Chapter 7

INPUT- OUTPUT ORGANIZATION

Overview

- I/O Organization and Interface Units
- Isolated I/O and Memory-Mapped I/O
- Different Modes of Data Transfer between Memory and Peripherals
- Interrupts
- Vectored and Non-Vectored Interrupts
- Subroutines
- Direct Memory Access
- Input-Output Processors
- Asynchronous Data Transfer

7.1 Input-Output Organization

The Input-Output (I/O) organization of a computer provides the mode of communication between the computer and the outside world i.e. with the help of the I/O devices, users can interact with the computer. The I/O devices that are attached to the computer are also known as ***peripherals***.

Some of the I.O devices that are commonly used in a computer are:

Input Devices: Keyboard, Mouse, Joystick, Scanner etc.

Output Devices: Printer, Monitor etc.

7.2 Input-Output Interface Units

I/O interface units are special hardware components that lie between the I/O devices and the processor bus. These devices serving as special communication links between the CPU and the peripherals, actually synchronize and supervise i.e. control all input and output transfers.

Need of Interface Units:

Certain differences between the characteristics (functional and behavioral) of the peripherals and the CPU (and also memory) exist. Interface units are needed to resolve the differences, which are:

(a) Operations of the peripherals, which are either electromagnetic or electromechanical devices, are different from that of the CPU and memory, which are generally electronic devices. So a signal conversion between the two is required and is done by the interface unit between them.

(b) The interface unit has to synchronize the data transfer mechanism between the slow peripherals and the fast CPU.

(c) Interface units handle the differences between the data codes and formats in peripherals and the CPU or memory word formats.

Working Mechanism of Interface Units:

Each of the I/O devices has their own interface unit. The I/O devices have got their own identifying addresses also, with which the CPU identifies an I/O device (i.e. an I/O device's address is actually the address of the registers storing the I/O information). In order to communicate with an I/O device, the CPU places the address of that I/O device in the address bus and places the appropriate control command (i.e. read or write) in the control bus (I/O bus also consists of address, data and control lines like memory bus). The I/O bus from the processor is attached to all peripherals via the respective peripheral interfaces. The address from the processor is received by all the interfaces. Each of them

decodes the address (each interface has its own address decoders). Only that particular interface which detects its own address (i.e. address of its own I/O device) activates the path between the bus lines and the device that it controls i.e. only the particular I/O device for which the address is meant for is activated and acts accordingly (for example, if there is a write command from CPU, the peripheral receives the data from the data line and the write command from the control line). All other I/O devices, whose address does not match with the address provided by the CPU, are not activated by their respective interfaces.

Diagram:

Figure 1, shows the connection between the processor and the I/O devices with the interface units lying in-between.

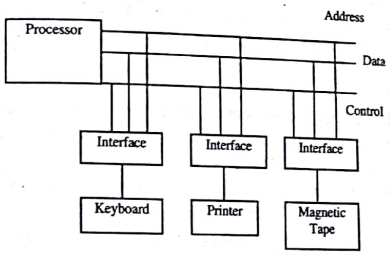

Fig: 1 Connection of processor to I/O devices via the I/O bus and respective interface units

7.3 Isolated I/O and Memory-Mapped I/O

In a computer system, CPU communicates with the memory unit as well as with the I/O devices via the buses. Generally there are three ways that computer buses can be used to serve as a communicating link between the CPU, memory and the I/O devices. These are:

(a) CPU communicates with the memory and the I/O devices via two separate buses, the memory bus (for the memory units) and I/O bus (for the I/O devices).

(b) CPU communicates with the memory and I/O units via one common bus (i.e. common address and data lines) but uses two different pairs of control lines (one for memory and another for I/O).

(c) CPU uses a completely common bus (with all control lines) for both memory and I/O units.

Isolated I/O or I/O mapped I/O:

In this mechanism, CPU communicates with the memory and I/O units via one common bus (with same address and data lines) having two different pairs of control lines (separate read and write lines for memory and I/O). This differentiates between memory and I/O transfers.

While communicating, the CPU specifies whether the address on the address lines is meant for memory or for an I/O device by enabling either the memory control lines (memory read or write line) or the I/O control lines (I/O read or write line). So during a memory transfer, either of the memory read or memory write lines are enabled while during an I/O transfer, either of the I/O read or I/O write lines are enabled.

This configuration isolates all I/O device addresses from the memory addresses i.e. in this mechanism, CPU treats the I/O addresses separately from the memory addresses. Each has its own separate address spaces. Hence this mechanism is known as isolated I/O method (also known as I/O mapped I/O method).

Memory-Mapped I/O:

Unlike the isolated I/O method where memory and I/O addresses are assigned separately in their respective address spaces, in the memory-mapped I/O method, same address space is used for both memory and I/O addresses.

In such configuration, CPU communicates with the memory and I/O devices via the same bus having completely common lines. Here there is no distinction between memory and I/O addresses and the assigned address spaces for the I/O registers are assumed to be a part of the memory system (the I/O addresses are treated as part of the memory system). Though the assigned I/O addresses are kept in the memory itself, however they cannot be used to store any memory words. Hence the total storage capacity of the memory is reduced.

Differences between Isolated I/O and Memory- Mapped I/O:

Differences between Isolated I/O and Memory-Mapped I/O methods:

	Isolated I/O	Memory-Mapped I/O
1.	Though common address and data lines but distinct pair of control lines (in the bus) are there for memory and I/O communications with the CPU.	Completely common address, data and control lines (in the bus) for both memory and I/O communications with the CPU.
2.	CPU uses distinct instructions for handling memory and I/O devices.	No separate instructions for I/O units. CPU uses the same instructions for handling memory words and I/O data.
3.	Memory and I/O address spaces are different.	Memory and I/O address spaces are the same. I/O addresses reside in the memory unit.
4.	Memory address range or space available is more.	Due to mapping of I/O addresses in the memory. The available memory address range or space is reduced.
5.	Number of instructions needed in such computers is more, as separate memory and I/O instructions are needed.	Number of instructions needed in such computers is less, as same instruction can be used for handling of memory and I/O data.

7.4 Different Modes of Data Transfer between the Processor and the Peripherals

There are three possible modes of data transfer between the processor and the I/O devices. These are:

(a) Programmed I/O:

Such transfers are initiated as a result of I/O instructions written in the computer program. Once the transfer is initiated, the CPU has to constantly monitor the interface units of the I/O devices to see when the transfer can be actually made i.e. when the device will be ready to transfer data.

This constant monitoring requires lot of CPU time. In these time, the CPU remains idle as it cannot do any other work but to check whether the I/O device is ready to transfer data or not. Lot of CPU time is wasted in this mechanism.

(b) Interrupt I/O or Interrupt-Initiated I/O:

In this mechanism, the CPU need not constantly monitor to see when the I/O device will be ready to transfer data. Here the I/O device itself will let the CPU know that it is ready to transfer data and that it wants service from the CPU, meanwhile the CPU can continue to execute other programs. When the device gets ready to transfer data, it will send a signal (interrupt) to the CPU to provide service. CPU will then stop other tasks and will branch to the service program of that I/O device to process the I/O data transfer. When finished, CPU will return to the task it was originally performing and continue with it. Here unlike the previous method, CPU cycle is not wasted, as the CPU need not remain idle.

(c) Direct memory Access (DMA):

In programmed I/O data transfer occur between CPU and the peripherals. However in DMA, data transfer occurs between I/O devices and the memory unit without direct intervention by the CPU. CPU only initiates the transfer by supplying the starting address

of the memory location from where data is to be transferred and the number of words (bytes) to be transferred.

Comparison of the relative advantages and disadvantages of the three modes:

Relative advantages and disadvantages of the three modes:

Programmed I/O	Interrupt-initiated I/O	DMA
Slowest.	Medium.	Fastest.
Least expensive.	Medium.	Most expensive.
Simple to design.	Slightly complicated.	Highly complicated.
Wastage of CPU cycle degrades the performance of the computer.	CPU cycle is not wasted.	CPU directly does not play any role in such transfer.

7.5 Interrupt

Interrupt is a signal from an I/O device to let the CPU know that it is ready to transfer data and that it is requesting service from the CPU.

As soon as CPU receives this signal, it leaves its current unfinished task as it is and branches to the interrupting device's interrupt service routine and executes it to process the transfer. When finished, CPU again comes back to its unfinished task and continues with it.

Explanation of the Interrupt Handling Mechanism:

The entire ***interrupt handling mechanism*** is as follows:

(i) Suppose there are some data on the output register (OUTR) of a printer and they are to get printed.

(ii) If there are data on the OUTR, it means that the output flag (FGO or flag output) will be enabled. This will let the device's interface (printer's interface unit) unit to know that something is to be printed i.e. data transfer is needed and thus CPU service is required.

[Also if there are some data to be fed to the computer via the keyboard, suppose, then the data gets stored in the input register (INPR) and the flag input (FGI) will be enabled]

(iii) Then the printer's interface unit will send an interrupt request (a signal) to the CPU (via a control line known as the ***interrupt-request line*** in the control bus) to request the CPU to provide service, as the printer is ready to transfer data.

(iv) Meanwhile the CPU may be busy with another program execution. On getting the interrupt signal from the printer, CPU will stop the execution of the current program, save the return address of the next instruction in the current program in a stack and will branch to the interrupt service routine of the printer to handle the interrupt.

(v) CPU will thus execute the interrupt service routine, which is a program stored in the memory and thus will process the printer's data transfer request.

(vi) When finished, CPU will come back to execute its previous unfinished program and will continue to execute from the next instruction onwards. Address of the next instruction to be executed was already in store of the stack. Thus CPU will continue with the execution of its current program as if nothing has happened.

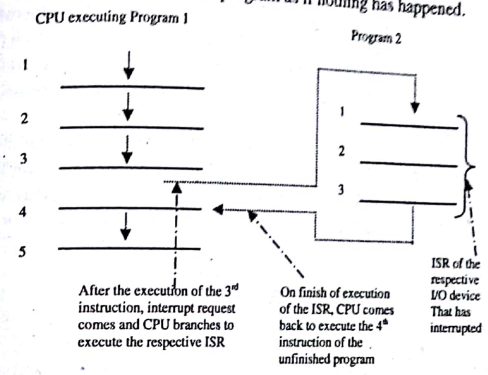

Fig: 2 How control is transferred when there is an interrupt request

[***Note:*** Both the programs in the figure 2 are stored in the memory but in different address locations i.e. all ISRs of the different I/O devices are stored in the memory in different address locations. While executing a particular ISR, the CPU just branches to that particular starting address of the ISR and executes them.]

Interrupt Service Routine:

An ***interrupt service routine (ISR)*** is actually a program stored in the memory of the computer. In order to handle or process an interrupt request from an I/O device, CPU has to execute the ISR (program) of that particular I/O device. Each of the I/O devices has its own interrupt service routine.

Different types of Interrupts:

There are generally three types of interrupts, which are:

(a) External Interrupts:

Such type of interrupts may come from any external sources like, from ***I/O devices*** when they are ready to transfer data, from a ***timing device*** to signify that the time of an event is over, it may occur due to some ***power failures*** etc.

(b) Internal Interrupts:

Such type of interrupts, called traps, may occur due to some illegal or erroneous conditions in the program (i.e. illegal or erroneous use of instructions or data in the program). Whenever there are internal errors, like ***register overflow, stack overflow, attempt to divide by zero*** etc. in the program, they give rise to internal interrupts.

(c) Software Interrupts:

Such type of interrupts may be incorporated or embedded in the program as an instruction by a programmer and are thus initiated by executing that instruction (interrupt instruction). So if the programmer want to initiate any sort of interrupt procedure at any desired point in the program, he may write an interrupt instruction at that point in the program.

Example of software interrupt is: ***INT 32*** (say). On execution of this interrupt instruction, control branches to the ISR of the number 32 interrupt (i.e. 32 number interrupt line).

[***Note:*** Number of lines in the CPU chip, allocated for external interrupts may vary from machine to machine. As for example in 8086, 0-255 lines are there, in 8085, RST lines, ISR lines etc. are there]

Differences between external and internal interrupt:

Soln. Differences are:

	External Interrupt	Internal Interrupt
1.	Initiated by an external event.	Initiated by some exceptional conditions caused by the program itself.
2.	Asynchronous in nature.	Synchronous in nature.
3.	As external interrupts are dependent on the external conditions and are independent of the program being executed at any time, hence on rerunning the program there is very less chance that interrupt will occur in the same place each time.	On rerunning the program, internal interrupt will occur in the same place each time.

Similarities between external and internal interrupt:

Both kinds of interrupts are initiated from signals occurring in the CPU hardware.

7.6 Vectored and Non-Vectored Interrupts

When an interrupt request comes from an I/O device, CPU stores the address of the next instruction to be executed in the current program from the program counter to the stack and branches to the ISR that processes the required I/O transfer of that particular device. There are generally two ways by which the processor chooses the branch address of the ISR.

They are:

(a) Non-Vectored Interrupt:

In this method, the branch address (i.e. address of the ISR) is always assigned to a fixed location in the memory and the processor always branches to that particular location.

(b) Vectored Interrupt:

In this method, the branch address (i.e. address of the ISR) is supplied by the interrupting I/O device itself and the processor branches accordingly.

7.7 Subroutine

A ***subroutine*** is actually a self-contained sequence of instructions that performs a given computational task each time it is called at various points in the main program. When a subroutine is called, the control branches to the beginning of the subroutine to start executing its set of instructions and on finishing, control branches back to the main program and start its execution again from the next instruction onwards.

Differences between subroutine and interrupt:

An interrupt, though, is very much similar to a subroutine, however there are certain differences between the two. These are:

	Subroutine	Interrupt
1.	A subroutine is called on execution of an instruction.	Interrupt is generally initiated as a result of an external or internal generated request (signal).
2.	Branch address of the subroutine is generally given in the address field of the instruction.	Branch address of the ISR is determined by the interrupt hardware.
3.	When the control branches to the subroutine from the main program, only the return address of the control in the main program is stored i.e. only the PC content is stored.	In this case, all information like, the content of program counter, content of all processor registers, content of some specific status conditions etc, necessary to define the state of the CPU when the control branches back to the main program, are stored.

7.8 Direct Memory Access

Direct Memory Access or DMA is the data transfer technique directly between the fast peripheral devices (line magnetic disk) and the memory unit through the memory bus without the direct intervention of the CPU.

Need of Direct Memory Access:

DMA is needed to transfer of block of data between the fast magnetic disk and memory. If the transfer is done via the CPU (speed of magnetic disk almost same as of CPU) then time required to transfer the entire block would have been much more. So, when data transfer must be done in blocks and when communicating devices are very fast then DMA is preferable. Generally in the computer, DMA mode of data transfer occurs.

DMA Controller:

The figure 3 below show a DMA controller:

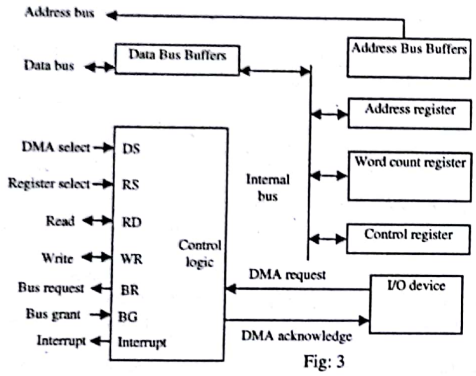

Fig: 3

The DMA controller communicates with the CPU via the bus. With the help of the DMA controller, I/O devices directly communicate with the memory unit. CPU only initializes the transfer by providing the following information:

(a) The starting address of the memory location from where the data is to be read or in which the data is to be stored.
(b) Word count i.e. number of memory words to be transferred.
(c) Specifies the mode of transfer, whether read or write.
(d) A control signal to start the DMA transfers.

Different Registers in a DMA Controller and their Functions:

DMA controller has got the following ***registers***:

Address Register: This register holds the starting address of the memory location (supplied by the CPU) from where the information is to be read or in which the information is to be stored.

Word Count Register: Holds the number of words to be transferred between the memory and I/O device. As soon as one word is transferred, the register counts decrements by one until all the words are transferred.

Control Register: This register specifies the mode of transfer i.e. holds a read or write as per supplied by the CPU. If read, then information will be transferred from memory to I/O device directly under the control of DMA. If write, then information will be transferred from I/O device to memory under the control of DMA.

Address Bus Buffers: The address bits (i.e. the starting memory address) go to the address bus (and then to the memory) via the address bus buffers from the address register.

Different Lines in a DMA Controller and their Functions:

The control logic has the following ***lines***:

DMA select (DS): To the CPU, the DMA controller is just like another I/O device. Multiple DMA controller may be present in a system each serving number of I/O devices. So, for a particular DMA controller to be activated when needed, CPU service is needed i.e. CPU must select it to make the controller active. Hence, CPU selects (activate) a DMA controller via the DMA select input.

Register select (RS): CPU selects the registers (to provide the starting address of the memory location, number of words to transfer and to specify the mode of transfer whether read or write) in the DMA controller by enabling the register select input.

Read (RD) and write (WR): These two lines are bidirectional. CPU communicates with the DMA registers through the data bus with the help of read and write lines. CPU writes the starting address of the memory location, number of words to be transferred in the DMA registers through the '***write***' line. Also CPU reads the status of the transfer through the '***read***' line.

Bus request (BR) and bus grant (BG): CPU generally serves as the bus master i.e. CPU controls the buses and other units act as bus slaves. But in case of DMA, where transfer occur directly between memory and I/O device under control of DMA without direct intervention of CPU, the control of buses goes to the DMA controller and not the CPU. So, when a DMA controller is ready for a DMA transfer, it requests the CPU to release the control of the bus through the '***bus request***' line by enabling it.
On getting the request from the DMA controller to release the bus, CPU enables the 'bus grant' line to let the controller know that the CPU has released the bus and the controller can take care of it i.e. the DMA controller can now act as the bus master and can carry on with the DMA transfer.

Different Mechanisms of DMA Transfer or Different types of DMA controller:
The different mechanisms of DMA transfer are:

(a) Burst Transfer:
In such a DMA transfer mechanism, the entire block of information (consisting of number of sequential memory words) is transferred at a time, continuously, until finished, when the DMA controller controls the bus. Such transfer is needed for fast devices like magnetic disk.

(b) Cycle Stealing:
In this mechanism, DMA controller transfers one word at a time and then returns the control of the bus to the CPU. Then again after a CPU cycle, the control comes back to the DMA controller, which again sends one word, and gives back the bus control to the CPU. This carries on until the entire block of data is transferred. So, DMA transfer virtually 'steals' one memory in between every CPU cycle.

7.9 Input-Output Processors

Input-Output Processors (IOP) are special type of processors that communicate directly with the I/O devices. These processors have direct memory access capabilities. Such processors free the CPU from managing I/O transfers and take care of the input output tasks.

Functions of IOPs:
IOPs generally handle all sorts of I/O transfers and have he capability of fetching and executing their own instructions without getting dependent on the CPU for initializing the transfer (unlike DMA controllers). Also IOPs can handle other processing tasks like arithmetic, logic information processing etc.

Block Diagram:

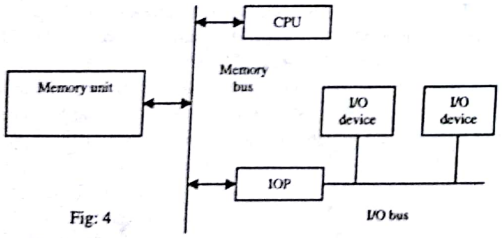

Fig: 4

As shown in the block diagram, the IOP is directly communicating with the I/O devices via the I/O bus. The memory unit can communicate with the IPO by means of direct memory access. CPU only initiates the I/O program and the IOPs then carry on with the data transfer between memory and I/O devices directly without any intervention from the CPU.

7.10 Asynchronous Data Transfer

When two units or devices in a digital system transfer data among themselves independently then such data transfer technique is known as asynchronous data transfer. In such technique the communicating units do not share a common clock and each has their own private clock providing them clock pulses independent of each other i.e. the internal timing in each is independent of each other. To indicate the time at which data is being transmitted, such data transfers requires the transmission of control signals between the communicating units.

Different Mechanisms of Asynchronous Data Transfer:
There are generally ***two mechanisms*** of asynchronous data transfer. They are:

(a) Strobe control method:
In this method, the sender unit supplies a strobe pulse to indicate the receiving unit about the potential time of the data transfer to take place.

(b) Handshaking method:
In this method, the presence of the data in the bus is indicated by a control signal that accompanies each data item transferred.

Related Questions & Answers

Question 1

Explain an interrupt cycle with a flowchart.

Answer:

Figure 5 shows the flowchart for an interrupt cycle.

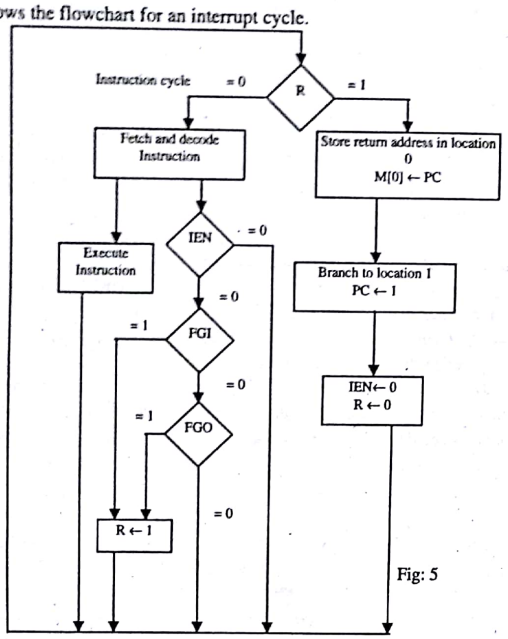

Fig: 5

In the figure 5 R is an interrupt flip-flop, which indicates whether an instruction cycle will occur or an interrupt cycle will occur. If R = 0, an instruction cycle occurs and if R = 1, control branches to execute an interrupt cycle.

IEN or ***Interrupt Enable*** is a flag that indicates whether the CPU is ready to accept any interrupt request or not. IEN = 0, means that the CPU is busy with another interrupt process and cannot take any further interrupt request until finished. On finishing,

IEN = 1, which indicates that the CPU is now ready to accept further interrupt requests from other I/O devices.

FGI or ***flag input*** is another flag, which indicates whether the input register (INPR) has any data to send to the CPU from the I/O device. If there are data to be transferred from the I/O device to the CPU then **FGI = 1** else FGI = 0.

Similarly, ***FGO*** or ***flag output*** is another flag, which indicates whether the output register (OUTR) has any data to send to the I/O device. If there are data to be transferred from the CPU to the I/O device then **FGO = 1** else FGO = 0.

Whenever **FGI or FGO = 1**, it means that the I/O device has some data to transfer either to the CPU or from the CPU and thus needs service from the CPU. Hence the I/O device will interrupt the CPU to get service. Now, if the IEN line if enabled i.e. only if the **IEN = 1**, the device can send interrupt request to the CPU.

So, if **IEN = 1** and **either of FGI or FGO = 1**, then **R = 1** and it indicates that an interrupt request has come and an interrupt cycle will occur now. In all other cases R = 0 and an instruction cycle occurs.

In case of an interrupt cycle, CPU branches to execute the initial instruction in the first address location of the ISR. It is needed to store the initial address of the ISR in the program counter as the address of the next instruction to be executed **(PC ← 1)**. Also it is required to store the address of the next instruction that the CPU was about to execute, of the unfinished program the CPU was executing when the interrupt request came. The return address was stored in the top of the stack from the PC **(M[0] ← PC)**. Until the present interrupt process an end, CPU is unable to take any more interrupt requests from other I/O devices and hence IEN remains 0 and thus R remains 0. Once this request is handled successfully, both IEN and R will again become 1, thus indicating that the CPU is ready to take further interrupt requests.

Question 2

What are the major functions of an I/O module?

Answer:

The major functions or requirements of the I/O unit or the I/O module are:

a. Communication with the central processing unit
b. Communication with an external device
c. Controlling and timing the different I/O transfers
d. Buffering of data
e. Detection of any errors that have occurred.

Question 3

In very simple words, describe the sequence of steps involved in the transfer of data between an external device and the CPU.

Answer:

Suppose the CPU wants to read data from a scanner. The steps involved are as follows:

a. The CPU sends message to the I/O module to check the status of the external device (i.e. whether the scanner is ready to transmit or not).
b. The I/O module communicates with the external device and sends a message back to the CPU letting it know the device status.
c. Provided the device is functional (i.e. ready to transmit), the CPU sends a command to the I/O module to request the device to transmit data (i.e. I/O module again sends a command to the device to transmit data).
d. The device on receiving this command from the I/O module, transfers data to the I/O module.
e. I/O module transfers the received data to the CPU.

Question 4

What are the different commands that an I/O module may receive from the CPU?

Answer:

There are four general commands that an I/O module may receive from the CPU. These are:

a. ***Test Command:*** CPU tests the various status conditions (whether on or off etc.) associated with the I/O module and the external devices.
b. ***Control Command:*** Different types of control commands from the CPU activate the external devices to perform a particular task as instructed.
c. ***Read Command:*** This command helps the CPU to read data from an external device via the I/O module. I/O module reads data as requested from the external device and places the data in the internal data bus for the CPU to read the data.
d. ***Write Command:*** I/O module writes or transmits the data, which is written by the CPU (on the data bus), to the external device.

Question 5

What do you mean by priority interrupt?

Answer:

Sometimes it may happen that multiple I/O devices send interrupt requests to the CPU simultaneously. In such condition, it is needed for the system to establish priority among the various sources to determine which request to serve first. This mechanism of establishing priority is called priority interrupt. Generally devices with high-speed of data transfer like magnetic disks etc. get the highest priority while the slowest devices like keyboard etc. get the lowest priority.

Question 6

What are the different modes of Interfacing I/O device with the processor? Compare their relative advantages & disadvantages? **[WBUT 2003]**

OR,

What are the various modes of data transfer between computer and peripherals? Explain. **[WBUT 2004]**

Answer: Refer to section 7.4.

Question 7

Explain the different mechanisms by which multiple interrupt requests are handled by the CPU. **[WBUT 2006]**

Answer:

When multiple interrupt requests arrive simultaneously, priority is established among them by means of software and also by means of hardware. The methods are:

(a) Polling:

This is a software mechanism of identifying the highest priority interrupt-source among the multiple interrupt devices. This method provides only one common branch address for all interrupts. The program which checks the interrupt lines polls the interrupt sources in sequence i.e. it checks out of the multiple interrupt lines of the multiple I/O devices, which interrupt signal is on. 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 the ISR of that particular source, else the second-priority source is tested and so on.

(b) Daisy-Chaining:

This is a hardware mechanism of identifying the highest priority interrupt-source among the multiple interrupt devices. In this method all I/O devices are connected serially to the CPU. The highest priority device is placed first in the chain while the lowest priority one is placed last. When multiple interrupt requests arrive to the CPU simultaneously, the CPU enables the interrupt acknowledge line (one of the many lines in the control bus)

and finds out which of the devices has interrupted. If the highest priority source (placed first in the chain) has not interrupted then the second one is tested and so on. The CPU, as per the priority maintained by the daisy-chaining method, serves the device that has interrupted.

Figure 6 shows the above mechanism.

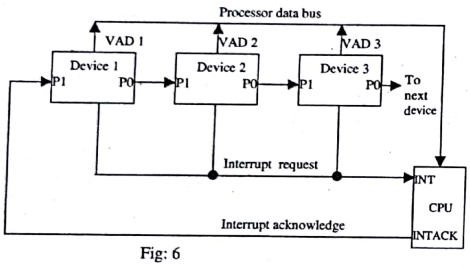

Fig: 6

Question 8

What are the different types of interrupt? Give examples. **[WBUT 2004]**

Answer: Refer to section 7.5.

Question 9

Explain the DMA transfer with the help of a block diagram.

Answer:

The DMA transfer occurs directly between the memory and the I/O devices under the control of the DMA controller. The transfer occurs as follows:

Steps:

(a) An I/O device when ready to transfer data to the memory, it sends a DMA request to the DMA controller it is attached to.

(b) On getting the request, the DMA controller enables the BR line to request the CPU to release the bus.

(c) Now two cases will arise:

(i) BG = 0 means buses are still not released by CPU and CPU will communicate with the DMA controller. CPU writes (with the help of the WR line of the DMA controller) into DMA address register the initial address of the memory location

from where data is to be transferred, CPU writes the number of words to be transferred into the DMA word count register, CPU specifies the mode of control whether 'read' or 'write'. Also CPU reads the status of the DMA controller through the RD line.

(ii) BG = 1 means that CPU has granted the DMA controller's request of releasing the bus and responds by enabling the bus grant line. The bus is now free and DMA controller can now control them.

(d) Now the bus is under the control of the DMA controller. CPU now opts out of the transfer and carries on with other tasks.

(e) DMA controller now places the initial address of the memory location from where the data is to be transferred into the address bus from its address register. Then the controller sends an acknowledgement to the I/O device, which has requested for the DMA transfer.

(f) Based on the content of the control register, DMA enables the RD or WR line i.e. if data is to be transferred from I/O device to memory then it's a write operation and WR line will be enabled else the RD line will be enabled.

(g) On transfer of each word, the word count register in the DMA controller decrements by one.

(h) If the entire block of data is transferred or if the word count register count down to zero, the DMA stops the transfer.

(i) It then removes the bus request by disabling the BR line. CPU then disables the BG line and again takes control of the bus. The DMA controller may also inform the CPU that the transfer is over and it has released the bus by sending an interrupt signal to the CPU.

(j) CPU then checks whether the transfer has been completed successfully by reading the count of the word count register. The zero value in the register indicates that the transfer has been done successfully.

The figure 7 shows the entire DMA process:

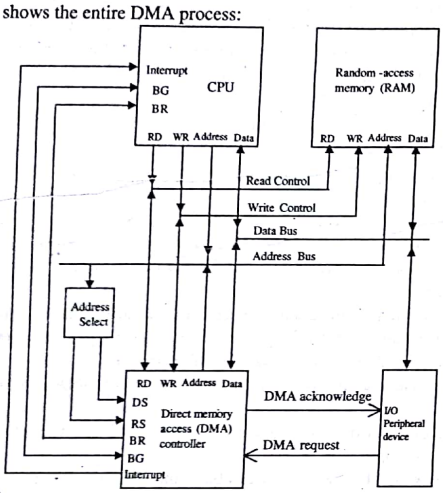

Fig: 7

Question 10

How does polling work? **[WBUT 2006]**

Answer: Refer to question number 7.

Question 11

Differentiate Isolated I/O and memory mapped I/O. **[WBUT 2004]**

OR,

Differentiate between memory mapped I/O and I/O mapped I/O. **[WBUT 2008, 2010]**

Answer: Refer to section 7.3.

Question 12

Differentiate between synchronous and asynchronous data transfer.

Answer:

Differences between ***synchronous*** and ***asynchronous*** data transfer:

Sl. No.	Synchronous data transfer	Asynchronous data transfer
1.	The two communicating units share a common clock frequency.	The two communicating devices operate depending on their own clock time, which are not known to each other.
2.	Each item is transferred during a time slot known in advance to both the units.	Not known in advance and each item being transferred is accompanied by a separate control signal to indicate its presence to the other unit.
3.	In such data transfer, bits are transmitted continuously at the common clock rate and the communicating channel does not remain idle any time.	Here bits are sent only when they are available otherwise the communicating channel remains idle.
4.	Here the communication performance is mainly dependent on how fast the device is able to transfer data. So faster devices cannot communicate at their maximum capability.	Each device can communicate according to slow their own needs and wish. They are not dependent on each other.
5.	Transmission rates, generally, are not flexible.	Flexible.
6.	Fast.	Slow.

Question 13

Explain the basic Direct Memory Access (DMA) operation for transfer of data bytes between memory and peripherals. **[WBUT 2004, 2007, 2008, 2010]**

Answer: Refer to review question number 9 and parts of section 7.8.

Question 14

Differentiate between serial and parallel data transfer.

Answer:

Differences between serial and parallel data transfer:

Sl. No.	Serial transfer of data	Parallel transfer of data
1.	Each bit in the message to transfer is sent sequentially one at a time.	All the bits in a message (i.e. the entire message) are transmitted at the same time.

Sl. No.	Serial transfer of data	Parallel transfer of data
2.	One (or one pair of) common wire to transmit all the bits one by one.	Each bit has its own separate path (wire) of transmission i.e. if there are k-bits in a message then there are k separate wires.
3.	Slow.	Fast.
4.	Requires only one pair of wires (or one conductor).	For k-bit message requires k different conductors i.e. number of wires required are much more.
5.	Less expensive.	More expensive.
6.	Useful in case of long-distance data transfer.	Useful for short-distance data transfer.

Question 15

Differeatiate between the different mechanisms of asynchronous data transfer.

Answer:

The different mechanisms of asynchronous data transfer are the strobe control method and handshaking method respectively. Differences between them are:

Sl. No.	Strobe control method	Handshaking method
1.	In this method the sender unit supplies a strobe pulse to indicate the other unit when the transfer has to occur.	Here each data item transferred is accompanied by a control signal, which indicate the presence of the data in the bus.
2.	Single control line is required for each transfer.	Two control lines are required for each transfer.
3.	Transfer is assumption based.	Transfer is acknowledgment based.
4.	Less reliable.	More reliable.
5.	Less flexible.	More flexible.
6.	'Timeout' mechanism is not used.	Uses 'timeout' mechanism to find out whether the data has been actually transferred or not.
7.	Example: Data transfer between CPU and an interface unit.	Example: Data transfer between an interface unit and the respective I/O device.

Question 16

Write short notes on:
a) Asynchronous data transfer **[WBUT 2004]**
b) Daisy chaining. **[WBUT 2004]**

Answer:

a) Refer to sections 7.10.
b) Refer to review question number 7.

Question 17

Explain strobe control method of asynchronous data transfer with the help of diagrams.

Answer:

In the strobe control method, one of the communicating units supplies the strobe pulse to let the other know when the transfer has to occur. This method uses a single control line to time each data transfer and this strobe pulse may be activated by either of the sender and the receiver units. The signal (strobe pulse) goes to the other unit via the bus to indicate it, when a valid data is available on the bus.

Depending on which unit is initiating the strobe pulse, strobe control mechanism is of two types:

(a) Source-initiated strobe for data transfer:

In this type of mechanism, the source unit activates the strobe pulse. Such transfer occurs as follows:

Steps:

(i) Source unit first places the data to be transmitted in the data bus.
(ii) After some delay, the source unit activates the strobe pulse to let the destination unit know that a valid data is there on the data bus.
(iii) The data and the strobe signal remain in the active state (i.e. remain valid) for quite some time for the destination to receive them.
(iv) After the time period, source unit disables the strobe pulse and then removes the data from the data bus. The source unit works on the assumption that the destination unit has received the data.

Figure 8, (a) and (b), shows the entire process and the corresponding timing diagram:

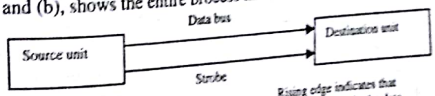

Fig: 8(a)

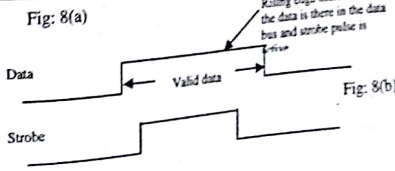

Fig: 8(b)

(b) Destination-initiated strobe for data transfer:

In this type of mechanism, the destination unit activates the strobe pulse to inform the source to provide data. Such transfer occurs as follows:

Steps:

(i) The destination unit wants data from the source unit and thus activates the strobe pulse to inform the source to place data on the data bus.
(ii) In response, the source unit places the data on the data bus.
(iii) After some predefined time interval, the destination unit disables the strobe pulse.
(iv) Source unit also removes the data from the data bus on assumption that the destination unit has received the data.

Figure 9, (a) and (b), shows the entire process and the corresponding timing diagram:

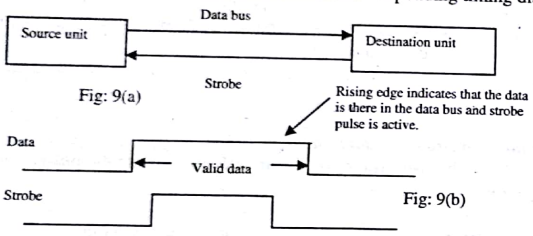

Fig: 9(a)

Fig: 9(b)

Question 18

Explain the handshaking method of asynchronous data transfer with the help of diagrams.

Answer:

The handshaking method of asynchronous data transfer solves the problem of strobe control method. This is acknowledgment based i.e. in this mechanism, each data item being transferred is accompanied by a control signal to indicate the presence of data in the bus. Also the destination unit responds back with a control signal to acknowledge the source that the data has been received successfully. This method uses a second control signal to provide a reply to the other unit. Hence handshaking method uses two control lines.

This method is also of two types:

(a) Source-initiated transfer using handshaking:

Here the source unit initiates transfer. Two handshaking lines, 'data valid' and 'data accepted' are used here.

Steps:

(i) The source unit places data on the data bus and enables the data valid control line to indicate the destination that the data is placed on the data bus.
(ii) The destination unit on receiving the data, enables the data accepted signal to let the source know that the data has been received successfully.
(iii) On getting this acknowledgement from the destination, the source unit disables the data valid signal and then removes the data from the bus.
(iv) When the destination sees that the 'data valid' signal is disabled, understanding that the acknowledgment signal has been received, it disables the 'data accepted' signal and gets ready to accept another data from the source.

Figure 10, (a) and (b), shows the entire process and the corresponding timing diagram:

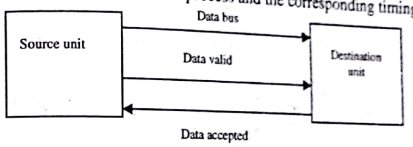

Fig: 10(a)

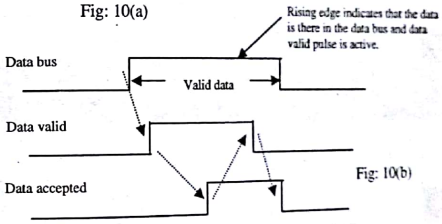

Fig: 10(b)

(b) Destination-initiated transfer using handshaking:

Here the destination unit initiates the data transfer. In this case, the source unit places the data on the data bus only after knowing that the destination unit is ready to accept data. Two handshaking lines, 'data valid' and 'ready for data' are used here.

Steps:

(i) The destination unit gets ready to accept data and thus enable the 'ready for data' control signal to let the source place the data on the data bus.
(ii) The source on getting this signal, places the data on the data bus and enables the 'data valid' signal to inform the placement of a valid data on the data bus.
(iii) The destination unit then accepts the data and disables the 'ready for data' signal.
(iv) The source understands that the data has been successfully received and thus disables the 'data valid' signal and removes the data from the data bus.

Figure 11, (a) and (b), shows the entire process and the corresponding timing diagram:

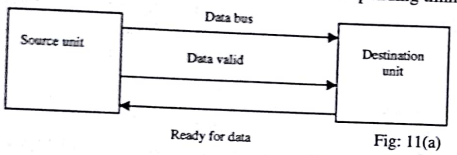

Fig: 11(a)

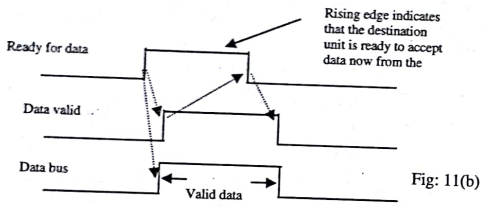

Fig: 11(b)

Question 19

What is interrupt? What are the differences between vectored and non-vectored interrupt? **[WBUT 2006, 2009]**

Answer: Refer to sections 7.5 and 7.6.

Question 20

Give the main reason why DMA based I/O is better in some circumstances than interrupt driven I/O. **[WBUT 2004, 2007]**

Answer:

In interrupt-driven I/O (as well programmed I/O) transfer of data between memory and an I/O module takes place with the active intervention of the CPU. The speed of such data transfer is often limited by the speed with which the CPU services a device. For transfer of small volume of data such CPU intervention (and thus limited speed of data transfer) is ok, but transfer of large volume of data via the CPU takes a lot of time. So while transfer of large volume of data between memory and I/O module by DMA based I/O method, CPU is removed from the path (CPU only initiates the transfer) and data flows very fast directly between the memory and the concerned I/O module(s) with the I/O device managing the memory buses.

Also during interrupt-driven I/O method, CPU, along with handling I/O tasks, is also busy handling other tasks. This, to some extent, hampers the I/O transfer rate. However with DMA based I/O method nothing of that sort occurs.

Hence for the above two reasons DMA based I/O method is better than interrupt-driven I/O method while transfer of large volume of data between memory and I/O module.

Question 21

What are vectored interrupts? How are they used in implementing hardware interrupts?

Answer:

Vectored interrupt or vector interrupt refers to a situation where after recognizing the source of an interrupt, the CPU seeks the address of the Interrupt Service Routine (ISR) from the device and the device, in response, provides the address vector. Since its address vector is fixed and known beforehand, the vector interrupt scheme can be implemented in hardware as follows:

The device permanently stores its address vector (a binary pattern) and connects it to the system address bus via tristate buffers that are normally kept disabled. When the device interrupts the CPU and the CPU acknowledges the interrupt by sending an Interrupt Acknowledge (INTA) signal, the device then utilizes this INTA signal to enable (activate) the tristate buffers so that the address vector is put on the address bus and the CPU can receive this address and jump to the ISR of the device to provide service.

Question 22

What is input device? How does it differ from output device? **[WBUT 2006]**

Answer: Refer to section 7.1.

Question 23

How stack is useful in subroutine call? **[WBUT 2004]**

Answer:

A subroutine is actually a self-contained sequence of instructions that performs a given computational task each time it is called at various points in the main program. When a subroutine is called, the control branches to the beginning of the subroutine to start executing its set of instructions and on finishing, control branches back to the main program and start its execution again from the next instruction onwards.

When the execution of subroutine finishes, control needs to get transferred to the main program starting the execution from the immediate instruction onwards. So the return address must be stored somewhere before the control branches to the subroutine. Using stack is a well-known and efficient mechanism for storing the return addresses. The advantages for using such architecture are that the return addresses, in case of succession of subroutine calls, can be pushed sequentially into the stack. On return, the respective return addresses of the subroutines can be sequentially popped out (in the reverse manner) from the stack and loaded into the program counter. So return is always to that particular program that last called a subroutine.

Also such automatic and collective storage mechanism of all the return addresses in a single hardware unit is easy to use and implement.

Question 24

In a system, the user gets the impression that the I/O is synchronous, whereas actually the processor does asynchronous I/O. Explain this anomaly.

Answer:

The processor in actual, performs asynchronous I/O. While an I/O device is doing a transfer, the processor switches itself to a different task. But to free the user from botherations regarding the low-level details of I/O transfer timings and interrupts and to provide a simple programming scenario, the user program is provided with a synchronous programming interface that looks like a function call. Such function calls occur as and when needed without bothering about the underlying scenario.

Question 25

What are the different types of DMA controllers and how do they differ in their functioning? **[WBUT 2006, 2010]**

Answer: Refer to section 7.8.

Question 26

What is programmed I/O technique? Why is it not very useful? **[WBUT 2004, 2007, 2010]**

OR,

What is programmed I/O technique? **[WBUT 2008]**

Answer:

Programmed I/O transfers are initiated as a result of I/O instructions written in the computer program. Once the transfer is initiated, the CPU has to constantly monitor the interface units of the I/O devices to see when the transfer can be actually made i.e. when the device will be ready to transfer data.

It is not very useful, as this constant monitoring of the devices by the CPU requires lot of CPU time. In these time, the CPU remains idle as it cannot do any other work but to check whether the I/O device is ready to transfer data or not. Lot of CPU time is wasted in this mechanism. So for this reason this technique is the slowest compared to interrupt I/O and DMA mechanisms and thus degrades the computer performance.

Question 27

Differentiate between Programmed I/O and Interrupt I/O.

Answer:

Differences between programmed I/O and interrupt I/O are:

(a) In ***programmed I/O*** method, CPU has to constantly monitor whether a device is ready to transfer data whereas in ***interrupt I/O***, the device when ready, itself indicates the CPU to provide it service.

(b) In ***programmed I/O***, much of CPU cycle is wasted, as CPU has to remain idle checking whether the device is ready to transfer or not whereas in ***interrupt I/O***, CPU continues with other jobs and device itself indicates CPU when ready. So CPU cycle is not wasted at all.

Question 28

What are the different types of interrupt? Give examples. **[WBUT 2008]**

Answer: Refer to section 7.5.

Question 29

Name the different types of errors that a I/O component has to handle in a system.

Answer:

These are of the following types:

A) Data errors: Errors related to any data while programming. For example, checksum or parity errors while reading data from memory.

B) Operational errors: Errors that may creep in while doing I/O operations. For example, buffer overflow, wrong controller commands, breakdown in communication between I/O driver and controller.

C) Transient errors: These are temporary errors like a voltage spike or dust spot (mark) in the I/O device or controller.

D) Permanent errors: These may be caused from damaged disk sector or damaged or burnt memory cell.

Question 30

The DMA break points usually occur at a point when the CPU is about to use the bus. Why won't it be considered to be useful to steal a DMA cycle when the CPU is not using the bus? For example, during the decode state in an instructional cycle.

Answer:

Precisely speaking, the points at which the CPU is about to use the bus are usually the points at which it has finished all previous bus transfers, and the bus is free to be stolen. This may not be guaranteed at other points in the instruction cycle. Now, the instruction fetch and decode sates are basically two different logical states in an instruction cycle. In practice, between two consecutive bus operations, a minimum allowable delay is required. But, it happens that the delay following the fetch state may creep into (or overlap into) the decode state, thus making it impossible to steal the bus during the decode state.

Question 31

What are the different modes of an I/O interface? Explain them.

Answer:

The I/O interface has two different modes: raw and cooked.

In a raw mode, the unit of communication is a character i.e. in this mode, every character is delivered or transferred individually without waiting for the end of line character. So, some special characters like erase or backspace, if needed, has to be done by the user process (and are not done by the terminal I/O driver).

Example: Any sort of interactive applications like games or graphics packages uses the raw mode.

In a cooked mode, unlike the raw mode, the communication unit is a line. In this case, much less interactions are required between the I/O device and the user process. The I/O device sends/processes an entire line when it is ready. So, erase or backspace characters can be handled easily in this mode. Cooked mode is more common than the raw mode.

Question 32

What is interrupt? What are the differences between vectored and non-vectored interrupt? **[WBUT 2009]**

Answer:

Refer to section 7.5 for the first part of the question.

Refer to section 7.6 for the second part.

Question 33

Explain the different functions that should be completed as part of a vectored interrupt initialization process.

Answer:

The following functions or steps need to be completed by the system.

- The device interrupts need to be enabled by setting the appropriate registers.
- CPU interrupt mask is to be set in order to allow getting interrupt requests from the device.
- The interrupt vector is to be initialized with the address of the Interrupt Service Routine.
- Interrupts should be enabled globally.

Question 34

Discuss the advantages and disadvantages of a memory-mapped I/O method.

Answer:

In memory-mapped I/O, the greatest advantage is that a single bus structure can be used all the time. Along with this, this method does not require any additional software instructions.

On the other hand, the greatest disadvantage is that all the different address spaces used for devices, are not available for memory. This means that this logical separation between the address spaces and the memory spaces, effectively reduces the actual / overall memory space available for storage.

Question 35

Where dose DMA mode of data transfer find its use? **[WBUT 2009]**

Answer: Refer to section 7.8.

Question 36

In an interrupt-service routine (ISR), it becomes important to check the status of a device before transferring the data. Why do you think is this necessary?

Answer:
This is necessary for the following reasons:
In a system where there can be multiple sources of an interrupt, the ISR needs to poll / check the status of each possible source. This is because, in a situation where an interrupt request may occur for both transmit and receive interrupts, the ISR needs to specifically determine the desired action before transferring any data.
On the other hand, this is somewhat different in case of a single source for an interrupt. If the request comes from software or on-chip interrupt sources, then the ISR may service the request without specifically checking the status but for an external interrupt request, the ISR needs to check the status I order to avoid servicing a false interrupt.

Question 37

What is interrupt latency? What can be the probable causes of this?

Answer:
Interrupt latency is the period between the generation of an interrupt by a device and the servicing of that interrupt generated by that device.
Interrupt latency depends on multiple different factors like the interrupt masking, nature of interrupt controllers and the interrupt handling methods of the individual operating systems. A trade-off exists between the interrupt latency, processor utilization and throughput of the system. So, improving the interrupt latency may sometime decrease the throughput and processor utilization and vice versa. So, in order to take care and improve the throughput and processor utilization, some systems deliberately increase / hamper the interrupt latency.

Question 38

What are the different methods for I/O synchronization?

Answer:
The five different methods for I/O synchronization are:

- Periodic Polling
- Occasional Polling
- Blind Cycle
- Interrupts
- Polling Loop.

Question 39

What are the different types of DMA controller and how do they differ in their functioning? [WBUT 2009]

Answer: Refer to section 7.8.

Question 40

Explain the significance of Input/output Interface? [WBUT 2003]

Answer:
I/O interface units are special hardware components that lie between the I/O devices and the processor bus. These devices serving as special communication links, between the CPU and the peripherals, actually synchronize and supervise i.e. control all input and output transfers.
Certain differences between the characteristics (functional and behavioral) of the peripherals and the CPU (and also memory) exist. Interface units are needed to resolve the differences, which are:
(a) Operations of the peripherals, which are either electromagnetic or electromechanical devices, are different from that of the CPU and memory, which are generally electronic devices. So a signal conversion between the two is required and is done by the interface unit between them.
(c) The interface unit has to synchronize the data transfer mechanism between the slow peripherals and the fast CPU.
Interface units handle the differences between the data codes and formats in peripherals and the CPU or memory word formats.

Question 41

Explain the concept of spooling.

Answer:
Sometimes, I/O devices are unable to interleave between the different job requests they get simultaneously in the form of data streams. Such requests are stored in special buffers called spools. For example, although a printer can serve only one job at a time, it may receive several printing requests simultaneously. So, while it prints in a first come first serve basis, remaining jobs are spooled separately.

Question 42

Where does DMA mode of data transfer find its use?

Answer:

DMA is needed to transfer of block of data between the fast magnetic disk and memory. If the transfer is done via the CPU (speed of magnetic disk almost same as of CPU) then time required to transfer the entire block would have been much more. So, when data transfer must be done in blocks and when communicating devices are very fast then DMA is preferable. Generally in the computer, DMA mode of data transfer occurs.

Question 43

Explain how an I/O operation based on an interrupt is more efficient than an I/O operation based on a programmed mode of operation.

Answer:

Programmed I/O transfers are initiated as a result of I/O instructions written in the computer program. Once the transfer is initiated, the CPU has to constantly monitor the interface units of the I/O devices to see when the transfer can be actually made i.e. when the device will be ready to transfer data.

It is not very useful, as this constant monitoring of the devices by the CPU requires lot of CPU time. In these times, the CPU remains idle as it cannot do any other work but to check whether the I/O device is ready to transfer data or not. Lot of CPU time is wasted in this mechanism. So for this reason this technique is the slowest compared to interrupt I/O and thus degrades the computer performance.

In the ***Interrupt I/O*** mechanism, the CPU need not constantly monitor to see when the I/O device will be ready to transfer data. Here the I/O device itself will let the CPU know that it is ready to transfer data and that it wants service from the CPU, meanwhile the CPU can continue to execute other programs. When the device gets ready to transfer data, it will send a signal (interrupt) to the CPU to provide service. CPU will then stop other tasks and will branch to the service program of that I/O device to process the I/O data transfer. When finished, CPU will return to the task it was originally performing and continue with it. Here CPU cycle is not wasted, as the CPU need not remain idle.

Differences between programmed I/O and interrupt I/O are:

(a) In ***programmed I/O*** method, CPU has to constantly monitor whether a device is ready to transfer data whereas in ***interrupt I/O***, the device when ready, itself indicates the CPU to provide it service.

(b) In ***programmed I/O***, much of CPU cycle is wasted, as CPU has to remain idle checking whether the device is ready to transfer or not whereas in ***interrupt I/O***, CPU continues with other jobs and device itself indicates CPU when ready. So CPU cycle is not wasted at all.

Question 44

What is a 'macro'? What is the basic difference between a 'macro' and a 'sub-routine'?

Answer:

A macro is a way to automate tasks that are performed repeatedly on a computer. It refers to a series of commands and actions that can be stored and run whenever you need to perform the task. A macro can be stored / recorded, and then played in order to automatically repeat the series of commands or actions.

Differences between macros and subroutines are:

Subroutine: Any task to be done repeatedly is written as a subroutine. The subroutine will be called each time to execute that task. Subroutines will be stored in some memory location and the program control will be transferred to that location each time when the subroutine is called.

In ***macro*** the number of instructions will be less than that in a subroutine. A macro can be stored / recorded, and then played in order to automatically repeat the series of commands or actions. It is faster than subroutine. However, memory requirements in macros are more than subroutine.

Short Questions & Answers

1. What is the difference between an exception/trap and an interrupt?
Answer:
An interrupt is caused by an external event and an exception/trap is generated within the program. Examples of exceptions/traps are: divide-by-zero, page fault (MMU exception), software interrupt.

2. What are device-independent I/O software?
Answer:
Such software are the set of programs, which provide device-independent interface to the user i.e. with the help of such device-independent software, an user can program in a device-independent manner irrespective of any device

3. What are the disadvantages of the strobe control method?
Answer:
The strobe control method is fully based on assumptions. The source unit has no confirmed way of knowing whether the data has been actually received by the destination unit or not. Similarly, the destination unit cannot be confirmed that the source has actually placed the data on the data bus.

4. What is an interrupt execution cycle?
Answer:
A CPU can either execute an instruction (instruction cycle) or it can handle an interrupt request from an I/O device (instruction cycle). So interrupt execution cycle shows the conditions that lead the CPU to branch to an interrupt handling process and how the CPU handles the interrupt.

5. DMA makes a CPU to wait. In that case, how can DMA be termed as a better option?
Answer:
In DMA, the CPU waits for one bus cycle per word, whereas in programmed I/O, it would take one instruction cycle per word, to execute the transfer instruction and an instruction cycle is typically larger than a bus cycle.

6. What are the advantages of interrupt-driven I/O over polling?
Answer:
In an interrupt-driven I/O, unlike polling, while the device driver is doing a work, the CPU can meantime do some other work. It need not wait for the device driver, which

takes thousands to millions of CPU cycles for a work, to come to an end. So, unnecessary wastages of CPU cycles can be avoided.

7. What is program status word?
Answer:
Program status word or PSW is the collection of all status bit conditions in the CPU. When the CPU branches to the ISR of an I/O device, the PSW of the CPU state (before branching to the ISR) is stored so that on coming back, CPU can resume the execution of the unfinished program exactly as before (i.e. as if nothing has happened).