

UNIT I

Introduction to Computer Peripherals

SYLLABUS


Unit I: Introduction to Computer Peripherals: interfacing of peripheral devices and working of Device Controllers. Role of Device Drivers. Review of various I/O methods – Programmed I/O, Interrupt driven I/O, DMA and I/O processor based.

INTERFACING OF PERIPHERAL DEVICES

The most common input output devices are:

- i) Monitor
- ii) Keyboard
- iii) Mouse
- iv) Printer
- v) Hard Disk etc.

Peripherals connected to a computer need special communication links for interfacing them with the central processing unit(CPU).

A decorative header with a repeating floral pattern in a dark grey color.

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

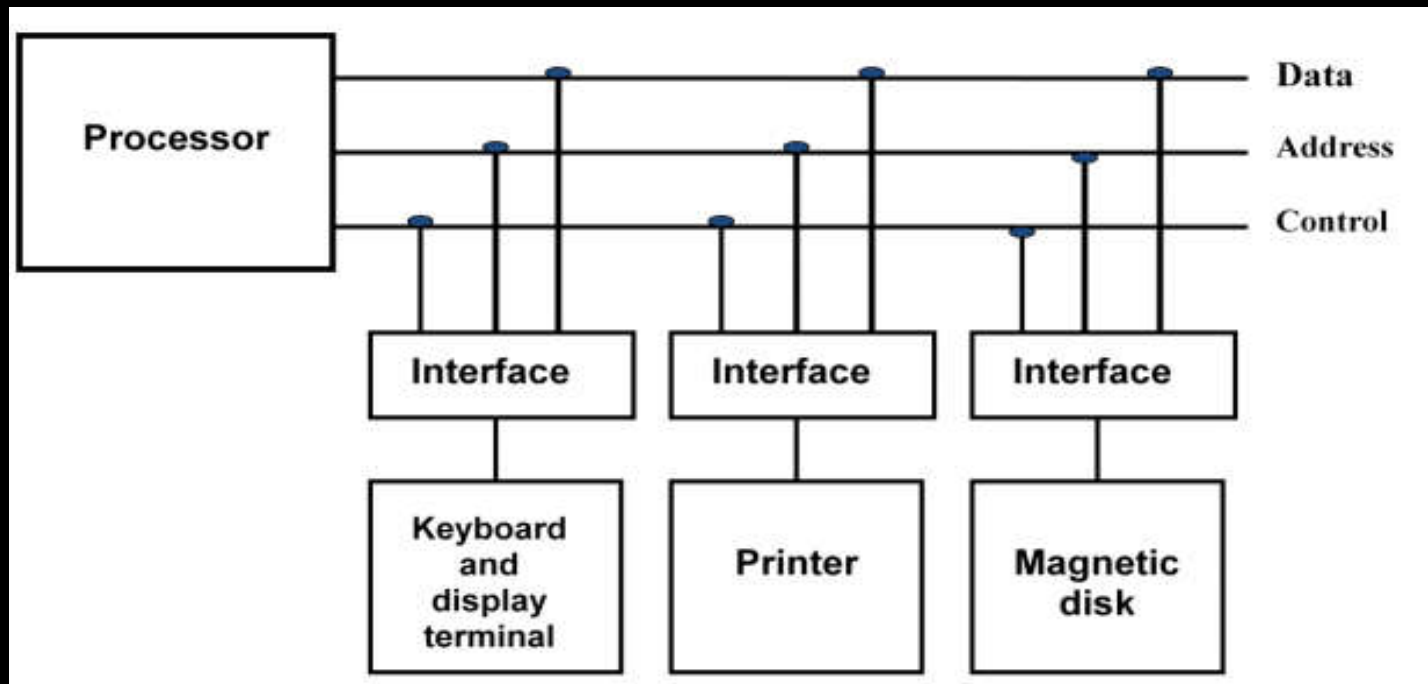
The Major Differences are:-

- Peripherals are electromechanical and electromagnetic devices, and CPU and memory are electronic devices. Therefore, a conversion of signal values may be needed.
- The data transfer rate of peripherals is usually slower than the transfer rate of CPU.
- Data codes and formats in the peripherals differ from the word format in the CPU and memory.
- The operating modes of peripherals are different from each other and must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

SOLUTION ?

- To Resolve these differences, computer systems include special hardware components between the CPU and Peripherals to supervises and synchronizes all input and output transfers.
- These special hardware components are called Interface Units because they interface between the processor bus and the peripheral devices.

I/O INTERFACING



I/O DEVICE

The input/output units consists of the following two parts:

- Mechanical component
- Electronic component

The mechanical component of the input/output unit is the device itself whereas the electronic component takes the form of a printed circuit card that can be inserted into an expansion slot.

Basically, **Device Controllers** are the electronic components of input/output units.

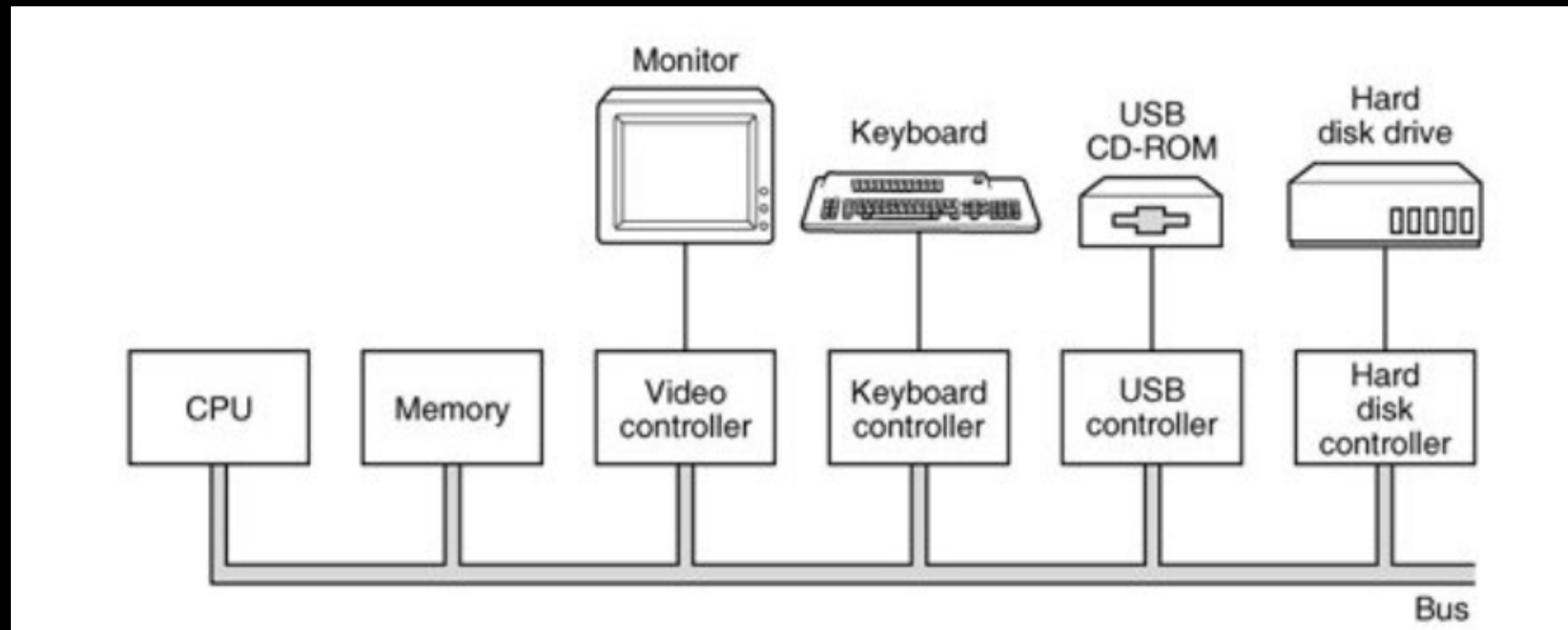
DEVICE CONTROLLER

- An electronic device in the form of chip or circuit board that controls functioning of the I/O device is called the device controller.
- Some device have their own inbuilt device controller.
- For example, the printer controller controls the paper motion, the print timing etc.

CONTD...

- Device controller reads the signal coming out and going into the CPU and act as a intermediary between the device and the Operating System.
- The Device Controller receives the data from a connected device and stores it temporarily in some special purpose registers (i.e. local buffer) inside the controller .
- The monitor is connected to the video controller while the keyboard is connected to the keyboard controller. Disk drive is connected to the disk controller, and the USB drive is connected to the USB controller. These controllers are connected to the CPU via the common bus.

LIST OF CONTROLLERS



DEVICE DRIVER

- When we get a peripheral device such as printer, scanner, keyboard or modem, the device comes together with a driver CD which needs to be installed before the device starts working.
- As soon we install the driver software into the computer, it detects and identifies the peripheral device and we become able to control the device with the computer.
- A device driver is a piece of software that allows your computer's operating system to communicate with a hardware device, the driver is written for.

CONTD...

- Device drivers works within the kernel layer of the operating system.
- Instead of accessing a device directly, an operating system loads the device drivers and calls the specific functions in the driver software in order to execute specific tasks on the device. Each driver contains the device specific codes required to carry out the actions on the device.
- Device controller is a hardware whereas device driver is a software.

HOW DEVICE DRIVER WORKS

- Let's take an example of a printer, when it is connected to the computer and the specific device driver is installed.
- When we choose an operation (like Control + P to print a document) on the printer then this command goes to the device driver through the kernel of the operating system.
- Resultantly a calling program invokes a routine in the device driver and the driver issues corresponding commands to the microcontrollers within the printer.
- Further these microcontrollers control the components of the printer like motors etc. to start printing the document.

WORKING OF DEVICE CONTROLLER

- Some devices have their own built-in controllers. For example, a disk drive has its own circuit board, which is attached to its one side. This circuit board is the disk controller.
- Each device controller maintains a set of device registers:
 - i. command registers (write-only);
 - ii. status registers (read-only);
 - iii. data registers (read/write).

CONTD...

- The CPU communicates with the I/O device through device controller of that device.
- The CPU sends signals to device controller to perform a specific I/O operation such as read or write operation.
- For example, the CPU sends a command to controller to read “n” byte of information from a serial device or to read a sector of information from a disk. In case of serial device, the controller collects a serial bit stream converts it into a block of bytes and performs necessary error corrections.

CONTD...

- The block of bytes is typically first assembled/ bit by bit, in a buffer inside the controller, and then is copied to main memory.
- Similarly, the controller for a monitor also works as a bit serial device. It reads bytes (containing the characters or graphics to be displayed) from main memory, and generates the signals used to modulate the CRT beam to display the output on screen in readable form.
- The operating system initializes the controller with a few parameters (such as number of characters per line and number of lines per screen to be displayed).
- The CPU gives the responsibility to the controller by sending commands and becomes busy to perform other tasks.
- Most of the controllers can handle multiple devices.

CONTROLLER REGISTERS

- Each controller has one or more registers that are used for communicating with the CPU. The number of registers in a module (or controller) and their functions depend on the type of module.
- A module for a simple input device may have two registers. In this case, one register is used for data holding, which is referred to as data buffer register and the other for control information, which is referred to as control register.
- The data buffer register is used to store the input value to pass to the CPU, while the control register contains control bits from CPU to command the I/O module.
- Some I/O controllers contain more than two registers that are used to control the I/O operations. For example, in disk- I/O operation, additional information is required to identify the location on the disk to perform I/O operation and the amount of data to be transferred.

CONTD...

- The CPU communicates with the controller by reading and writing bit patterns in controller's registers.
- The operating system performs I/O operations by sending commands into the controller's registers.
- For example, the IBM PC floppy disk controller accepts 15 different commands, such as Read, Write, Seek, Format etc.
- Many of the commands have parameters, which are also loaded into the controller's registers.
- When a command is accepted, the CPU gives the control to the controller and go off to do other work. When a command has been completed, the controller causes an interrupt in order to inform the operating system that operation is completed.
- The operating system again gains control of the CPU to test the results of the operation. It gets the results and checks device status by reading information from the controller registers.

ACCESSING I/O DEVICES

- A simple arrangement to connect I/O devices to a computer is to use a single bus arrangement.
- The bus enables all the devices connected to it to exchange information.
- Typically, it consists of three sets of lines used to carry address, data and control signals.
- Each I/O device is assigned a unique set of address.
- When the processor places a particular address on the address line, the device that recognizes this address responds to the commands issued on the control lines.
- The processor requests either a read or write operation, and the requested data are transferred over the data lines.
- When a I/O device and memory share the same address space, the arrangement is called as **memory-mapped I/O**.

CONTD...

- With memory-mapped I/O, any machine instruction that can access memory can be used to transfer data to or from an I/O device.
- For example, if DATAIN is the address of the input buffer associated with the keyboard, the instruction:

MOVE DATAIN, R0

Reads the data from DATAIN and stores them into processor register R0, similarly, the instruction

MOVE R0, DATAOUT

Sends the contents of the register R0 to location DATAOUT, which may be the output data buffer of a display unit or printer.

CONTD...

- Most computer system uses memory-mapped I/O.
- Some processor have special In and Out instructions to perform I/O transfers. Such mechanism is called as **I/O mapped I/O**.
- For example, processors in the Intel family have special I/O instructions and separate 16-bit address space for I/O device.
- When building a computer system based on these processors, the designer has the option of connecting I/O devices to use the special I/O address space or simply incorporating them as part of the memory address space.
- One advantage of a separate I/O address space is that I/O devices deal with fewer address lines.
- Note that a separate I/O address space does not mean that the I/O address lines are physically separate from the memory address lines.

QUESTION

Consider the hypothetical processor which supports 512k words memory. It uses the memory mapped input output configuration. In which when 2MSB bits of address are 1 & 1 then assigned to input output port. How many number of input output port address and memory address are possible in the processor respectively?

I/O INTERFACE

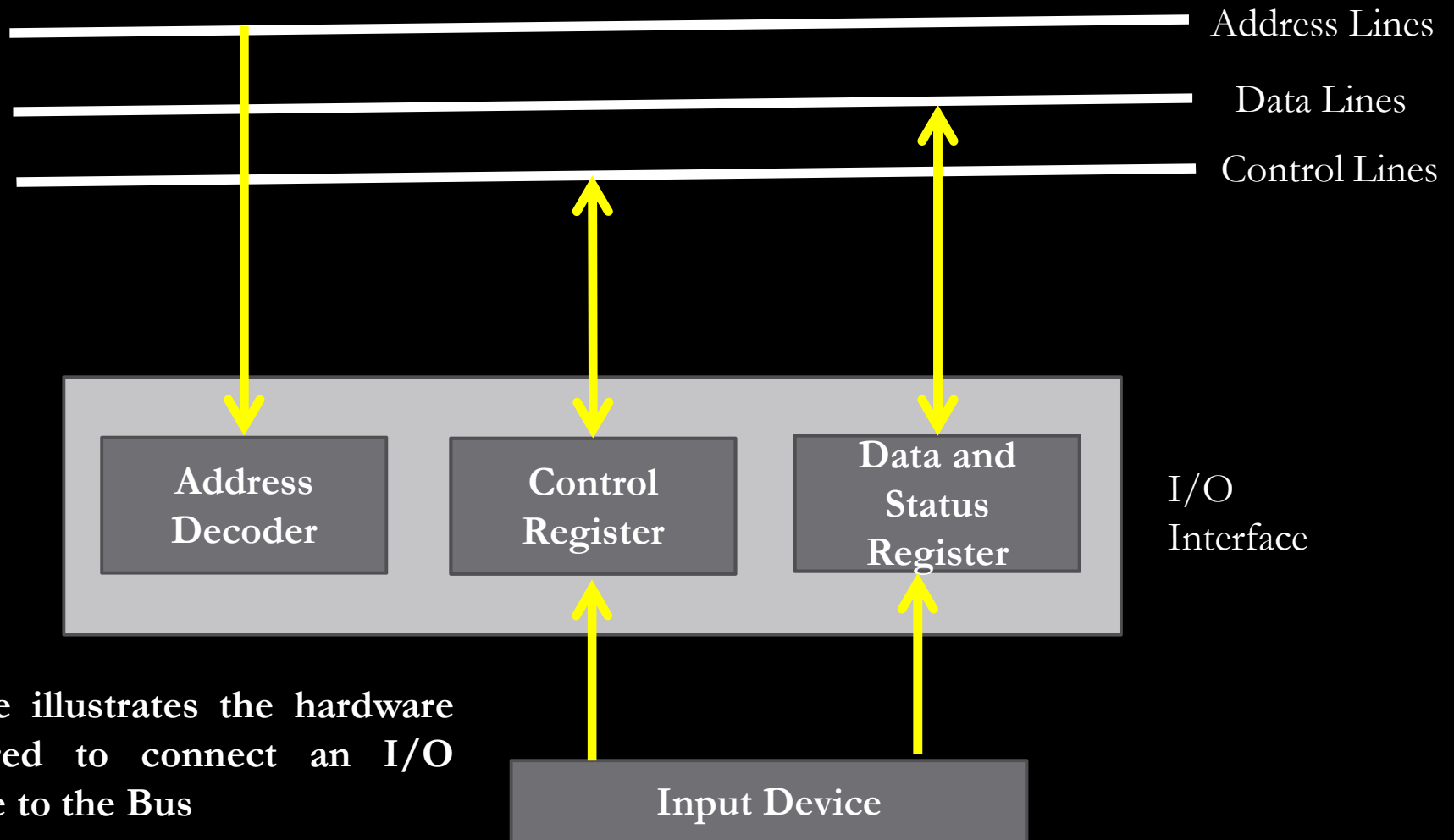


Figure illustrates the hardware required to connect an I/O device to the Bus

CONTD...

- The address decoder enables the device to recognize its address when this address appears on the address lines.
- The data register holds the data being transferred to or from the processor.
- The status register contains information relevant to the operation of the I/O device.
- Both the data and status registers are connected to the data bus and assigned unique address.

CONTD...

- Consider the problem of moving a character code from the keyboard to the processor.
- Striking a key stores the corresponding character code in an 8-bit buffer register associated with the keyboard.
- Let us call this register DATAIN, to inform the processor that a valid character is in DATAIN, a status control flag, SIN, is set to 1.
- A program monitors SIN, and when SIN is set to 1, the processor reads the contents of DATAIN.
- When the character is transferred to the processor, SIN is automatically cleared to 0.
- If a second character is entered at the keyboard, SIN is again set to 1 and the process repeats.

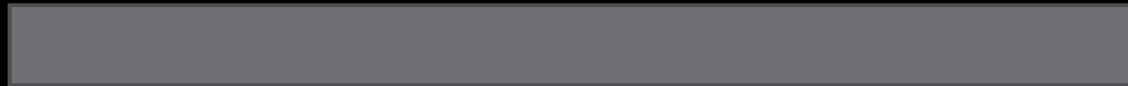
CONTD...

- An analogous process take place when characters are transferred from the processor to the display.
- A buffer register, DATAOUT, and a status control flag, SOUT, are used for this transfer.
- When SOUT equals 1, the display is ready to receive a character.
- Under program control, the processor monitors SOUT, and when SOUT is set to 1, the processor transfers a character code to DATAOUT.
- The transfer of character code to DATAOUT clears SOUT to 0.
- When the display device is ready to receive a second character, SOUT is again set to 1.

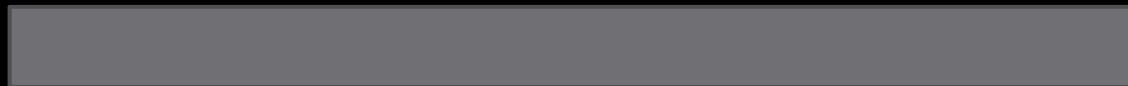
EXAMPLE

To review the basic concepts, let us consider a simple example of I/O operations involving a keyboard and a display device in a computer system. The four registers are used in data transfer operation. Register STATUS contains two control flags, SIN and SOUT, which provide status information for the keyboard and the display unit, respectively. The four flags KIRQ & DIRQ and KEN & DEN in the status and control registers are used in case of Interrupts. Data from the keyboard are made available in the DATAIN register and the data sent to the display are stored in the DATAOUT register.

DATAIN



DATAOUT



STATUS



CONTROL



7 6 5 4 3 2 1 0

PROGRAM

	Move	#LINE, R0	Initialize the memory pointer
WAITK	TestBit	#0, STATUS	Test SIN Flag
	Branch=0	WAITK	Wait for character to be entered
	Move	DATAIN, R1	Read character
WAITD	TestBit	#1, STATUS	Test SOUT Flag
	Branch=0	WAITD	Wait for display to become ready
	Move	R1, DATAOUT	Send character to display
	Move	R1, (R0)+	Store character and advance pointer
	Compare	#\$0D, R1	Check if carriage return
	Branch != 0	WAITK	If not, get another character
	Move	#\$0A, DATAOUT	Otherwise send line feed
	Call	PROCESS	Call a subroutine to process the input line

Note: Here #0 means the value at location 0 in status register. Similar the case for #1

CONTD...

- The program reads a line of characters from the keyboard and stores it in a memory buffer starting a location LINE. Then it calls a subroutine PROCESS to process the input line. As each character is read, it is echoed to the display.
- Register R0 is used as a pointer to memory buffer area. The contents of R0 are updated using the auto increment addressing mode so that successive characters are stored in successive memory locations.
- Each character is checked to see if it is the carriage return character, which has the ASCII code 0D(hex). If it is, a line feed character (ASCII code 0A) is sent to move the cursor one line down on the display and subroutine PROCESS is called. Otherwise, the program loops back to wait for another character from the keyboard.
- This example illustrates the **programmed controlled I/O**, in which the processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input or output device.
- There are two other commonly used mechanism for implementing I/O operations: **interrupts** and **direct memory access**.

CONTD...

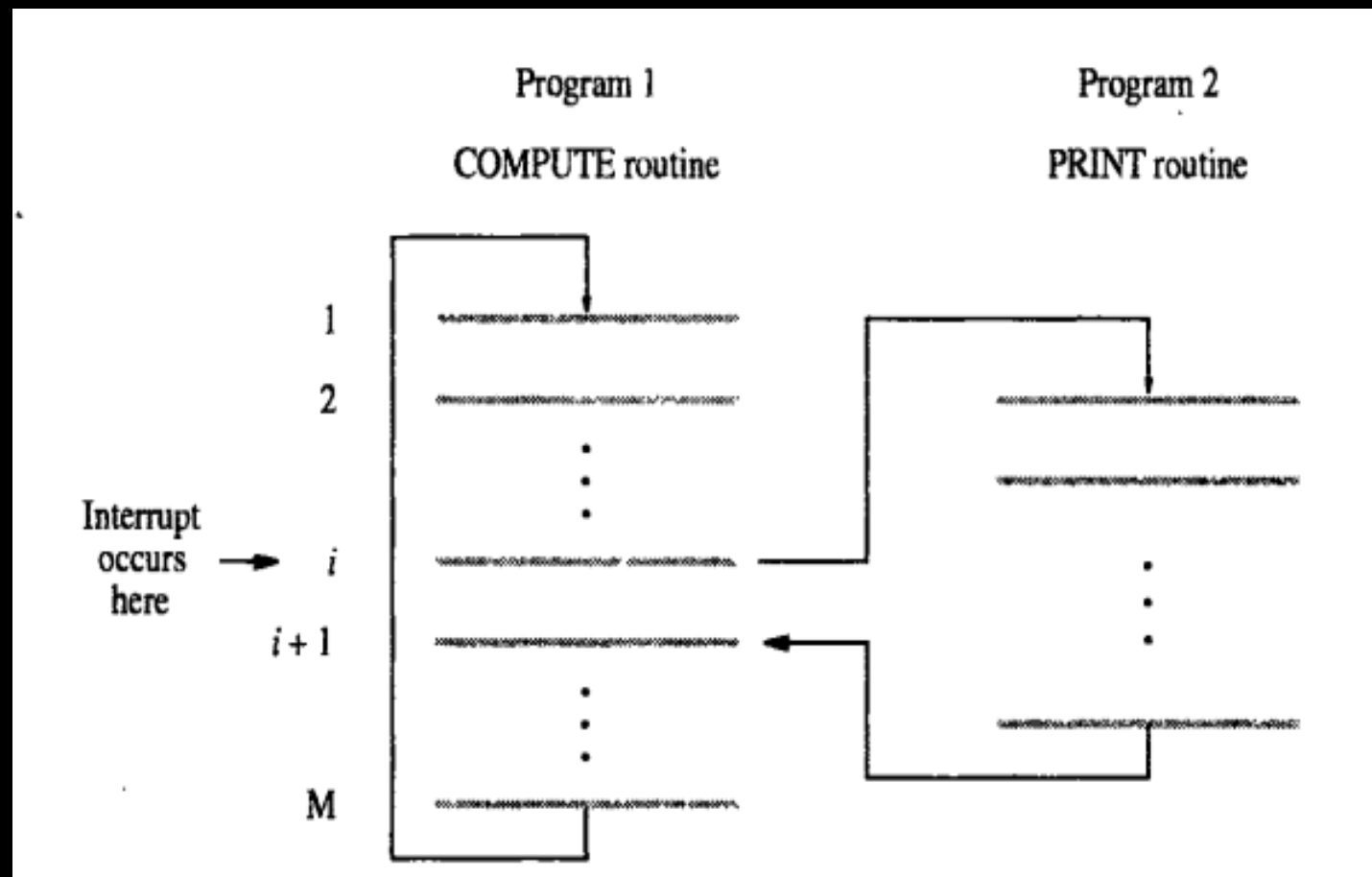
- In the case of interrupts, synchronization is achieved by having the I/O devices send a signal over the bus whenever it is ready for a data transfer operation.
- Direct memory access is a technique used for high speed I/O devices. It involves having the device interface transfer data directly to or from the memory without continuous involvement by the processor.

INTERRUPTS

- There are many situations where other tasks can be performed while waiting for an I/O device to become ready.
- To allow this to happen, we can arrange for the I/O device to alert the processor when it become ready.
- It can do so by sending a hardware signal called an interrupt to the processor.
- At least one of the bus control lines, called an interrupt request line, is usually dedicated for this purpose.
- Since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions.

EXAMPLE - FIGURE

This example illustrates the concept of interrupts. The routine executed in response to an interrupt request is called the interrupt service routine, which is the PRINT routine in our example.



CONTD...

- Assume that an interrupt request arrives during execution of instruction i in previous figure.
- The processor first complete the execution of instruction i . Then it loads the program counter with the address of the first instruction of the interrupt service routine i.e. PRINT in this case.
- After execution of the interrupt service routine (i.e. PRINT) , the processor has to come back to instruction $i+1$.
- Therefore, when an interrupt occurs, the current contents of the PC, which point to instruction $i+1$, must be put in temporary storage in a known location.
- A return from interrupt instruction at the end of the interrupt service routine reloads the PC from that temporary storage location, causing execution to resume at instruction $i+1$.

CONTD...

- As part of the handling the interrupts, the processor must inform the device that its request has been recognized so that it may remove its interrupt request signal.
- This may be accomplished by means of a special control signals on the bus, an interrupt acknowledge signal, used in some of the interrupt schemes.
- Before starting execution of the interrupt-service routine, any information that may be altered during the execution of the that interrupt service routine must be saved.
- This information must be restored before execution of the interrupted program is resumed.

CONTD...

- The task of saving and restoring information can be done automatically by the processor or by program instructions.
- The process of saving and restoring registers involves memory transfers that increase the total execution time, hence represent execution overhead.
- Saving registers also increase the delay between the time an interrupt request is received and the start of execution of the interrupt service routine. This delay is called the interrupt latency.
- In some applications, a long interrupt latency is unacceptable.
- For these reasons, the amount of information saved automatically by the processor when an interrupt request is accepted should be kept to a minimum.
- Typically, the processor saves only the content of the program counter and the processor status register.
- Any additional information that needs to be saved must be saved by program instructions at the beginning of the interrupt service routine and restored at the end of the routine.

ENABLING AND DISABLING INTERRUPTS

- The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program and start the execution of another.
- Because interrupts can arrive at any time, they may alter the sequence of events from that envisaged by the programmer.
- Hence the interruption of program execution must be carefully controlled.
- A fundamental facility found in all computers is the ability to enable and disable such interruptions as desired.
- There are many situations in which processor should ignore interrupt requests.
- For example, an interrupt request from the printer should be accepted only if there are output lines to be printed.
- After printing the last line of a set of n lines, interrupts should be disabled until another set becomes available for printing.
- For these reasons, some means for enabling and disabling the interrupts must be available for the programmer.

CONTD...

- A simple way is to provide machine instructions, such as Interrupt enable and Interrupt disable, that perform these functions.
- Let us consider in details, the specific case of a single interrupt from one device.
- When a device activates the interrupt request signal, it keeps this signal activated until it learns that the processor has accepted its request.
- This means that interrupt request signal will be active during execution of the interrupt service routine, perhaps until an instruction is reached that accesses the device in question.
- It is essential to ensure that this request signal does not lead to successive interruptions, causing the system to enter an infinite loop from which it cannot recover.
- One simple solution is to have the processor automatically disable interrupts before starting the execution of the interrupt service routine.
- After saving the contents of the PC and the processor status register (PS) on the stack, the processor performs the equivalent of executing an interrupt disable instruction..

CONTD...

- It is often the case that one bit in the PS register called interrupt enable, indicates whether interrupts are enabled.
- An interrupt request received while this bit is equal to 1 will be accepted.
- After saving the contents of the PS on the stack, with the interrupt enable bit equal to 1, the processor clears the interrupt enable bit in its PS register, thus disabling further interrupts.
- When a return from instruction is executed, the contents of the PS are restored from the stack, setting the interrupt enable bit equal back to 1.
- Hence, interrupts are again enabled.

CONTD...

Let us summarize the sequence of events involved in handling an interrupt request from a single device. Assuming that interrupts are enabled, the following is a typical scenario:

1. The device raises an interrupt request.
2. The processor interrupts the program currently being executed.
3. Interrupts are disabled by changing the control bits in the PS register.
4. The device is informed that its request has been recognized, and in response, it deactivates the interrupt request signal.
5. The action requested by the interrupt is performed by the interrupt service routine.
6. Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES

- Let us now consider the situation where a number of devices capable of initiating interrupts are connected to the processor. Because these devices are operationally independent, there is no definite order in which they will generate interrupts. For example, device X may request an interrupt while an interrupt caused by device Y is being serviced, or several devices may request interrupts at exactly the same time. This gives rise to a number of questions:
 1. How can the processor recognize the device requesting an interrupt?
 2. Given that different devices are likely to require different interrupt service routines, how can the processor obtain the starting address of the appropriate routine in each case?
 3. Should a device be allowed to interrupt the processor while another interrupt is being serviced?
 4. How should two or more simultaneous interrupt requests be handled?

SOLUTION

- When a request is received over the common interrupt request line, additional information is needed to identify the particular device that activated the line.
- Furthermore, if two devices have activated the line at the same time, it must be possible to break the tie and select one of the request for the service. When the interrupt service routine for the selected device has been completed, the second request can be serviced.
- The information needed to determine whether a device is requesting an interrupt is available in its status register.
- When a device raises an interrupt request, it sets to 1, one of the bits in its status register, which we call the IRQ bit.
- For example bits KIRQ and DIRQ are the interrupt request bits for the keyboard and display respectively.

CONTD...

- The simplest way to identify the interrupting device is to have the interrupt service routine poll all the I/O device connected to the bus.
- The first device encountered with IRQ bit set is the device that should be serviced. An appropriate subroutine is called to provide the requested service.
- The polling scheme is easy to implement, Its main disadvantage is the time spent interrogating the IRQ bits of all the devices that may not be requesting any service.
- An alternative approach is to use vectored interrupts.

VECTORED INTERRUPTS

- To reduce the time involving in the polling process, a device requesting an interrupt may identify itself directly to the processor.
- Then, the processors can immediately start executing the corresponding interrupt- service routine.
- A device requesting an interrupt can identify itself by sending a special code to processor over a bus.
- This enables the processor to identify individual devices even if they share a single interrupt request line.
- The code supplied by the device may represent the starting address of the interrupt-service routine for that device.
- The code length is typically in the range of 4 to 8 bits.
- The remainder of the address is supplied by the processor based on the area in its memory where the addresses for interrupt service routines are located.

CONTD...

- This arrangement implies that the interrupt service for a given device must always start at the same location.
- The location pointed to by the interrupting device is used to store the starting address of the interrupt service routine.
- The processor reads this address, called the interrupt vector, and loads it into the PC.
- In most computers, I/O devices send the interrupt vector code over the data bus, using the bus control signals to ensure that devices do not interfere with each other.
- When a device sends an interrupt request, the processor may not be ready to receive the interrupt vector code immediately.

CONTD...

- For example, it must first complete the execution of the current instruction, which may require the use of the bus.
- There may be further delays, if interrupt happen to be disabled at the time the request is raised.
- The interrupting device must wait to put the data on the bus only when the processor is ready to receive it.
- When the processor is ready to receive the interrupt vector code, it activates the interrupt-acknowledge line, INTA.
- The I/O device responds by sending its interrupt vector code and turning off the INTR signal.

INTERRUPT NESTING

- The interrupts should be disabled during the execution of an interrupt service routine, to ensure that a request from one device will not cause more than one interruption.
- The same arrangement is often used when several devices are involved, in which case execution of a given interrupt-service routine, once started, always continues to completion before the processor accepts an interrupt request from a second device.
- Interrupt service routines are typically short, and the delay they may cause is acceptable for most simple devices.
- For some devices however a long delay in responding to an interrupt request may lead to erroneous operation.
- So sometimes it is necessary that I/O devices should be organized in a priority structure.
- An interrupt request from a higher priority device should be accepted while the processor is servicing another request from a lower priority device.

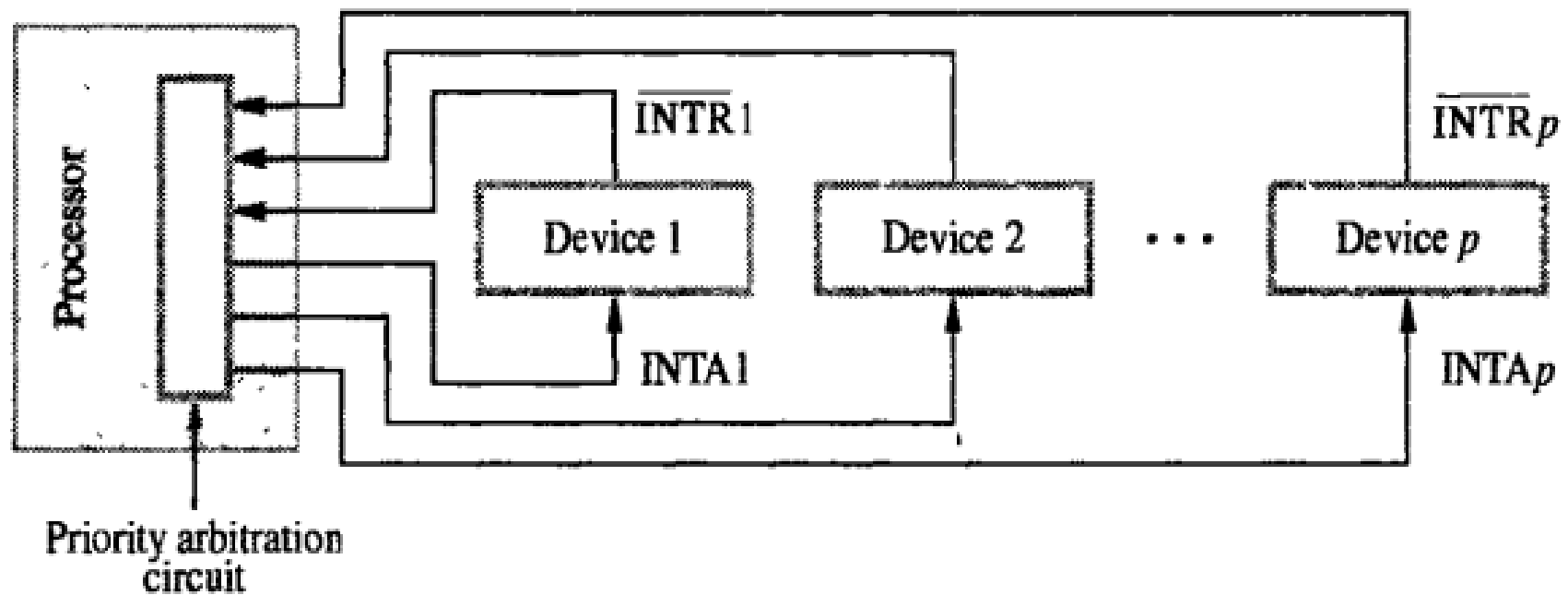
CONTD...

- A multiple level priority organization means that during execution of an interrupt-service routine, interrupt requests will be accepted from some devices but not from others, depending upon the device's priority.
- To implement this scheme, we can assign a priority level to the processor that can be changed under program control.
- The processor accepts interrupts only from devices that have priorities higher than its own.
- At the time the execution of an interrupt service routine for some devices is started, the priority of the processor is raised to that of the device.
- This action disables interrupts from devices at the same level of priority or lower.
- However, interrupt request from higher priority devices will continue to be accepted.
- The processor's priority is usually encoded in a few bits of the processor status word. It can be changed by program instructions that write into the PS.
- These are privileged instructions, which can only be executed while the processor is running in the supervisor mode.

CONTD...

- A multiple priority can be implemented easily by using separate interrupt request and interrupt acknowledgment line for each device.
- Each of the interrupt request lines is assigned a different priority level.
- Interrupt request over these lines are sent to a priority arbitration circuit in the processor.
- A request is accepted only if it has higher priority level than that currently assigned to processor.

FIGURE

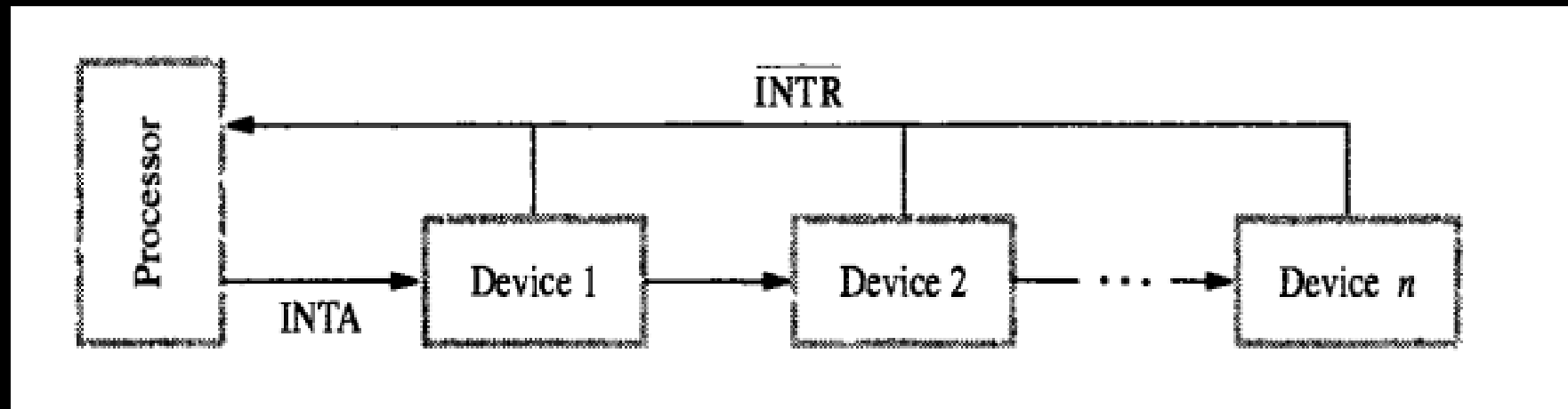


Implementation of interrupt priority using individual interrupt request and acknowledgement lines.

SIMULTANEOUS REQUESTS

- Let us now consider the problem of simultaneous arrivals of interrupt request from two or more devices.
- The processor must have some means of deciding which request to service first.
- A widely used scheme is to connect the devices to form a daisy chain as shown in figure.
- The interrupt request line $\overline{\text{INTR}}$ is common to all devices.
- The interrupt acknowledgement line, INTA , is connected in a daisy chain fashion, such that the INTA signal propagates serially through the devices.
- When several devices raise an interrupt request and the $\overline{\text{INTR}}$ line is activated, the processor responds by setting the INTA line to 1.
- This signal is received by device 1. Device 1 passes the signal to device 2 only if it does not require any service.
- If device 1 has a pending request for interrupt, it blocks the INTA signal and proceeds to put its identifying code on the data lines.
- Therefore, in the daisy chain arrangement, the device that is electronically closest to the processor has the highest priority. The second device along the chain has second highest priority, and so on.

FIGURE

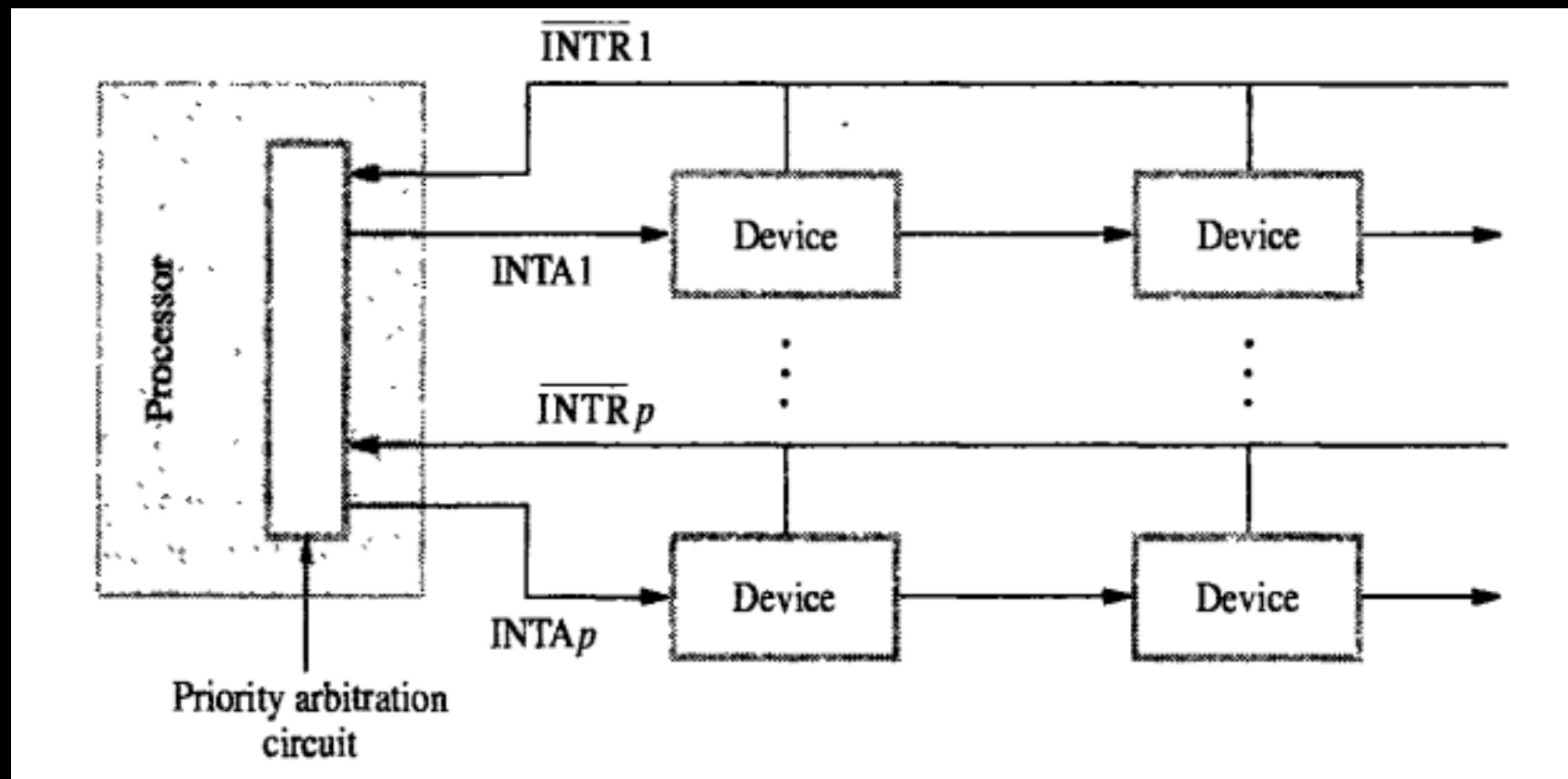


Daisy Chain



- Devices may also be organized into groups, and each group is connected at a different priority level.
- Within a group, devices are connected in a daisy chain. This organization is used in many computer systems.

FIGURE



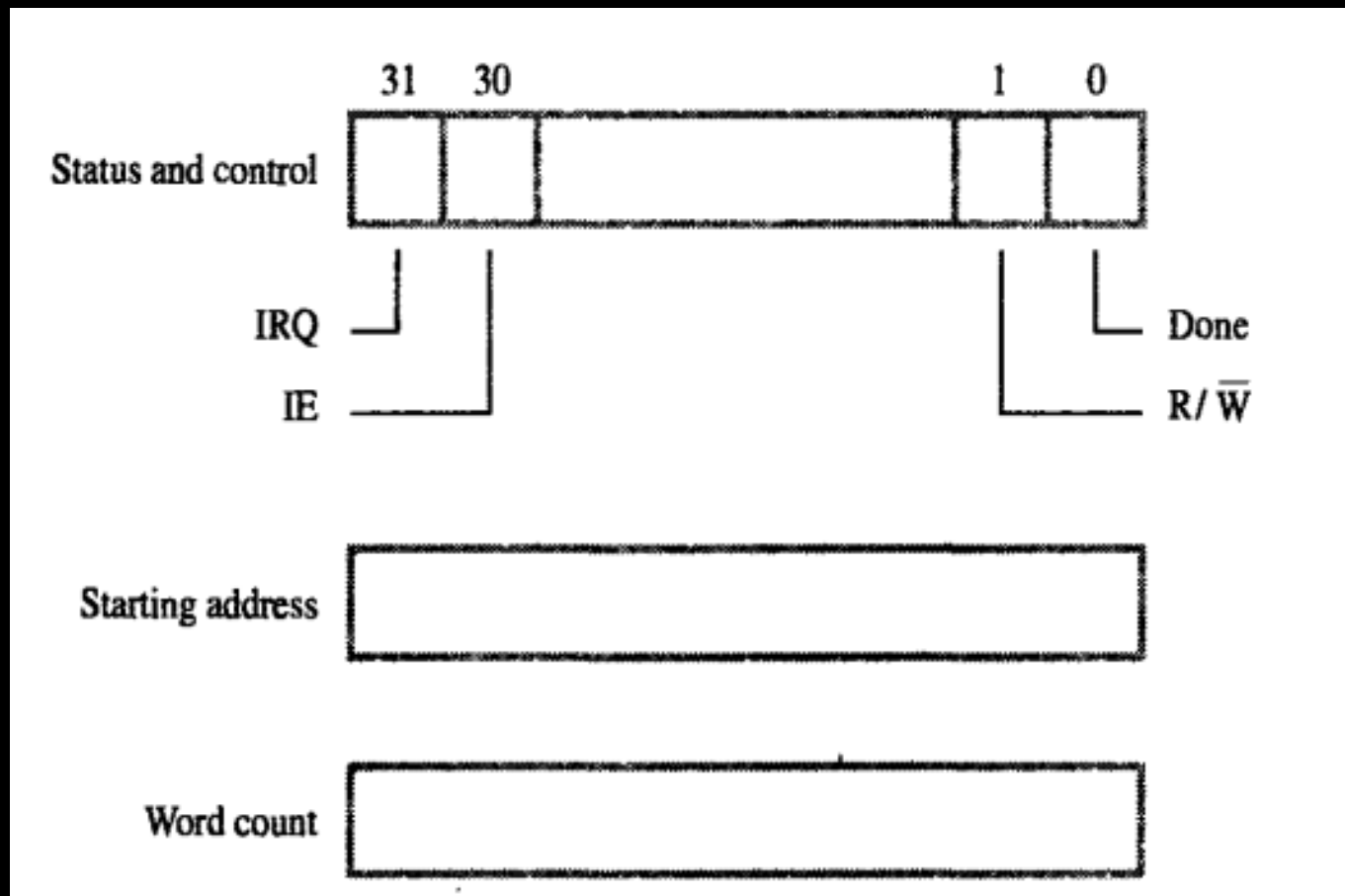
CONTROLLING DEVICE REQUESTS

- The control needed is usually provided in the form of an interrupt enable bit in the device's interface circuit.
- The keyboard interrupt enable, KEN and display interrupt enable, DEN, flags in register CONTROL performs this function.
- If either of these flags is set, the interface circuit generates an interrupt request whenever the corresponding status flag in register STATUS is set.
- At the same time, the interface circuit sets bit KIRQ or DIRQ to indicate that the keyboard or display unit, respectively, is requesting an interrupt.
- If an interrupt enable bit is equal to 0, the interface circuit will not generate an interrupt request, regardless of the state of the status flag.
- In summary, there are two independent mechanisms for controlling interrupt requests. At the device end, an interrupt enable bit in a control register determines whether the device is allowed to generate an interrupt request.
- At the processor end, either an interrupt enable bit in the PS register or a priority structure determines whether a given interrupt request will be accepted.

DIRECT MEMORY ACCESS

- Data are transferred by executing the instructions such as: *Move DATAIN, R0*
- An instruction to transfer input or output data is executed only after the processor determines that the I/O device is ready.
- To do this, the processor either polls a status flag in the device interface or waits for the device to send an interrupt request.
- In either case, considerable overhead is incurred, because several program instructions must be executed for each data word transferred.
- In addition to polling the status register of the device, instructions are needed for incrementing the memory address and keeping track of the word count.
- When interrupts are used, there is the additional overhead associated with saving and restoring the program counter and other state information.
- To transfer large blocks of data at high speed, an alternative approach is used.
- 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.

FIGURE



Registers in DMA Interface

CONTD...

- DMA transfers are performed by a control that is part of the I/O device interface.
- We refer to this circuit as DMA controller.
- The DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory.
- For each word transferred, it provides the memory address and all the bus signals that control data transfer.
- Since, it has to transfer blocks of data, the DMA controller must increment the memory address for successive words and keeps track of the number of transfers.
- Although a DMA controller can transfer data without intervention by the processor, its operation must be under the control of a program executed by the processor.
- To initiate the transfer of a block of words, the processor sends the starting address, the number of words in the block, and the direction of the transfer.
- On receiving this information, the DMA controller proceeds to perform the requested operation.
- When the entire block has been transferred, the controller informs the processor by raising an interrupt signal.

CONTD...

- Two registers are used for storing the starting address and the word count.
- The third register contains status and control flags.
- The R/\overline{W} bit determines the direction of the transfer. When this bit is set to 1 by a program instruction, the controller performs a read operation, that is, it transfers data from the memory to the I/O device. Otherwise it performs , a write operation.
- When the controller has completed transferring a block of data and is ready to receive another command, it sets the Done flag to 1.
- Bit 30 is the interrupt enable flag, IE. When this flag is set to 1, it causes the controller to raise an interrupt after it has completed transferring a block of data.
- Finally the controller sets the IRQ bit to 1 when it has requested an interrupt.

CONTD...

- Memory access by the processor and the DMA controller are interwoven.
- Requests by DMA devices for using the bus are always given higher priority than processor requests.
- Among different DMA devices, top priority is given to high speed peripherals such as disk, a high speed network interface, or a graphics display device.
- Most DMA controllers incorporate a data storage buffer.
- A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main memory.
- To resolve these conflicts, an arbitration procedure is implemented on the bus to coordinate the activities of all devices requesting memory transfers.

BUS ARBITRATION

- The device that is allowed to initiate data transfers on the bus at any given time is called the bus master.
- When the current master releases the control of the bus, another device can acquire this status.
- Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.

CONTD...

There are two approaches for bus arbitration:

1. Centralized arbitration
2. Distributed arbitration

CENTRALIZED ARBITRATION

- In centralized arbitration a single bus arbiter performs the required arbitration.
- The bus arbiter may be the processor or a separate unit connected to the bus.
- Normally the processor is the bus master unless it grants bus mastership to one of the DMA controllers.

CONTD...

- A DMA controller indicates that it needs to become the bus master by activating the bus request line, \overline{BR} .
- When the bus request is activated, the processor activates the bus grant signal, BG1, indicating to the DMA controller that they may use the bus when it become free.
- This signal is connected to all DMA controller using a daisy chain management.

CONTD...

- If DMA controller 1 is requesting the bus, it blocks the propagation of the grant signal to other devices. Otherwise it passes the grant downstream by asserting BG2.
- The current bus master indicates to all devices that it is using the bus by activating the Bus Busy signal, BBSY.

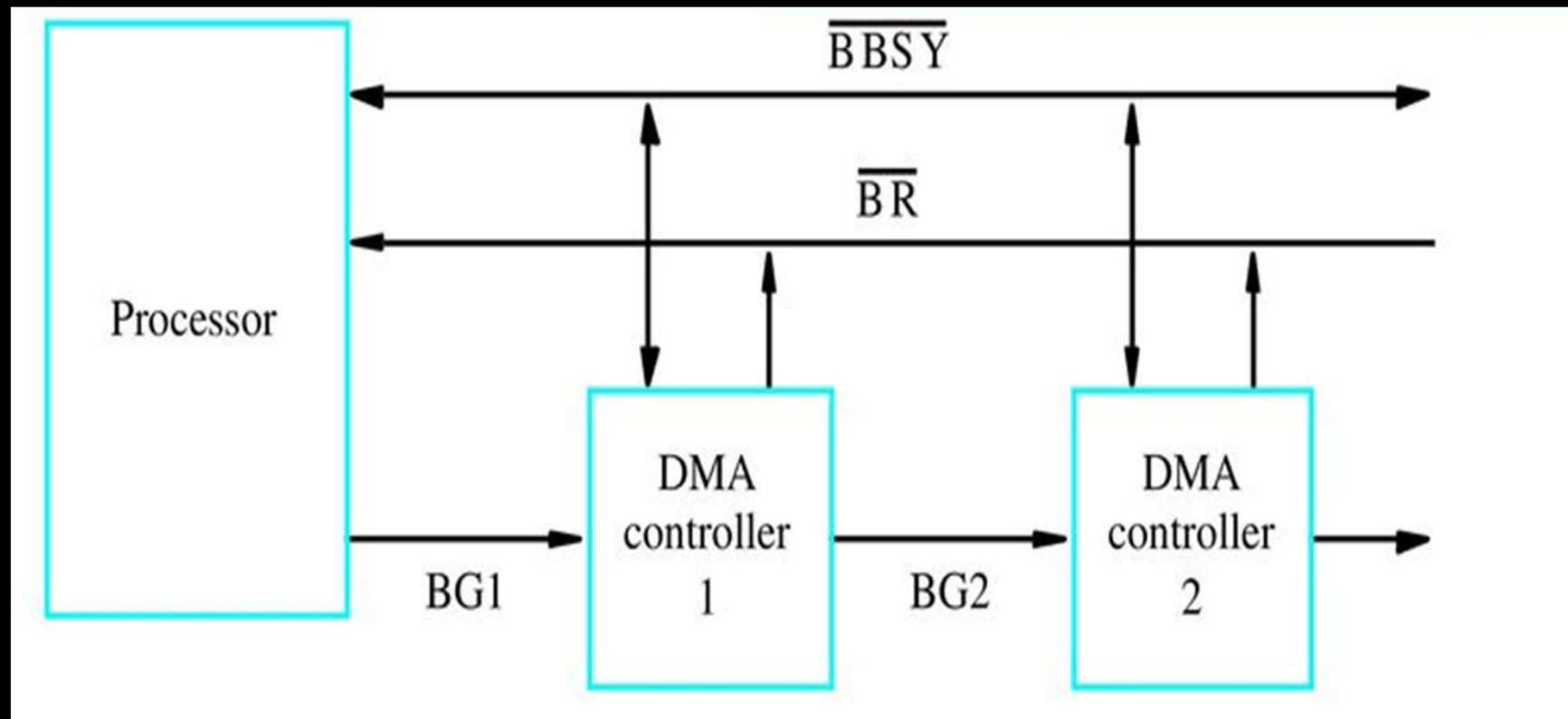
CONTD...

- After receiving the bus grant signal, a DMA controller waits for the Bus Busy signal to become inactive, then assumes the mastership of the bus.
- At this time, it activates the Bus Busy signal to prevent other devices from using the bus at the same time.

CONTD...

- The arbiter circuit ensures that only one request is granted at any given time, according to a predefined priority scheme.
- For example, if there are four bus request lines, BR1 to BR4, a fixed priority scheme may be used in which BR1 is given top priority and BR4 is given lowest priority.

CONTD...



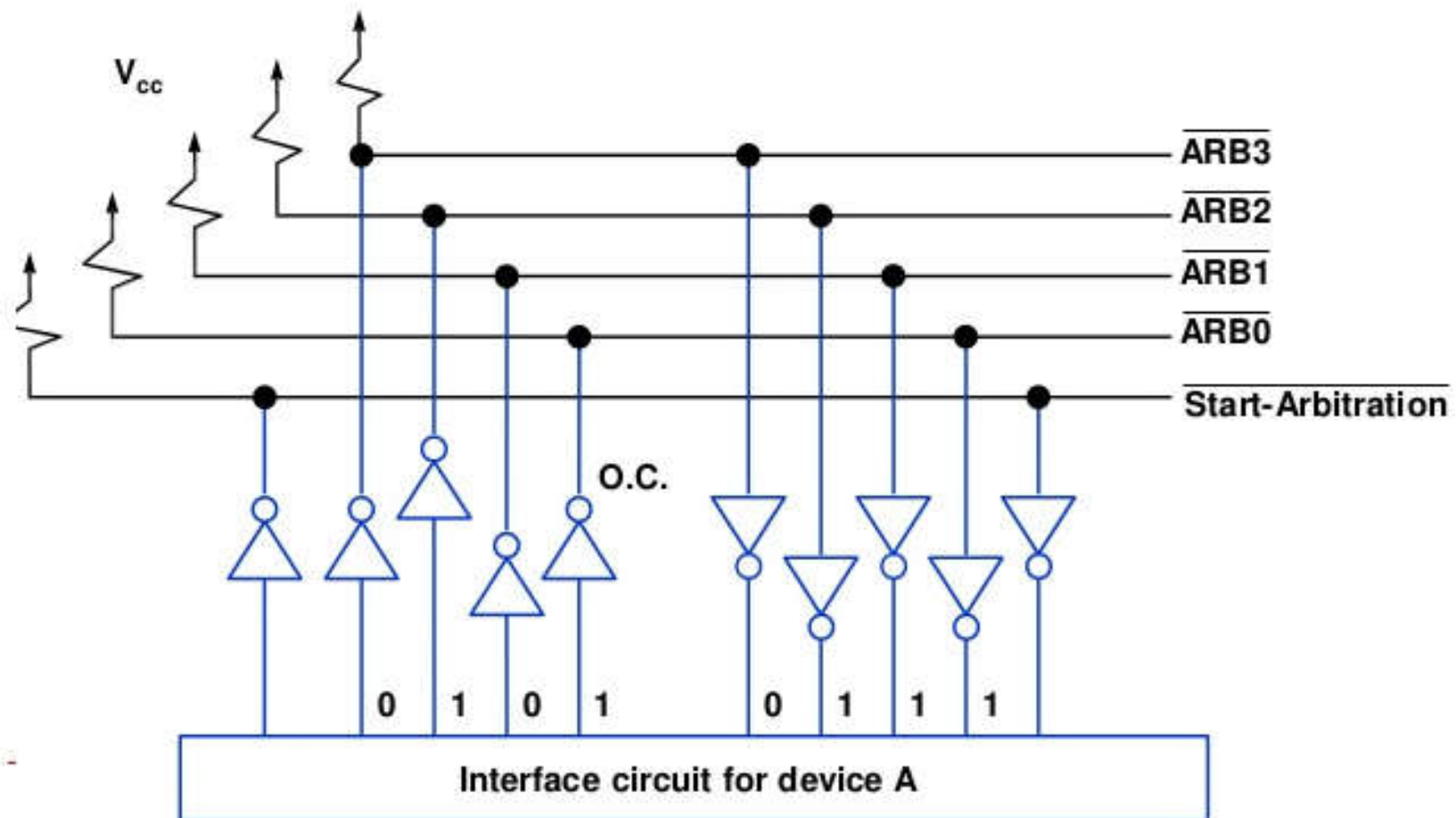
A simple arrangement for bus arbitration using a daisy chain

DISTRIBUTED ARBITRATION

- Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.
- Each device on the bus is assigned a 4-bit identification number.
- When one or more devices request the bus, they assert the Start Arbitration signal and place their 4-bit ID numbers on four open collector lines, ARB0 to ARB3.

CONTD...

- A winner is selected as a result of the interaction among the signal transmitted over these lines by all contenders.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.
- Decentralised arbitration has the advantage of offering the higher reliability, because the operation of the bus is not dependent on any single device.



A distributed arbitration scheme

QUESTIONS

In a vectored interrupt:

- a) The branch address is assigned to a fixed location in memory
- b) The interrupting source supplies the branch information to the processor through an interrupt vector
- c) The branch address is obtained from a register in the processor
- d) None of the above



For the daisy chain scheme of connecting I/O devices, which of the following statements is true?

- a) It gives non-uniform priority to various devices
- b) It gives uniform priority to all devices
- c) It is only useful for connecting slow devices to a processor
- d) It requires a separate interrupt pin on the processor for each devices



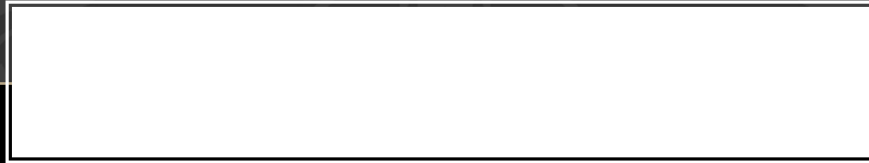
Which of the following device should get the highest priority in assigning interrupts?

- a) Hard Disk
- b) Printer
- c) Keyboard
- d) Floppy Disk



Which of the following is true?

- a) Unless enabled, a CPU will not be able to process interrupt
- b) Loop instructions cannot be interrupted till they complete
- c) A processor checks for interrupts before executing a new instruction
- d) Only level triggered interrupts are possible on microprocessors



A processor needs software interrupt to:

- a) Test the interrupt system of the processor
- b) Implement co-routines
- c) Obtain system services which need execution of privileged instructions
- d) Return from subroutine



Which one of the following is true for a CPU having a single interrupt request line and single interrupt grant line?

- a) Neither vectored interrupt nor multiple interrupting devices are possible
- b) Vectored interrupts are not possible but multiple interrupting devices are possible
- c) Vectored interrupts and multiple interrupting devices both are possible
- d) Vectored interrupt is possible but multiple interrupting devices are not possible



A CPU generally handles an interrupt by executing an interrupt service routine:

- a) As soon as an interrupt is raised
- b) By checking the interrupt register at the end of fetch cycle
- c) By checking the interrupt register after finishing the execution of the current instruction
- d) By checking the interrupt register at fixed time intervals



A device with data transfer rate 10 KB/sec is connected to a CPU. Data is transferred byte wise. Let the interrupt overhead be 4 microsecond. The byte transfer time between the device interface register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program controlled mode?



The size of the data count register of a DMA controller is 16bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to the main memory is_____.

REFERENCES

- Chapter 4, Computer Organization by Carl Hamacher, Fifth Edition

A decorative border at the top of the slide featuring a repeating floral and vine pattern in a dark gray color.

Thank You