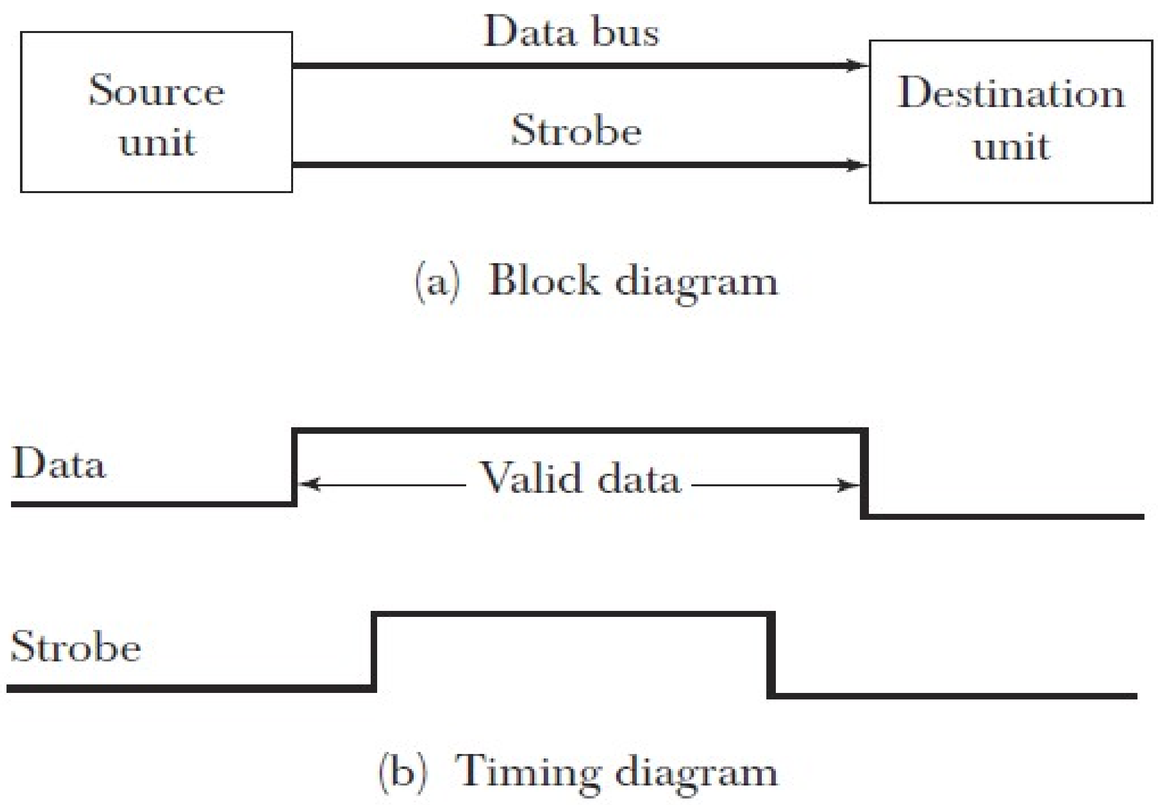
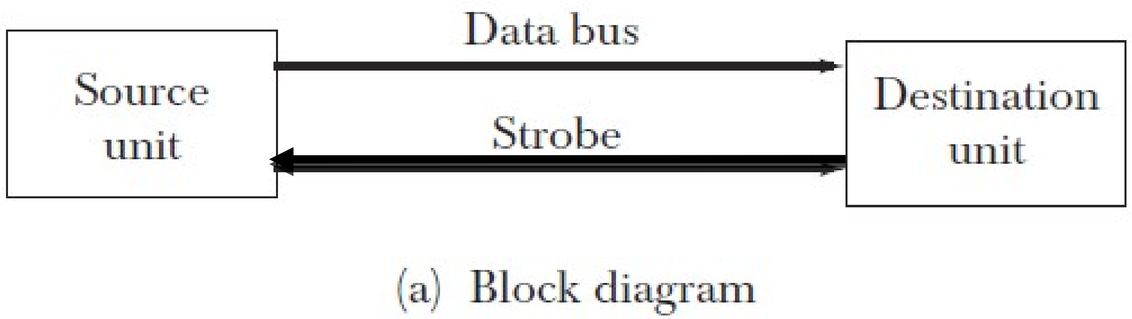


Figure 11-3 Source-initiated strobe for data transfer.

Strobe Control

Destination –initiated strobe for data transfer



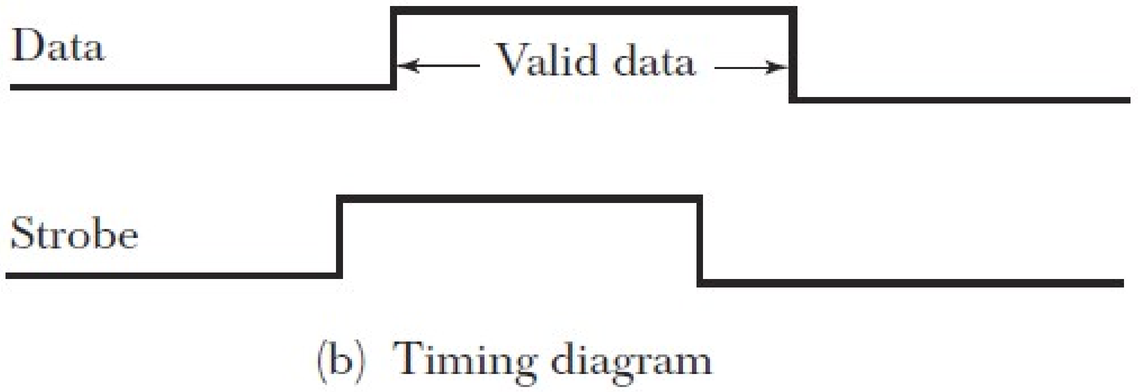


Figure 11-4 Destination-initiated strobe for data transfer.

Strobe Control

Strobe Control Disadvantage-

•The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus.

•Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

•The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.

Two – Wire Control (Handshaking)

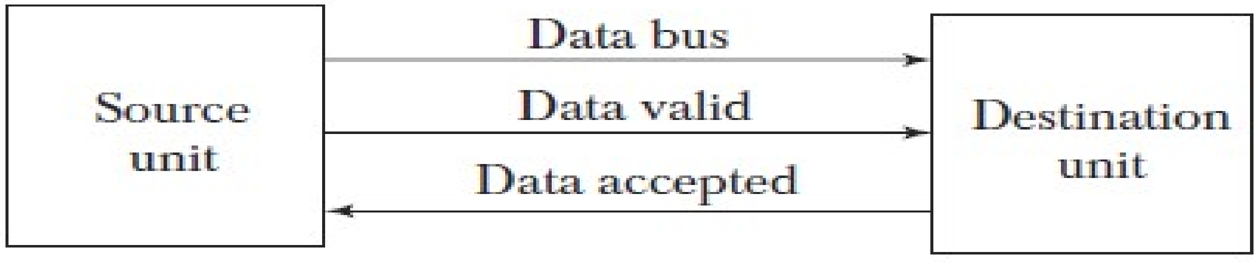
Two – Wire Control (two wire handshaking) -

The basic principle of the two-wire handshaking method of data transfer is as follows-

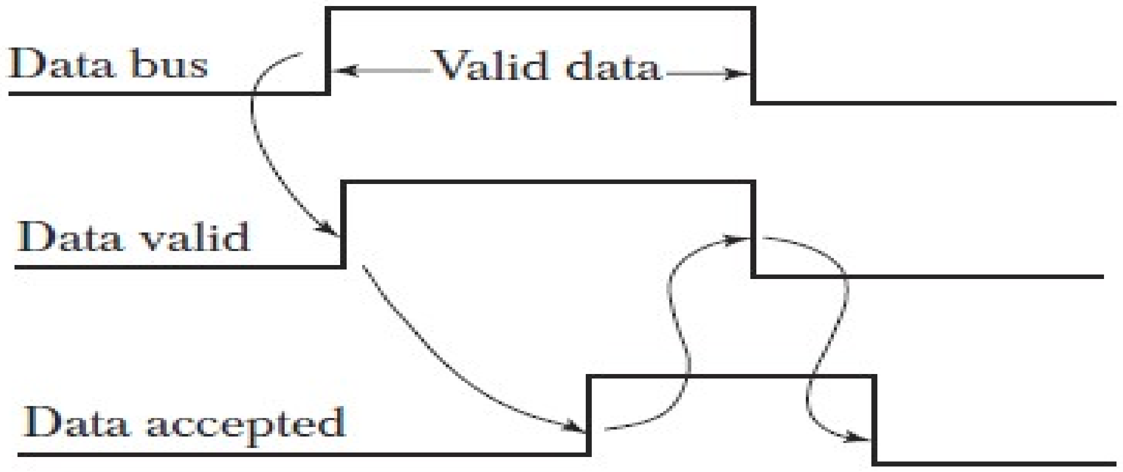
- One control line is in the same direction as the data flow in the bus from the source to the destination.
- It is used by the source unit to inform the destination unit whether there are valid data in the bus.
- The other control line is in the other direction from the destination to the source.
- It is used by the destination unit to inform the source whether it can accept data.
- The sequence of control during the transfer depends on the unit that initiates the transfer.

Two – Wire Control (Handshaking)

Source initiated transfer using handshaking



(a) Block diagram



(b) Timing diagram

Two – Wire Control (Handshaking)

Source initiated transfer using handshaking

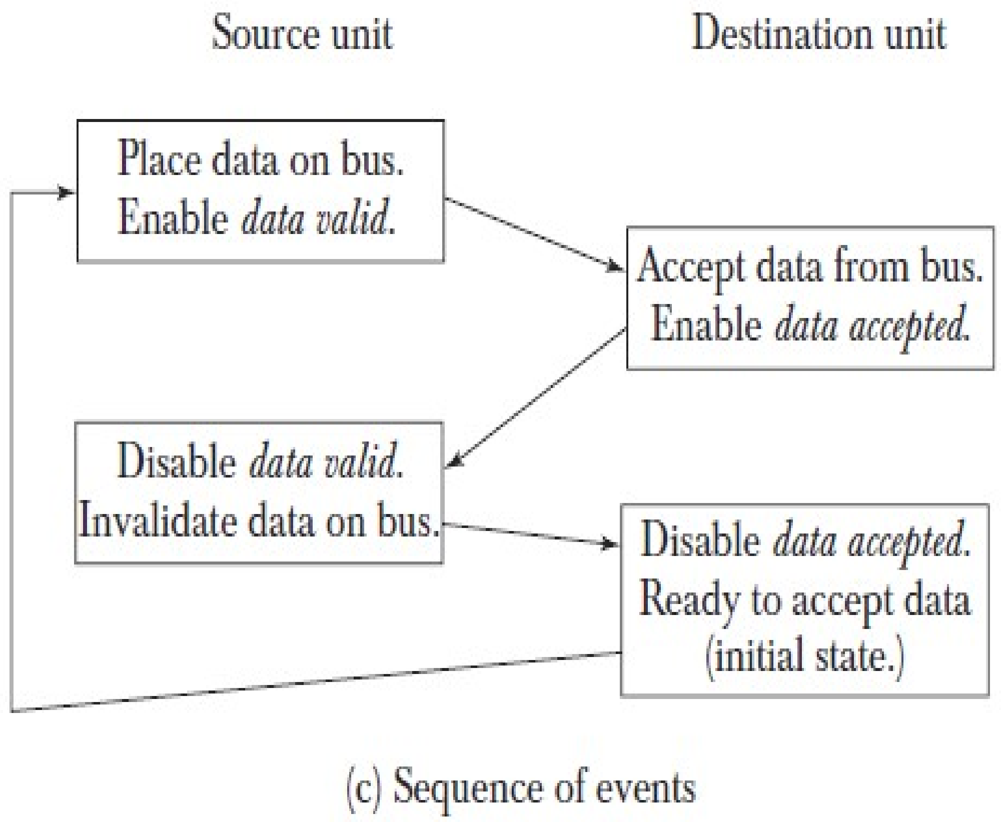
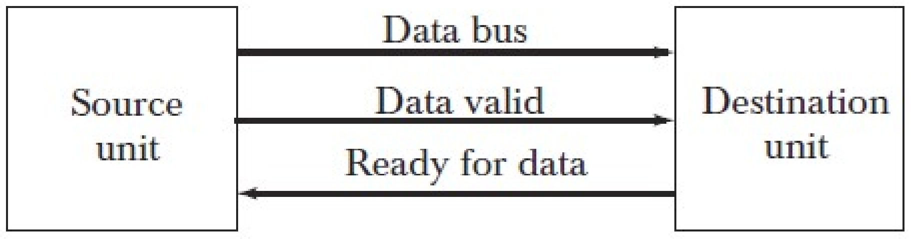

Source unit
Destination unit
Place data on bus. Enable *data valid.*
Accept data from bus. Enable *data accepted.*
Disable *data valid.* Invalidate data on bus.
Disable *data accepted.* Ready to accept data (initial state.)
(c) Sequence of events

Figure 11-5 Source-initiated transfer using handshaking.

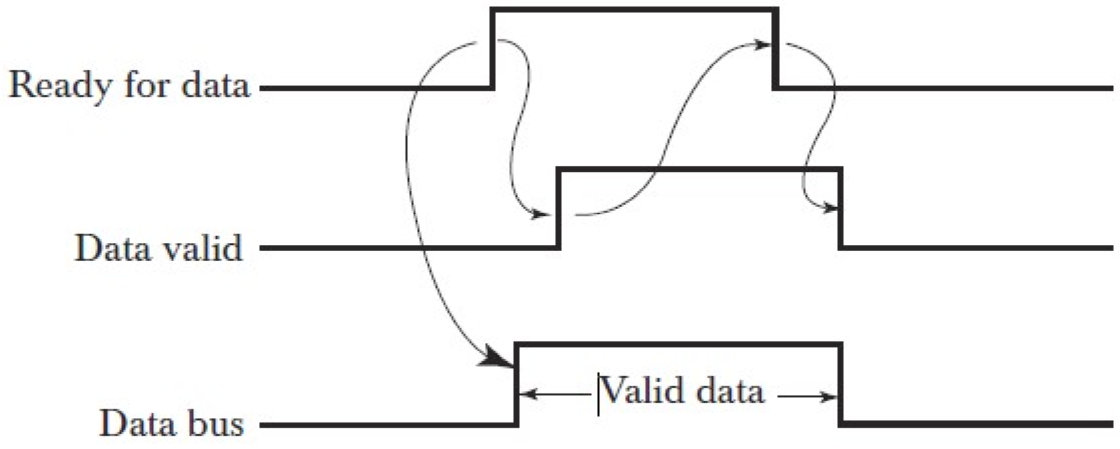
Two – Wire Control (Handshaking)

Destination initiated transfer using handshaking

Figure 11-6 Destination-initiated transfer using handshaking.



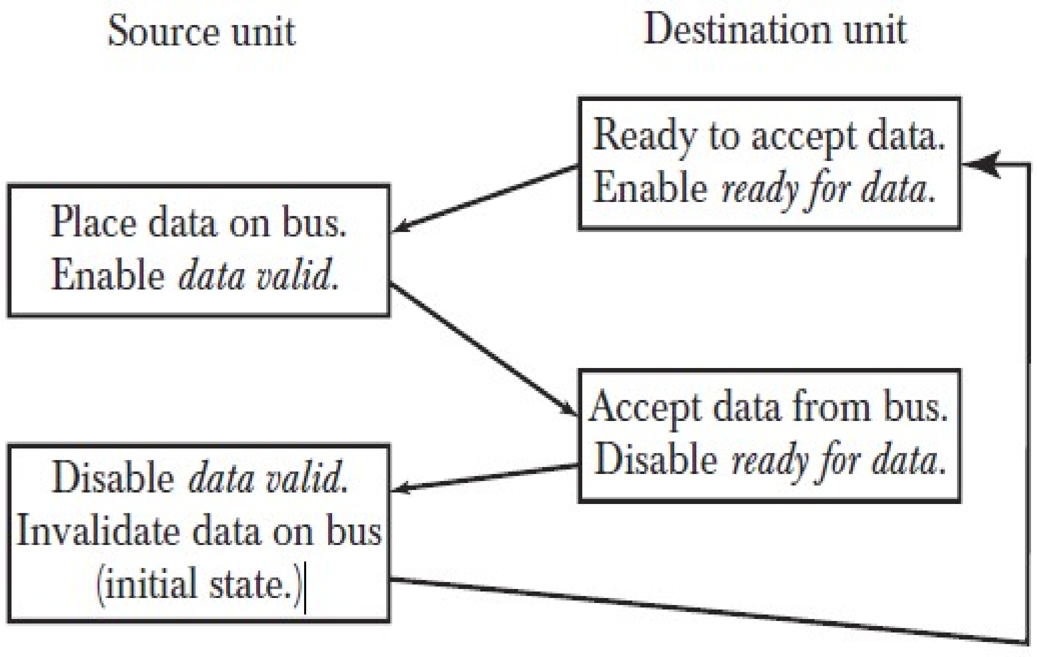
(a) Block diagram



(b) Timing diagram

Two – Wire Control (Handshaking)

Destination initiated transfer using handshaking


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.)

(c) Sequence of events

Two – Wire Control (Handshaking)

Time Out-

•The successful completion of a data transfer in handshaking relies on active participation by both units.

•If one unit is faulty, the data transfer will not be completed.

•Such an error can be detected by means of a *timeout* mechanism, which produces an alarm if the data transfer is not completed within a predetermined time.

•The timeout is implemented by means of an internal clock that starts counting time when the unit enables one of its handshaking control signals.

•If the return handshake signal does not respond within a given time period, the unit assumes that an error has occurred.

•The timeout signal can be used to interrupt the processor and hence execute a service routine that take appropriate error recovery action.

Asynchronous Serial Transfer

- A serial asynchronous data transmission technique used in many interactive terminals employs special bits that are inserted at both ends of the character code.
- **With this technique, each character consists of three parts: a start bit, the character bits, and stop bits.**

Asynchronous Serial Transfer

Start Bit -

•The convention is that the transmitter rests at the 1-state when no characters are transmitted.

•The **first bit, called the start bit, is always a 0** and is used to indicate the beginning of a character.

Stop Bit –

•The **last bit called the stop bit is always a 1**.

•After the character bits are transmitted, one or two stop bits are sent.

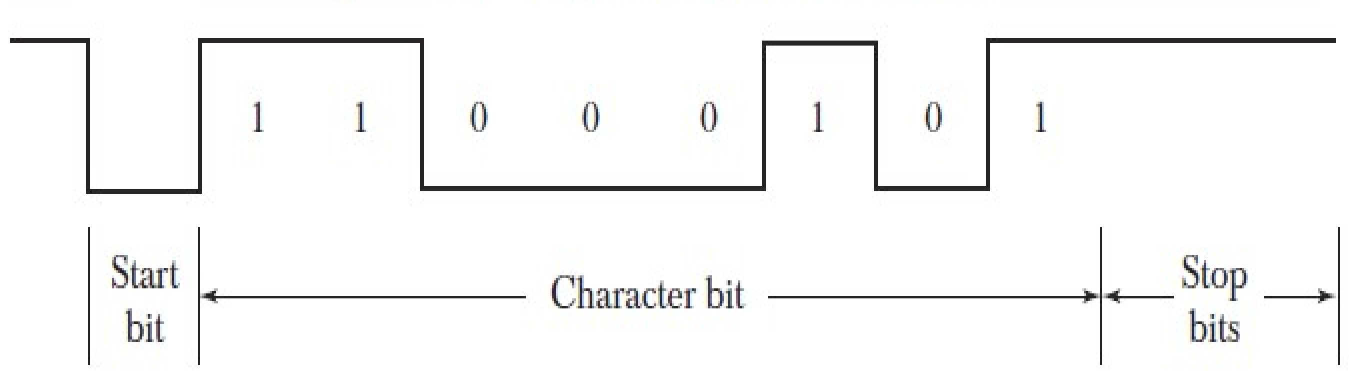
•The stop bits are always in the 1-state and frame the end of the character to signify the idle or wait state.

At the end of the character the line is held at the 1-state for a period of at least one or two bit times so that both the transmitter and receiver can resynchronize.

Asynchronous Serial Transfer

- Consider the serial transmission of a terminal whose transfer rate is 10 characters per second.
- Each transmitted character consists of a start bit, eight information bits, and two stop bits, for a total of 11 bits.

Figure 11-7 Asynchronous serial transmission.



Asynchronous Serial Transfer

A transmitted character can be detected by the receiver from knowledge of the transmission rules:

1.When a character is not being sent, the line is kept in the 1-state.
2.The initiation of a character transmission is detected from the start bit, which is always 0.
3.The character bits always follow the start bit.
4.After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for at least one bit time.

Asynchronous Serial Transfer

Baud rate -

•The ***baud rate*** **is defined as the rate at which serial information is transmitted** and is equivalent to the data transfer in bits per second.

UART-

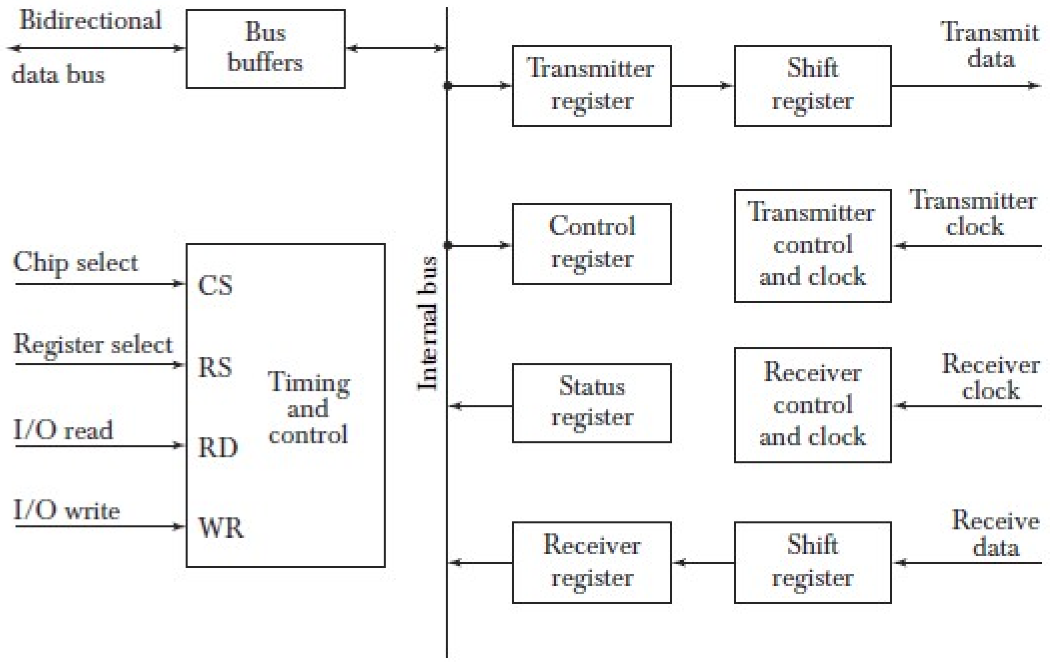
•A circuit is called an *asynchronous communication interface* or a *universal asynchronous receiver transmitter* (UART).

•The terminal has a keyboard and a printer

Asynchronous Communication Interface

- The block diagram of an asynchronous communication interface is shown in Fig. 11-8.
- It functions as both a transmitter and a receiver.
- **The operation of the asynchronous communication interface is initialized by the CPU by sending a byte to the control register.**
- The initialization procedure places the interface in a specific mode of operation as it defines certain parameters such as the baud rate to use, how many bits are in each character, whether to generate and check parity, and how many stop bits are appended to each character.
- **Two bits in the status register are used as flags.**
- **One bit is used to indicate whether the transmitter register is empty and another bit is used to indicate whether the receiver register is full.**

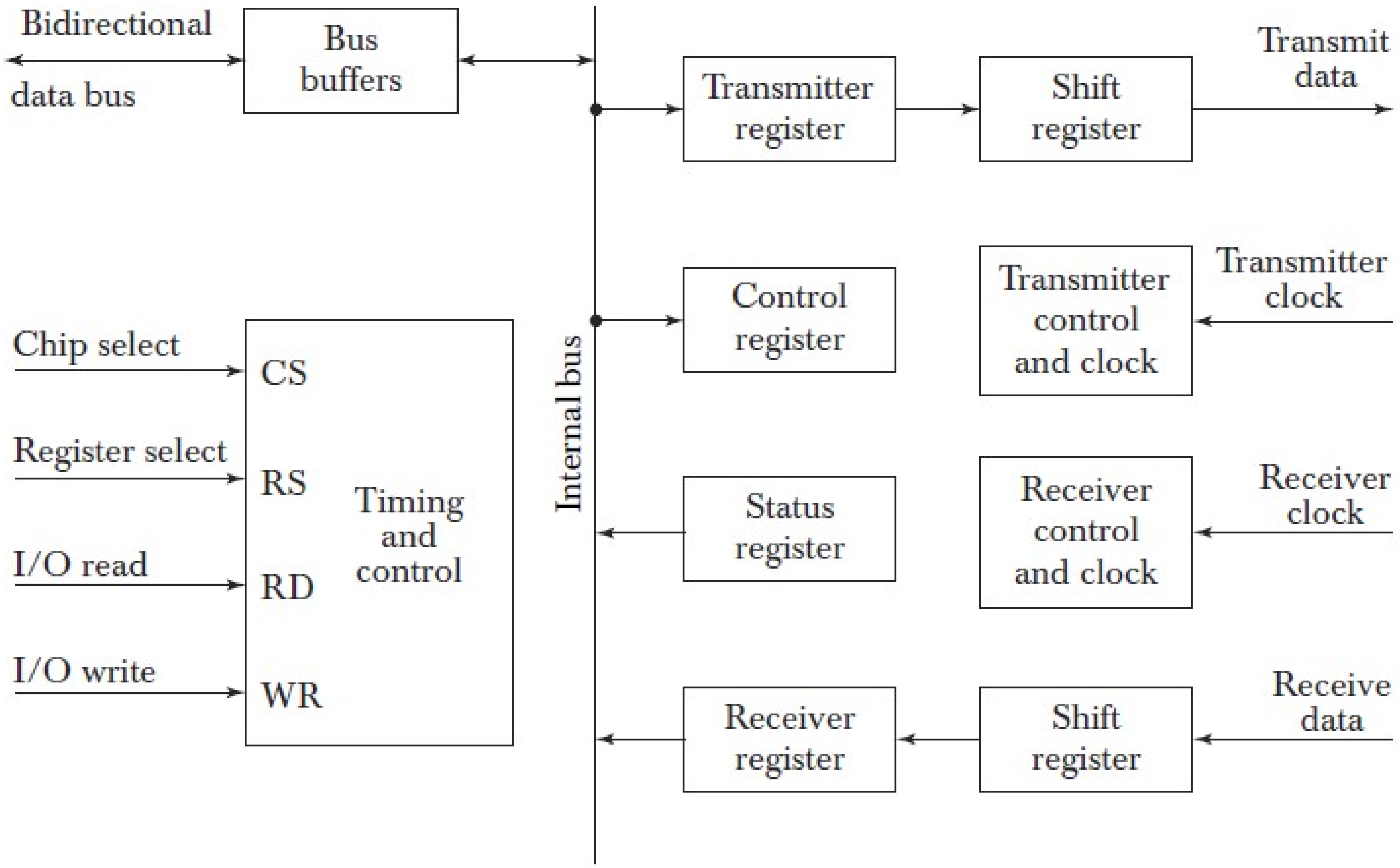
Block Diagram of Typical Asynchronous Communication Interface



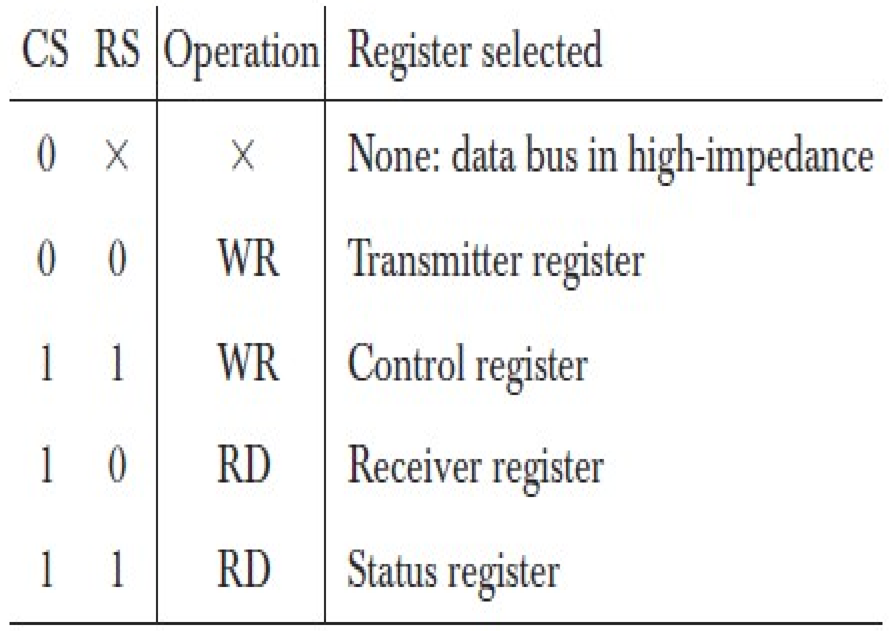
CS	RS	Operation	Register selected
0	×	×	None: data bus in high-impedance
0	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

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

Block Diagram of Typical Asynchronous Communication Interface



Block Diagram of Typical Asynchronous Communication Interface



CS	RS	Operation	Register selected
0	×	×	None: data bus in high-impedance
0	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.

Asynchronous Communication Interface

Errors

•The CPU can read the status register at any time to check if any errors have occurred.

•Three possible errors that the interface checks during transmission are **parity error, framing error, and overrun error.**

1.Parity error occurs if the number of 1's in the received data is not the correct parity.

2.A framing error occurs if the right number of stop bits is not detected at the end of the received character.

3.An overrun error occurs if the CPU does not read the character from the receiver register before the next one becomes available in the shift register.

Overrun error results in a loss of characters in the received data stream.

Modes of Transfer

- Binary information received from an external device is usually stored in memory for later processing.
- 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:

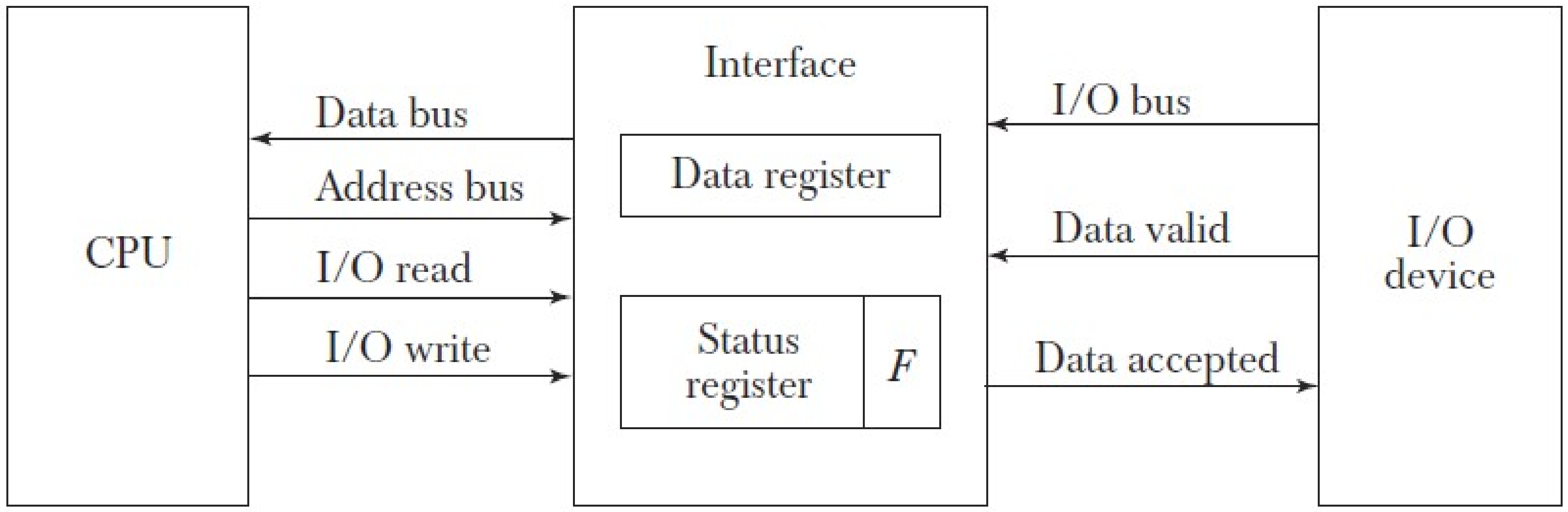
1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

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.**
- **In the programmed I/O method, the I/O device does not have direct access to memory.**
- A transfer from an I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from the device to the CPU and a store instruction to transfer the data from the CPU to memory.
- An example of data transfer from an I/O device through an interface into the CPU is shown in Fig. 11-10.

Programmed I/O

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



F = Flag bit

Programmed I/O

- A flowchart of the program that must be written for the CPU is shown in Fig. 11-11.

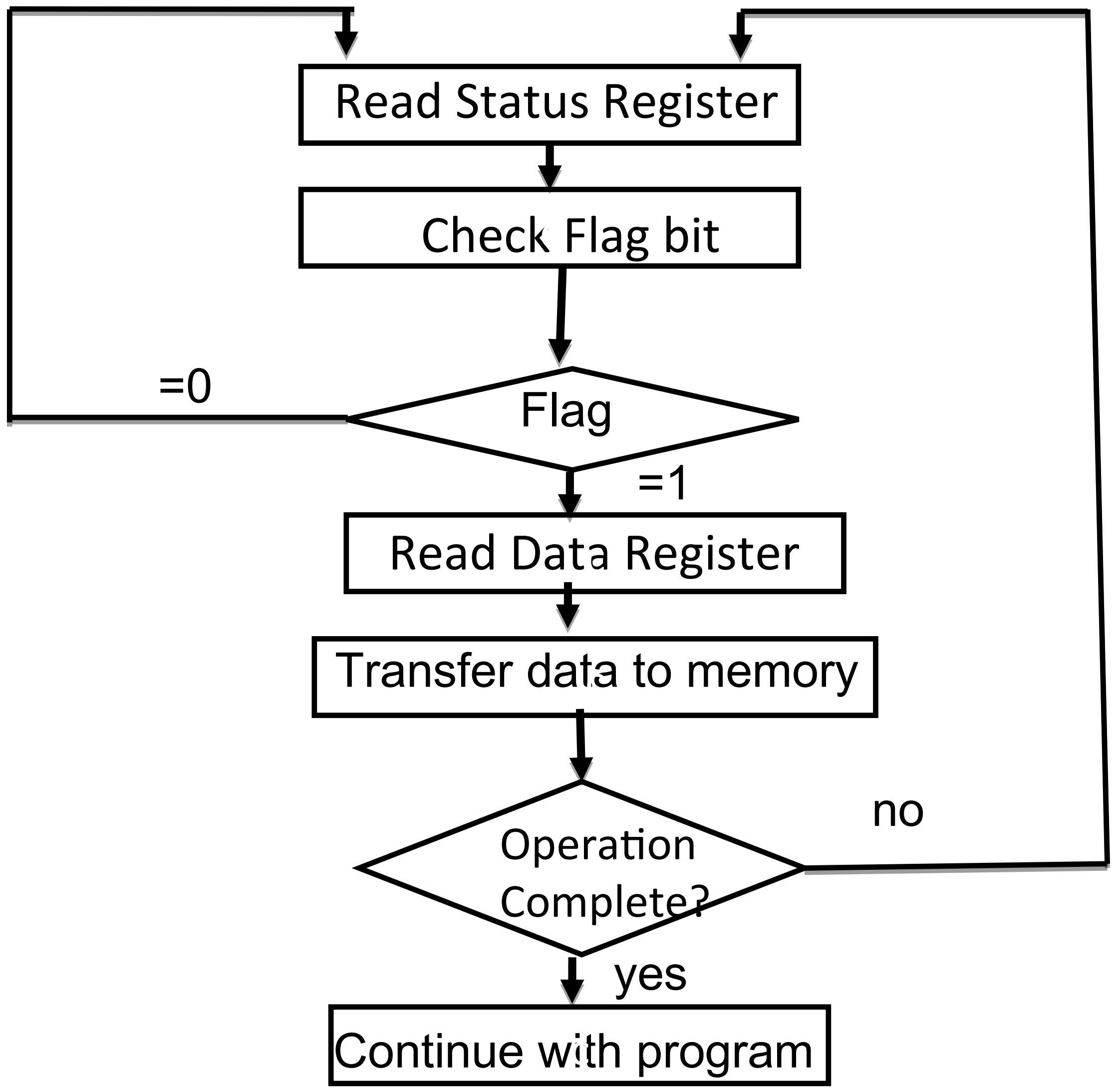
- It is assumed that the device is sending a sequence of bytes that must be stored in memory.

- The transfer of each byte requires three instructions:

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.

Programmed I/O



Programmed I/O

Disadvantage-

•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 an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

Interrupt Initiated I/O

- An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.
- This **mode of transfer uses the interrupt facility**.
- While the CPU is running a program, it does not check the flag.
- However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.

Interrupt Initiated I/O

- In principle, there are two methods for accomplishing this.

1. ***Vectored interrupt***
2. ***Non-vectored interrupt***

- **In a non-vectored interrupt, the branch address is assigned to a fixed location in memory.**
- **In a vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the *interrupt vector.***
- In some computers the interrupt vector is the first address of the I/O service routine.
- In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the I/O service routine is stored

Priority Interrupt

- **A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.**

- The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.

- Higher-priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences.

- **Devices with highspeed transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority**.

- **When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.**

Priority Interrupt

Establishing the priority of simultaneous interrupts can be done by –

1.Software – Polling
2.Hardware –
  a) **Daisy chaining Priority (Serial connection)**
  b) **Parallel Priority Interrupt (Parallel Connection)**

Priority Interrupt

1. **Polling**

- **A polling procedure is used to identify the highest-priority source by software means.**
- In this method there is one common branch address for all interrupts.
- The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence.
- **The order in which they are tested determines the priority of each interrupt.**
- The highest priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source.

Priority Interrupt

- The disadvantage of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device.
- In this situation a hardware priority-interrupt unit can be used to speed up the operation.

Daisy-Chaining Priority

- **The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.**

- The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain.
-
- This method of connection between three devices and the CPU is shown in Fig. 11-12.

- The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU.

- The farther the device is from the first position, the lower is its priority.

Daisy-Chaining Priority

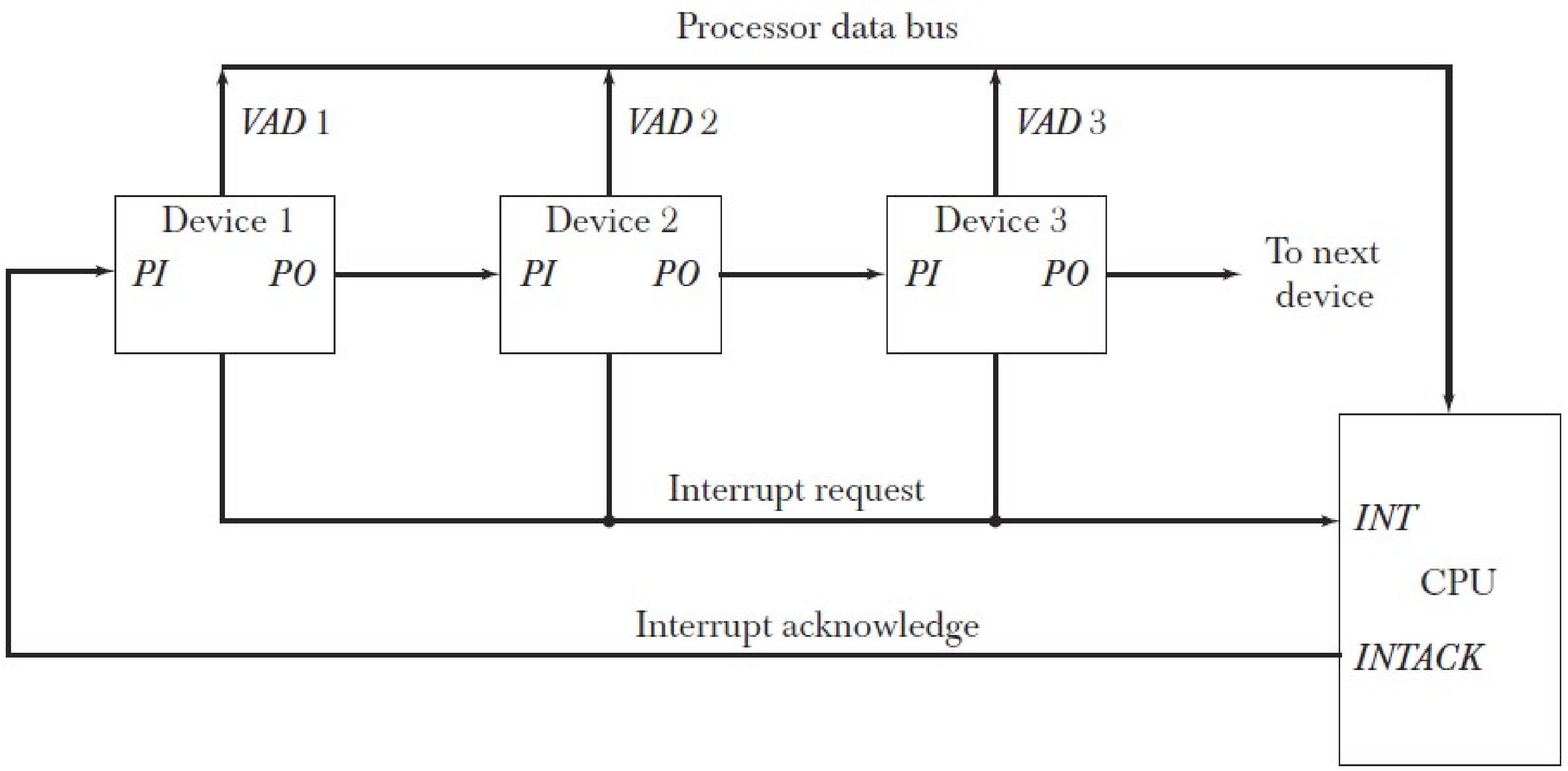


Figure 11-12 Daisy-chain priority interrupt.

Parallel Priority Interrupt

- The parallel priority interrupt method uses-

1. Interrupt Register-

- A register whose bits are set separately by the interrupt signal from each device.
- Priority is established according to the position of the bits in the register.

2. Mask Register -

- It’s purpose is to control the status of each interrupt request.
- The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced.
- It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower-priority device is being serviced.

The priority logic for a system of four interrupt sources is shown in Fig. 11-14.

Parallel Priority Interrupt

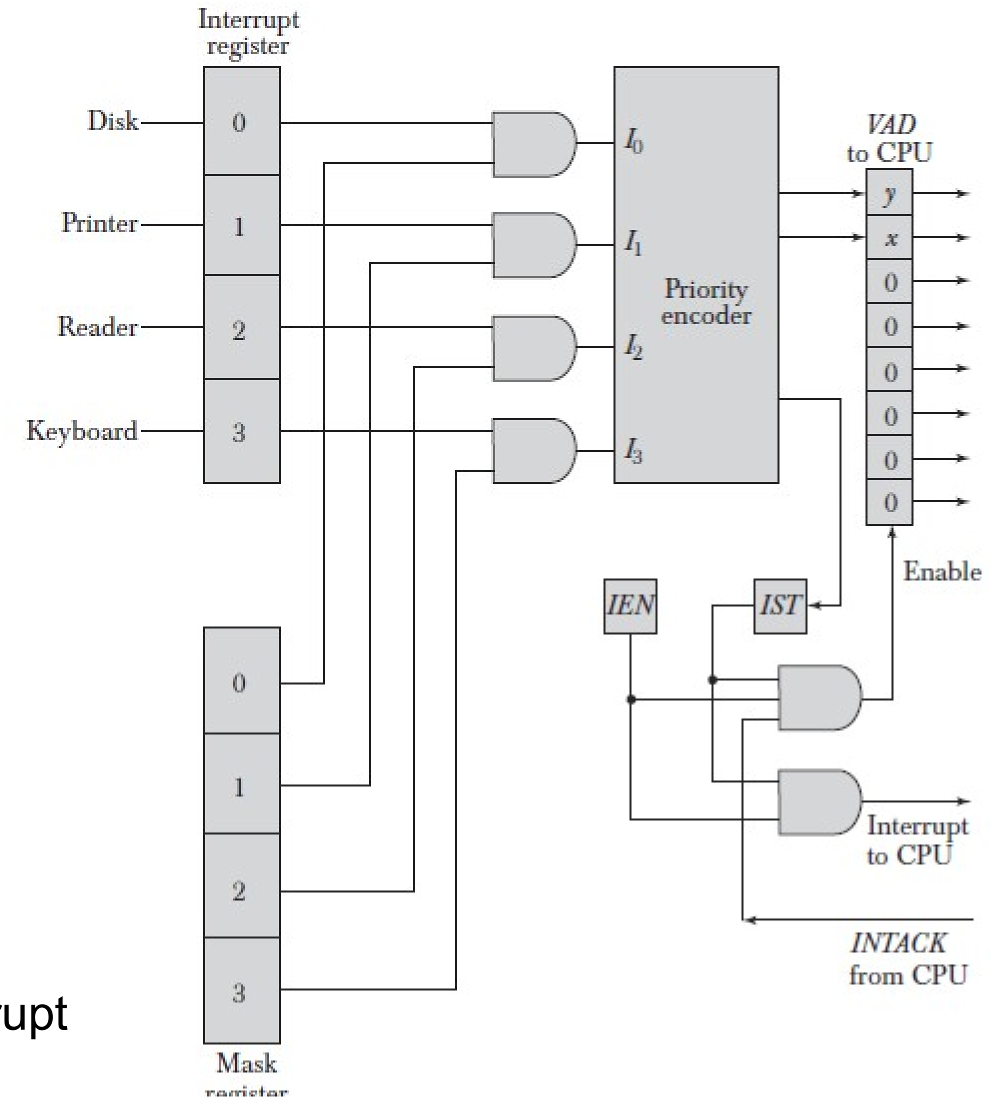
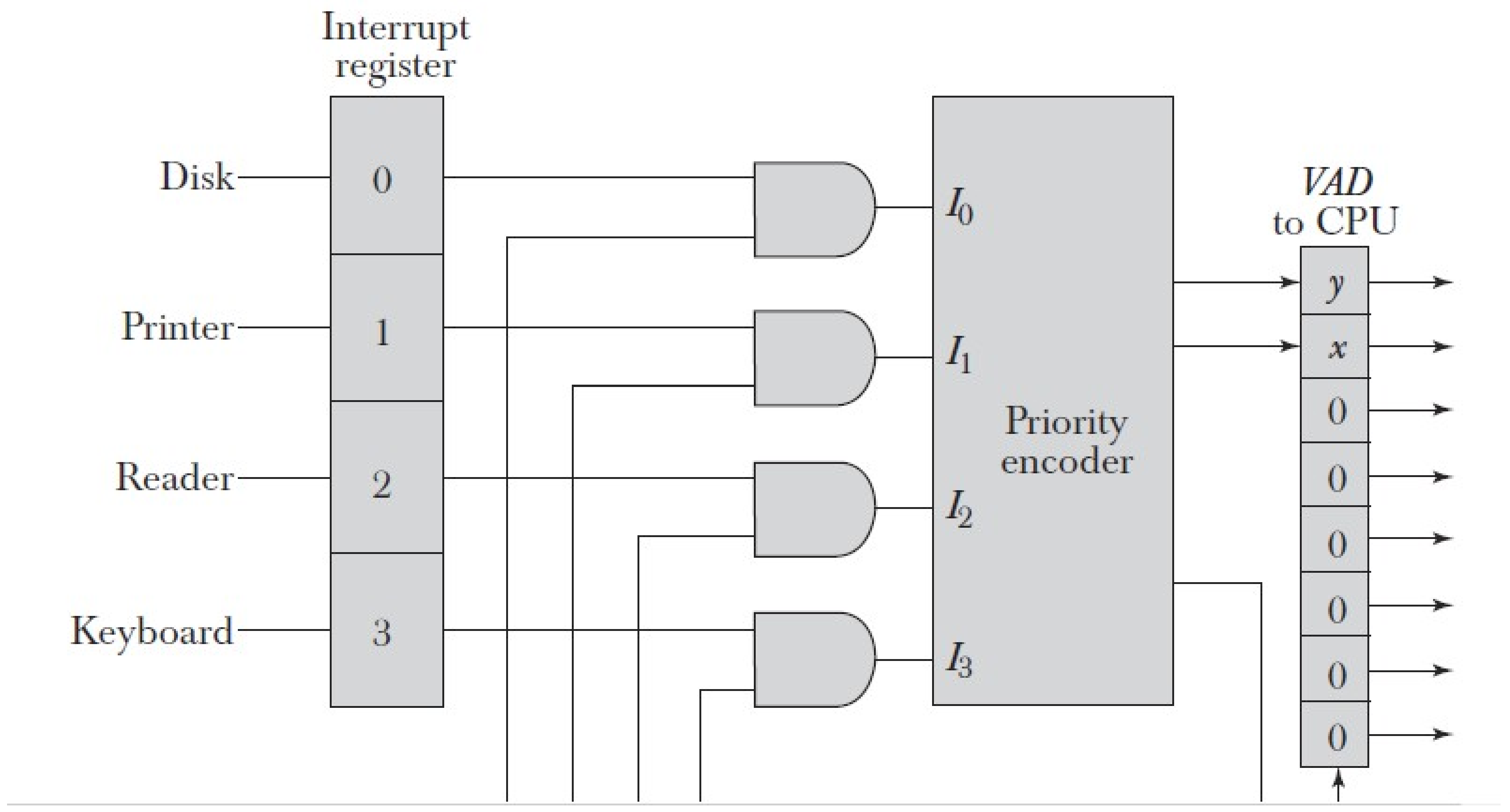
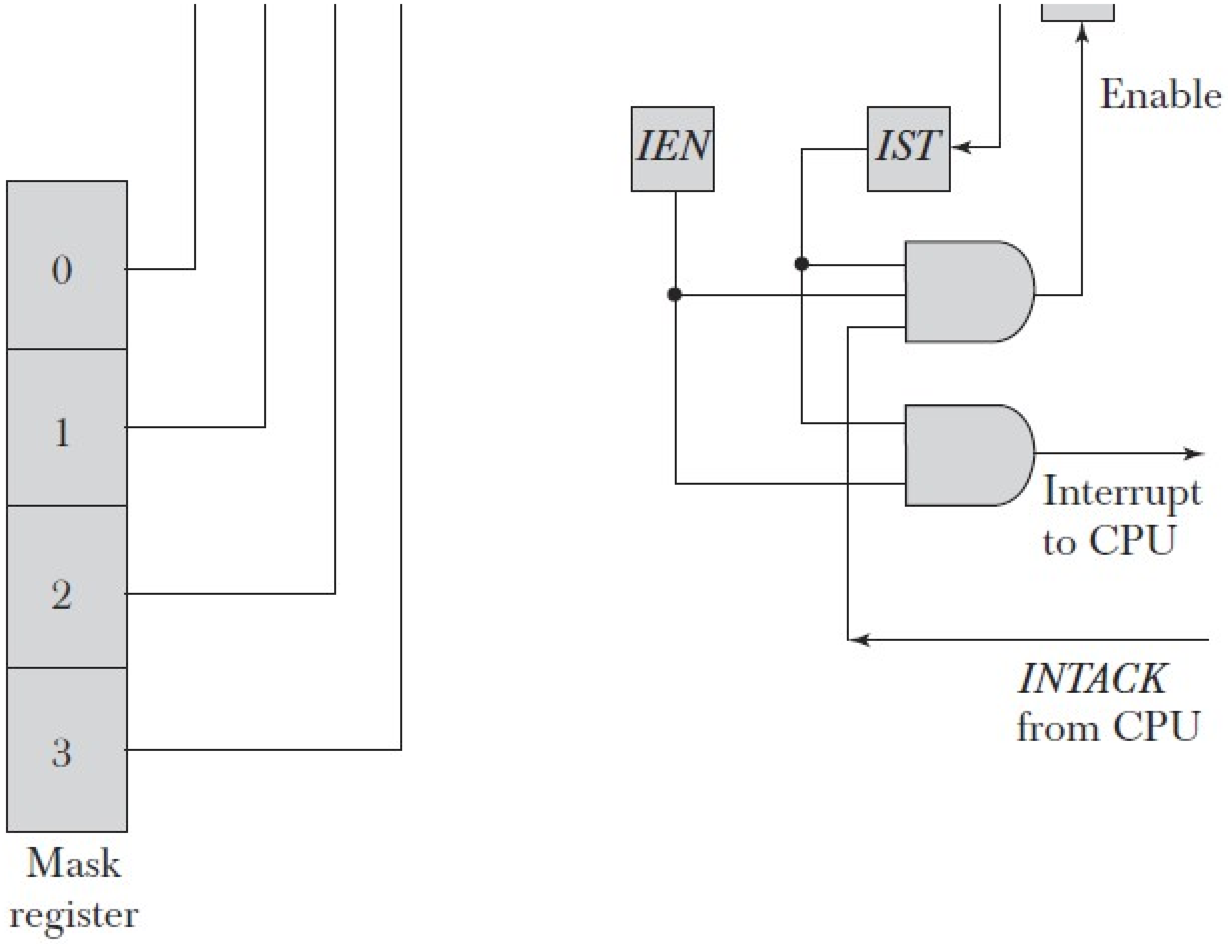


Fig: 11-14 Priority Interrupt Hardware

Parallel Priority Interrupt

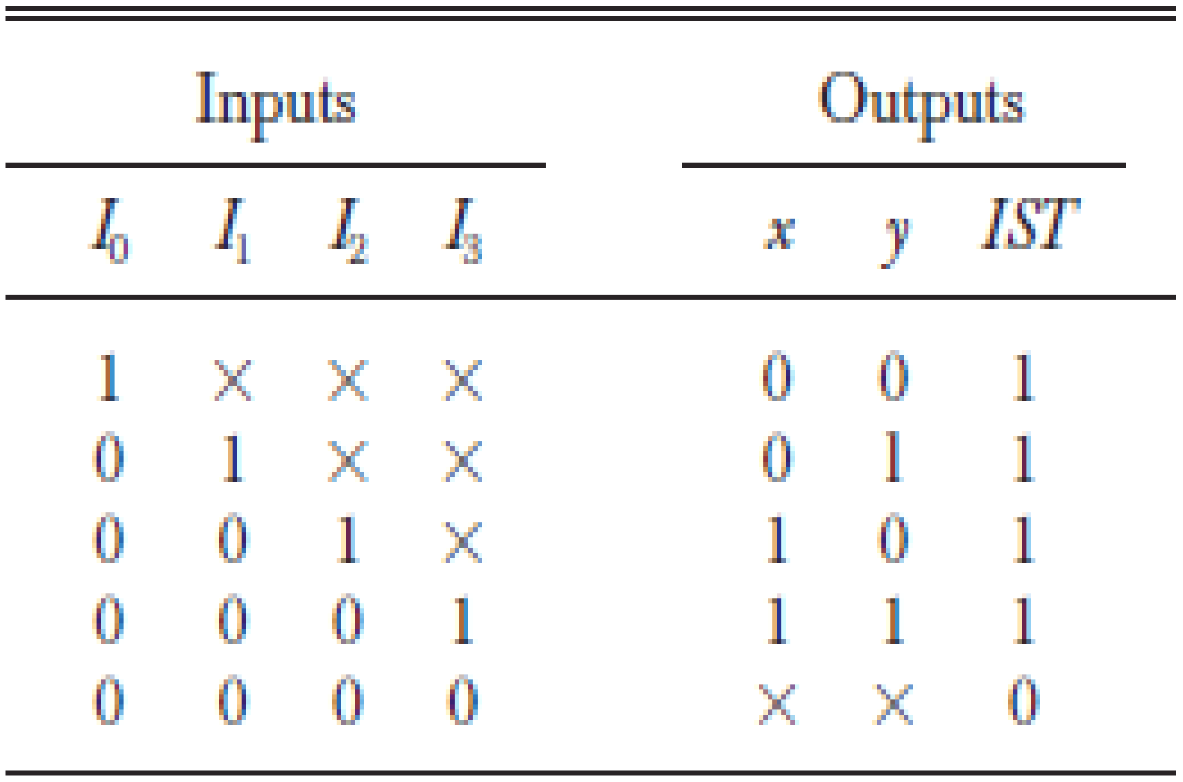


Parallel Priority Interrupt



Priority Interrupt

Priority Encoder



Inputs				Outputs		
I_0	I_1	I_2	I_3	x	y	IST
1	×	×	×	0	0	1
0	1	×	×	0	1	1
0	0	1	×	1	0	1
0	0	0	1	1	1	1
0	0	0	0	×	×	0

Interrupt Cycle

- The interrupt enable flip-flop *IEN* shown in Fig. 11-14 can be set or cleared by program instructions.
- 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 microoperations:

$SP \leftarrow SP\ \ 1$ 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 next instruction

- The CPU pushes the return address from *PC* into the stack. It then acknowledges the interrupt by enabling the *INTACK* line.

Interrupt Cycle

Software Routines

A priority interrupt system is a combination of hardware and software techniques. The computer must also have software routines for servicing the interrupt requests and for controlling the interrupt hardware registers.

Initial and Final Operations

•Each interrupt service routine must have an initial and final set of operations for controlling the registers in the hardware interrupt system.

•Remember that the interrupt enable *IEN* is cleared at the end of an interrupt cycle.

•This flipflop must be set again to enable higher-priority interrupt requests, but not before lower-priority interrupts are disabled.

Interrupt Cycle

- **The initial sequence of each interrupt service routine must have instructions to control the interrupt hardware in the following manner**:

1. Clear lower-level mask register bits.
2. Clear interrupt status bit *IST.*
3. Save contents of processor registers.
4. Set interrupt enable bit *IEN.*
5. Proceed with service routine.

Interrupt Cycle

- **The final sequence in each interrupt service routine must have instructions to control the interrupt hardware in the following manner:**

1. Clear interrupt enable bit *IEN*.
2. Restore contents of processor registers.
3. Clear the bit in the interrupt register belonging to the source that has been serviced.
4. Set lower-level priority bits in the mask register.
5. Restore return address into *PC* and set *IEN.*

Direct Memory Access (DMA)

- The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU.
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.
- This **transfer technique is called direct memory access (DMA).**
- During DMA transfer, the CPU is idle and has no control of the memory buses.
- **A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.**

Direct Memory Access (DMA)

- The CPU may be placed in an idle state in a variety of ways.
- One common method extensively used in microprocessors is to disable the buses through special control signals.

1. **Bus request (BR)**
2. **Bus Grant (BG)**

Bus request (BR)

- The *bus request* (*BR*) input is used by the DMA controller to request the CPU to relinquish control of the buses.
- When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state.

Direct Memory Access (DMA)

Bus Grant (BG)

•The CPU activates the *bus grant* (*BG*) output to inform the external DMA that the buses are in the high-impedance state.

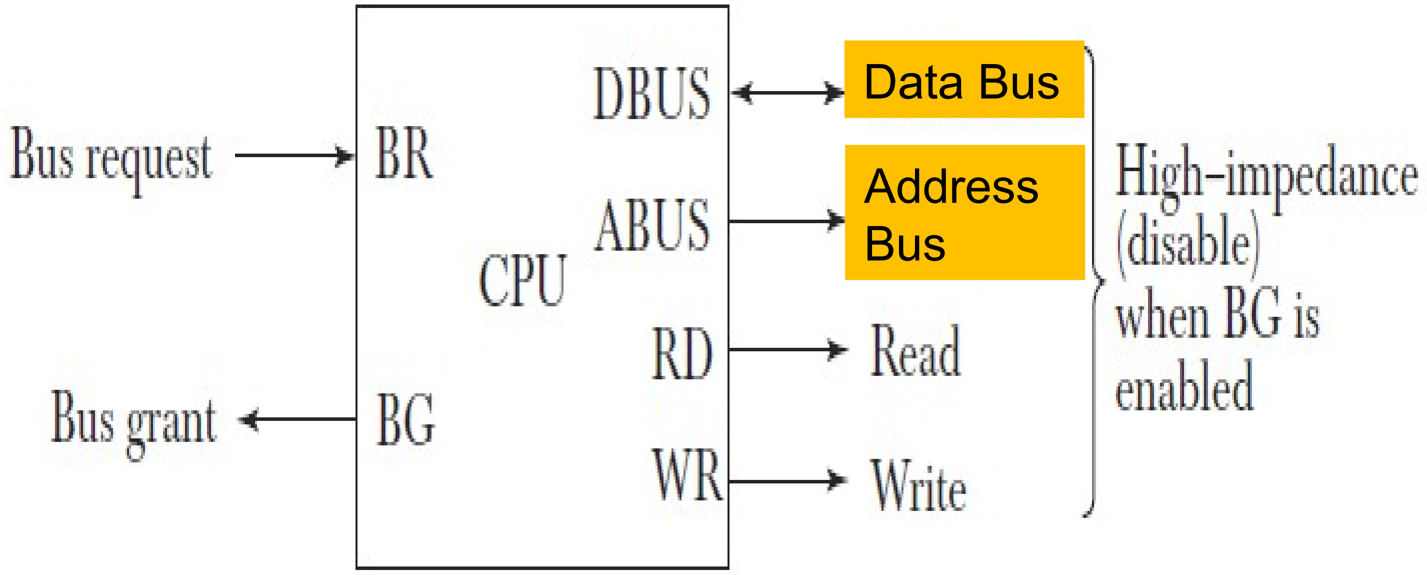
•The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention.

•When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

Figure 11-16 shows two control signals in the CPU that facilitate the DMA transfer.

Direct Memory Access (DMA)

Figure 11-16 CPU bus signals for DMA transfer.



Direct Memory Access (DMA)

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

1. Burst Transfer
2. Cycle stealing

BURST TRANSFER

- In DMA *burst transfer,* **a block sequence consisting of a number of memory words is transferred in a continuous burst** while the DMA controller is master of the memory buses.
- This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred.

Direct Memory Access (DMA)

CYCLE STEALING

- This technique allows DMA controller to **transfer one data word at a time, after which it must return control of the buses to the CPU**.

- The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to **“steal” one memory cycle.**

DMA Controller

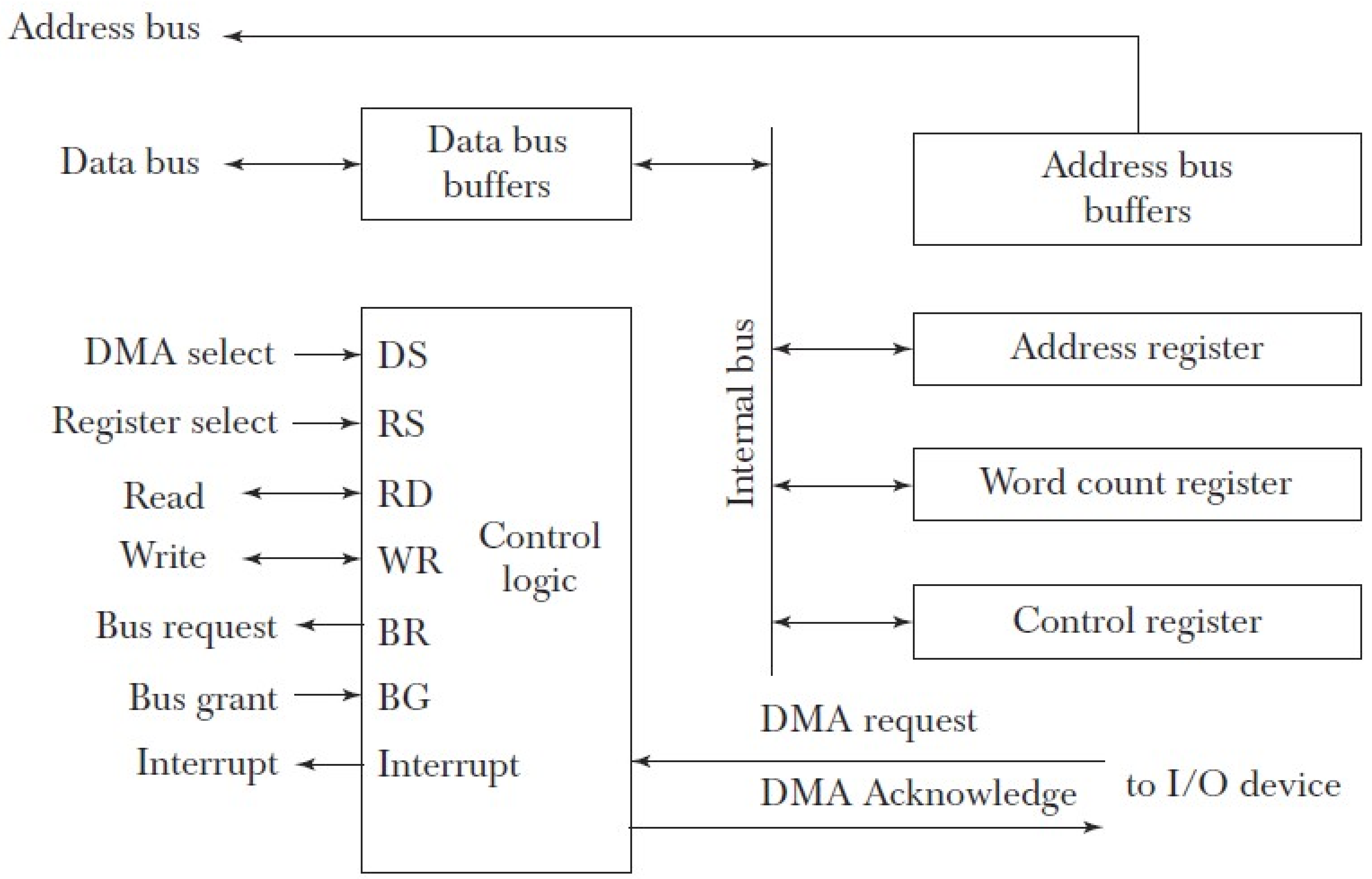
- Figure 11-17 shows the block diagram of a typical DMA controller.
- The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the *DS* (DMA select) and *RS* (register select) inputs.
- The *RD* (read) and *WR* **(**write**)** inputs are bidirectional.
- When the *BG* (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.
- When *BG* = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the *RD* or *WR* control.
- The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.

DMA Controller

- The DMA controller has three registers: an address register, a word count register, and a control register.
- **The address register contains an address to specify the desired location in memory.** The address bits go through bus buffers into the address bus.
- *The address register is incremented after each word that is transferred to memory.*
- **The word count register holds the number of words to be transferred.**
- This *register is decremented by one after each word transfer and internally tested for zero.*
- **The control register specifies the mode of transfer**.
- All registers in the DMA appear to the CPU as I/O interface registers.
- Thus the CPU can read from or write into the DMA registers under program control via the data bus.

DMA Controller

Figure 11-17 Block diagram of DMA controller.



DMA Transfer

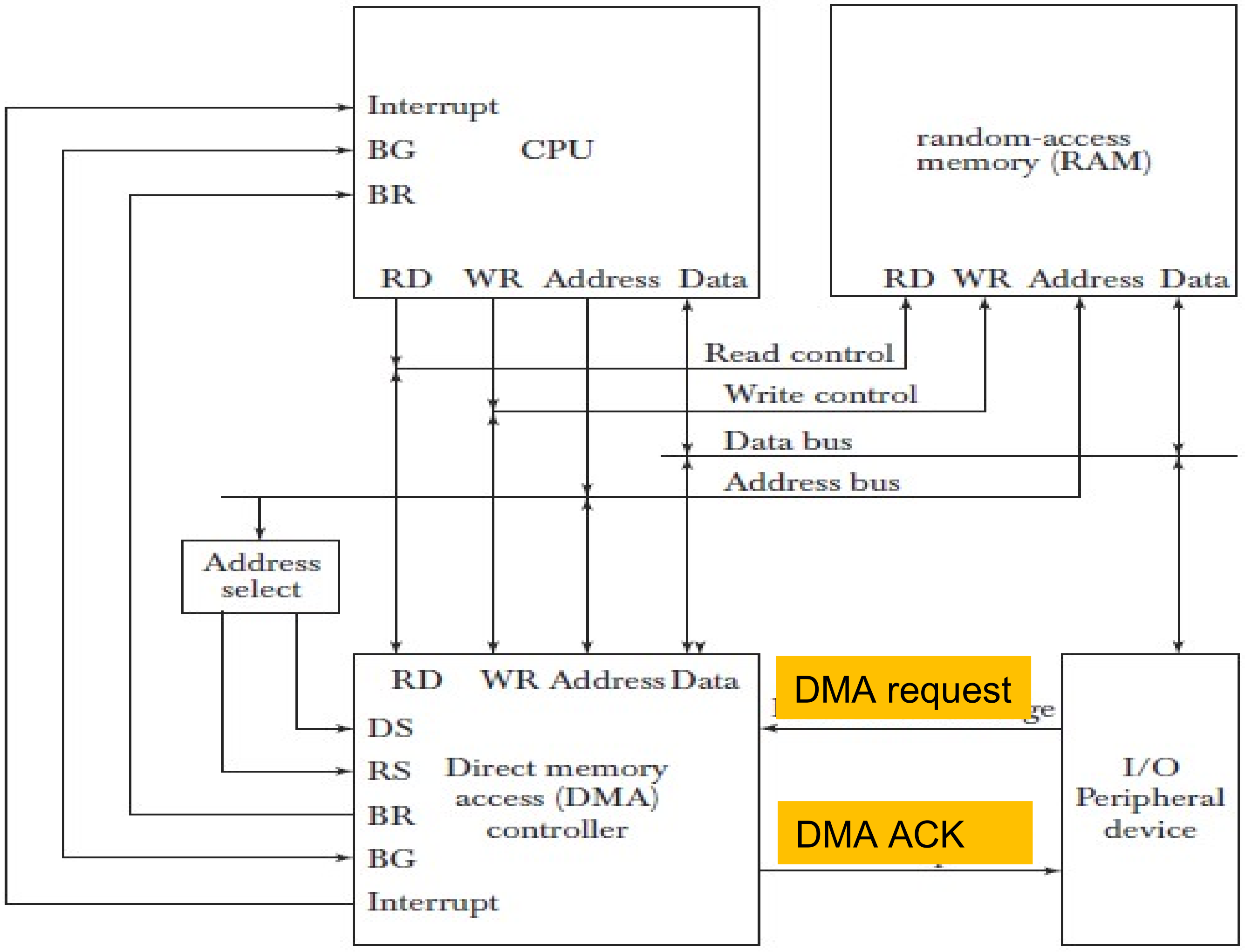


Figure 11-18 DMA transfer in a computer system.

Input-Output Processor (IOP)

- **An input–output processor (IOP) may be classified as a processor with direct memory access capability that communicates with I/O devices.**
- In this configuration, the computer system can be divided into a memory unit, and a number of processors comprised of the CPU and one or more IOPS.
- A processor that communicates with remote terminals over telephone and other communication media in a serial fashion is called a **data communication processor (DCF).**

Input-Output Processor (IOP)

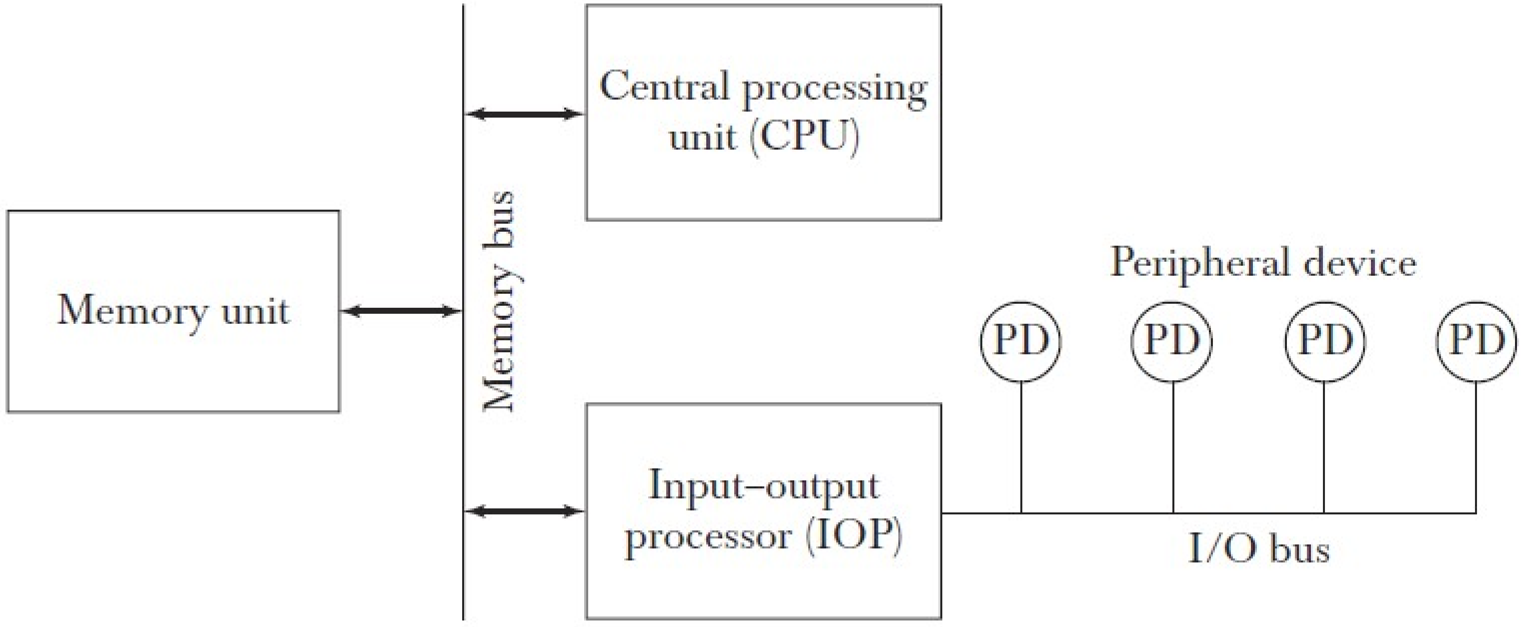


Figure 11-19 Block diagram of a computer with I/O processor.

Input-Output Processor (IOP)

CPU—IOP Communication

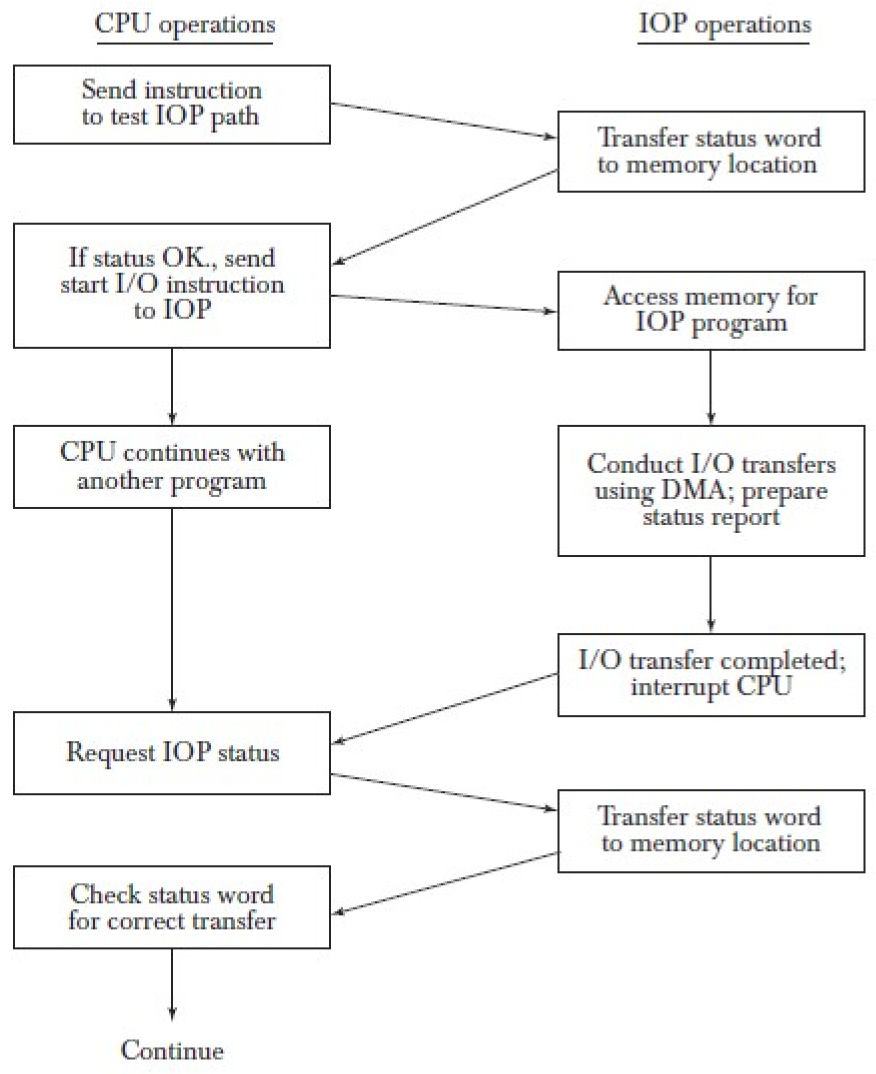
•The communication between CPU and IOP may take different forms**.**

•In most cases the memory unit acts as a message center where each processor leaves information for the other.

•The sequence of operations may be carried out as shown in the flowchart of Fig. 11-20.

Input-Output 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

Input-Output Processor (IOP)

CPU—IOP Communication

•The sequence of operations may be carried out as shown in the flowchart of Fig. 11-20.

•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 memory to decide what to do next.

•If all is in order, the CPU sends the instruction to start I/O transfer.

•The memory address received with this instruction tells the IOP where to find it program.

Input-Output Processor (IOP)

- The CPU can now continue with another program while the IOP is busy with the I/O program.
- Both programs refer to memory by means of DMA transfer.
- When the IOP terminates the execution of its program, it sends an interrupt request to the CPU.
- The CPU responds to the interrupt by issuing an instruction to read the status from the IOP.
- The IOP responds by placing the contents of its status report into a specified memory location.
- The status word indicates whether the transfer has been completed or if any errors occurred during the transfer.
- From inspection of the bits in the status word, the CPU determines if the I/O operation was completed satisfactorily without errors.

Serial Communication

Data communication processor

- A data communication processor is an I/O processor that distributes and collects data from many remote terminals connected through telephone and other communication lines.
- It is a specialized I/O processor designed to communicate directly with data communication networks.

Serial Communication

Difference between Data communication processor & IOP

•The most striking difference between an I/O processor and a data communication processor is in the way the processor communicates with the I/O devices.

1.An I/O processor communicates with the peripherals through a common I/O bus that is **comprised of many data and control lines.** All peripherals share the common bus and use it to transfer information to and from the I/O processor.

2.A data communication processor communicates with each terminal through a single pair of wires. Both data and control information are transferred in a serial fashion with the result that the transfer rate is much slower.

Serial Communication

Synchronous transmission

•It does not use start-stop bits to frame characters and therefore makes more efficient use of the communication link.

•Highspeed devices use synchronous transmission to realize this efficiency.

•The modems used in synchronous transmission have internal clocks that are set to the frequency that bits are being transmitted in the communication line.

Serial Communication

- For proper operation, it is required that the clocks in the transmitter and receiver modems remain synchronized at all times.
- The message consists of a group of bits transmitted sequentially as a block of data.
- The entire block is transmitted with special control characters at the beginning and end of the block.
- The control characters at the beginning of the block supply the information needed to separate the incoming bits into individual characters.

Serial Communication

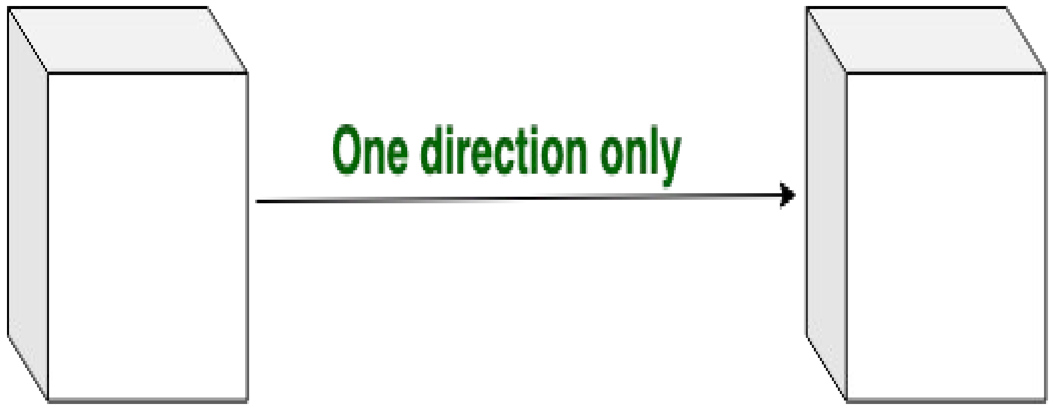
- Data can be transmitted between two points in three different modes: Simplex mode, Half duplex mode, and Full duplex mode.

1. **Simplex Mode**

- In Simplex mode, the communication is unidirectional, as on a one-way street.
- Only one of the two devices on a link can transmit, the other can only receive.
- The simplex mode can use the entire capacity of the channel to send data in one direction.
- Example: Keyboard and traditional monitors.

  The keyboard can only introduce input, the monitor can only give the output.

Serial Communication



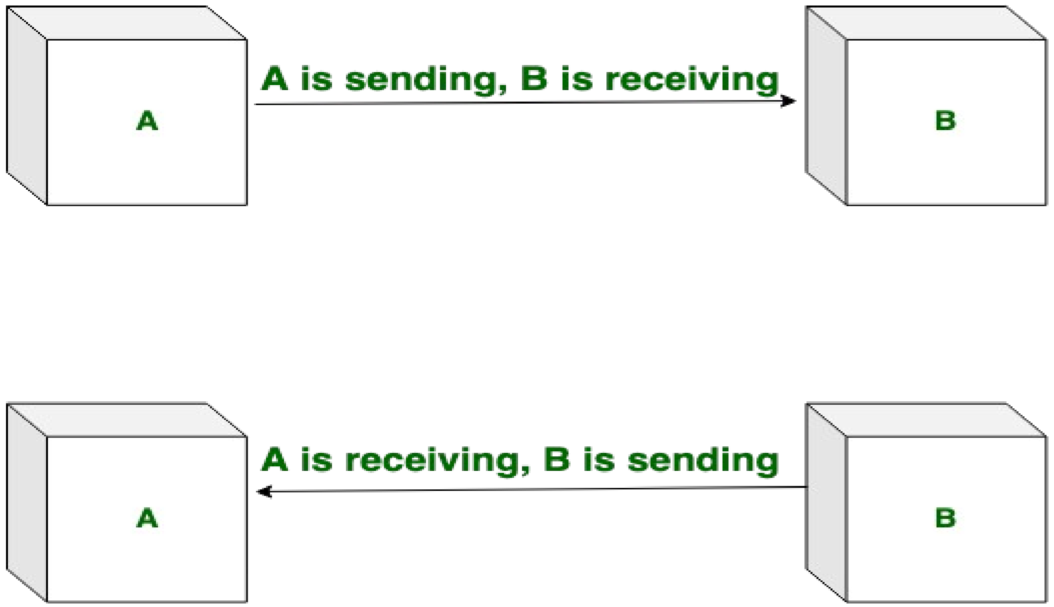
Simplex mode

Serial Communication

2. Half-Duplex transmission

- In half-duplex mode, each station can both transmit and receive, but not at the same time.
- When one device is sending, the other can only receive, and vice versa.
- The half-duplex mode is used in cases where there is no need for communication in both directions at the same time.
- The entire capacity of the channel can be utilized for each direction.
- The time required to switch a half-duplex line from one direction to the other is called turnaround time.
- Example: Walkie-talkie in which message is sent one at a time and messages are sent in both directions.

Serial Communication



Half duplex mode

Serial Communication

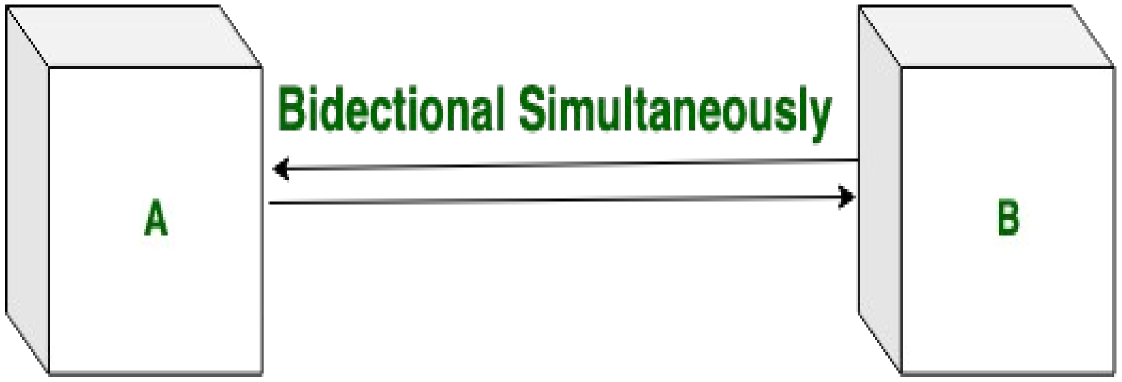
3. **Full-Duplex transmission Mode**

- **In full-duplex mode, both stations can transmit and receive simultaneously.**
- In full-duplex mode, signals going in one direction share the capacity of the link with signals going in another direction, this sharing can occur in two ways:

1. Either the link must contain two physically separate transmission paths, one for sending and the other for receiving.
2. Or the capacity is divided between signals travelling in both directions.

- Full-duplex mode is used when communication in both directions is required all the time.
- Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time

Serial Communication



Full duplex mode

Serial Communication

3. Full-Duplex transmission Mode

- **In full-duplex mode, both stations can transmit and receive simultaneously.**
- In full-duplex mode, signals going in one direction share the capacity of the link with signals going in another direction, this sharing can occur in two ways:

1. Either the link must contain two physically separate transmission paths, one for sending and the other for receiving.
2. Or the capacity is divided between signals travelling in both directions.

- Full-duplex mode is used when communication in both directions is required all the time.
- Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time

Daily Quiz

1. Can a single DMA Controller perform operations on two different disks simultaneously?

a) True b) False

2. The devices with variable speeds are usually connected using synchronous bus.

a) True **b) False**

3. In a parallel priority system, the priority of the device is obtained by adding the contents of the interrupt register and the mask register.

a) True b) False

4. How many types of modes of I/O Data Transfer

a) 3 b) 2 c) 4 d) 5

Daily Quiz

1. The method which offers higher speeds of I/O transfers is ______

a) DMA b) Interrupts c) Memory mapping d) None

2. The technique whereby the DMA controller steals the access cycles of the processor to operate is called _________.

a) Memory stealing b) Memory Con **c) Cycle stealing** d) Fast conning

3. The key feature of UART is _________

a) Its enhancement of connecting low speed devices b) Its general purpose usage c) Its simple implementation d) Its architectural design

4. When the R/W bit of the status register of the DMA controller is set to 1.

a) Write operation is performed b) Read & Write operation is performed

c**) Read operation is performed** d)None of the mentioned

Daily Quiz

1. Instructions that are read from memory by an IOP are sometimes called __________, to distinguish them from instructions that are read by the CPU.

a) Commands b) Instructions c)Program d) Subroutine

2. A _________ command is issued to activate the peripheral and to inform it what to do.

a) Control b) Status c) Data output d) Data Input

3. The main job of the interrupt system is to identify the ______ of the interrupt.

a) Source b) Signal c) Device d) Peripherals

4. The ______ is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second.

a) Stop bit b) Start bit **c) Baud rate** d) Read bit

5. UART stands for ________

a) Universal Asynchronous Receiver Transmitter b) Universal Asynchronous Relay Transmission c) Universal Accumulator Register Transfer d) None

Daily Quiz

3. CPU has two modes privileged and non-privileged. ___________ interrupt is required to change the mode from privileged to non-privileged.

a) Hardware

b) Software

4. ___________ line is a control line along which the device is allowed to send the interrupt signal.

Ans: The Interrupt-request line

5. The signal sent from processor to the device after receiving an interrupt is ____________.

Ans: interrupt acknowledge

6. What do mean by Exception & TRAP ?

7. Write down various types of interrupts

Daily Quiz

4. In memory-mapped I/O ____________

a) The I/O devices have a separate address space

b) A part of the memory is specifically set aside for the I/O operation

c) The I/O devices and the memory share the same address space

d) The memory and I/O devices have an associated address space

5. The technique whereby the DMA controller steals the access cycles of the processor to operate is called __________.

Ans: cycle stealing

6. _________ gives the required signals and addresses during DMA transfers.

Ans: DMA controller

7. Write down full form of DMA.

Daily Quiz

1. In ____________ transmission, the two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.

a) Parallel

b) Serial

c) Synchronous

d) Asynchronous

2. The _______ is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second.

a) Stop bit

b) Start bit

c) Baud rate

d) Read bit

Daily Quiz

3. In ________ data transmission, each bit of the message has its own path and the total message is transmitted at the same time.

a) Parallel

b) Serial

c) Asynchronous

d) None

4. UART stands for _________

a) Universal Asynchronous Receiver Transmitter

b) Universal Asynchronous Relay Transmission

c) Universal Accumulator Register Transfer

d) None

Daily Quiz

5. A hand-shake based protocol for data transfer is an example of ______ type of data transfer.

a) Parallel

b) Synchronous

c) Asynchronous

d) Serial

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

MCQ

8. After the completion of the DMA transfer, the processor is notified by ____________

a) Acknowledge signal

b) Interrupt signal

c) WMFC signal

d) None of the mentioned

9. The key feature of UART is __________

a) Its enhancement of connecting low speed devices

b) Its general purpose usage

c) Its simple implementation

d) Its architectural design

MCQ

10. When the R/W bit of the status register of the DMA controller is set to 1

a) Write operation is performed

b) Read & Write operation is performed

c) Read operation is performed

d)None of the mentioned

11. The technique whereby the DMA controller steals the access cycles of the processor to operate is called __________.

a) Memory stealing

b) Memory Con

c) Cycle stealing

d) Fast conning

Sessional and Old Question Papers

Sessional 1

2017-18

2018-19

2019-20

Expected Questions for University Exam

1. Explain the working of DMA controller with help of suitable diagrams.
2. Write short notes on- a) I/O Ports b) I/O Interface c) I/O channels & Processor
3. Discuss the Synchronous and Asynchronous communication with suitable examples.
4. Discuss the various type of Interrupts and Exception in computer organization with suitable examples.
5. Explain Modes of data transfer : Programmed I/O, Interrupt initiated I/O.
6. Why are the read and write control lines in DMA bidirectional?
7. Write the difference between serial and parallel communication. Explain difference between vectored and non- vectored interrupts.
8. Explain CPU - IOP Communication. Draw the flowchart showing the sequence of operations to be carried out.
9. What is FIFO buffer? Draw and explain the circuit diagram of 4X4 FIFO buffer.

Expected Questions for University Exam

10. Write down the difference between Isolated I/O and Memory mapped I/O. Also discuss the advantages and disadvantages of both.
11. What do you mean by Asynchronous data transfer? Explain source and destination initiated strobe for data transfer using block diagram and timing diagram.
12. What is DMA Controller? Draw and explain the block diagram of DMA Controller.
13. What is parallel priority interrupt? Explain with help of diagram the priority logic for four interrupt sources.
14. Define handshaking. Explain source - initiated and destination-initiated transfer using handshaking with help of block diagram and timing diagram.
15. Why does the DMA priority over CPU when both request memory transfer?
16. Explain the concept of Strobe control with the help of timing diagram.

Recap of Unit

- 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.
- An **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- Types of Interrupt: Hardware Interrupts, Software Interrupts, Maskable and Non-Maskable, Vectored and Non-Vectored and Priority Interrupt
- Data transfer to and from peripheral device can be ahndlesd in three modes:

1.Programmed I/O
2.Interrupt initiated I/O
3.DMA