**UNIT-IV  
 INPUT/OUTPUT ORGANIZATION**

**Syllabus:**

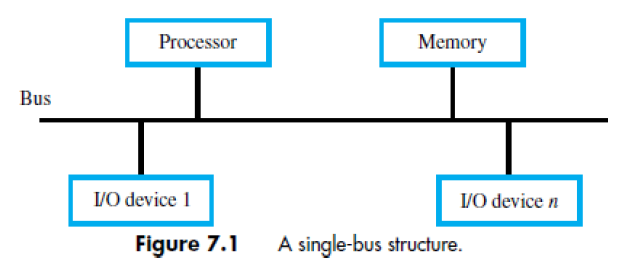
* I/O-mapped I/O and memory-mapped I/O,
* interrupts and interrupt handling mechanisms, vectored interrupts,
* synchronous vs. asynchronous data transfer,
* Direct Memory Access
* COMPUTER PERIPHERALS:
  + I/O devices such as magnetic disk,
  + magnetic tape
  + CDROM systems

**I/O-mapped I/O and memory-mapped I/O:**

**Accessing I/O devices:**

* A **single bus-structure can be used for connecting I/O-devices to a computer.**
* Each I/O device is assigned a unique set of address.
* Bus consists of 3 sets of lines to carry address, data & control signals.
* When processor places an address on address-lines, the intended device responds to the command.
* The processor requests either a read or write-operation.

The requested-data are transferred over the data-lines.



There are 2 ways to deal with I/O-devices:

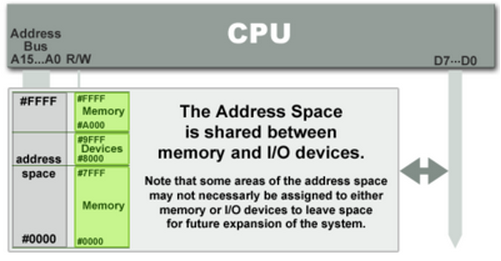
1) Memory-mapped I/O &

2) I/O-mapped I/O

**Memory-mapped I/O:**

* In case of single-bus machine , the same bus serves as both the memory and I/O bus.
* Hence, the address of I/O devices are not distinguished from the address of the memory location.
* Therefore, to select I/O devices, a portion of memory address space is used and in them unique address codes are placed.
* In this way, it is possible to access I/O device like that of any memory location.
* This is known as memory mapped I/O.

**Same address bus to address memory and I/O devices:**



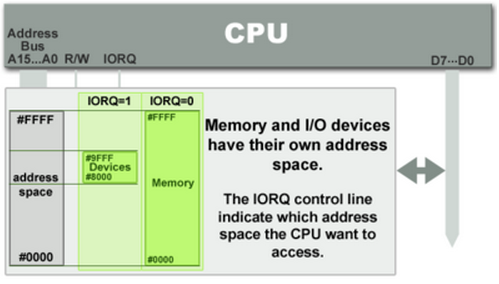
**Advantage and disadvantage:**

1. Every instruction which can access memory can be used to manipulate an I/O device.
2. Entire address bus must be fully decoded for every device.  For example, a machine with a 32-bit address bus would require logic gates to resolve the state of all 32 address lines to properly decode the specific address of any device. This increases the cost of adding hardware to the machine.

**I/O mapped I/O:**

* Here, the memory and I/O address space are kept separate by read/write control lines.
* A memory reference instruction activates the Read M or Write M line and does not affect the I/O device.
* The I/O instruction such as for example IN or OUT activates the Read I/O and write I/O line and causes a word to be transferred between the addressed I/O and the CPU.
* Therefore, the memory location and the I/O device may have the same address.
* This is known as I/O mapped I/O.

**Different address spaces for memory and I/O devices:**

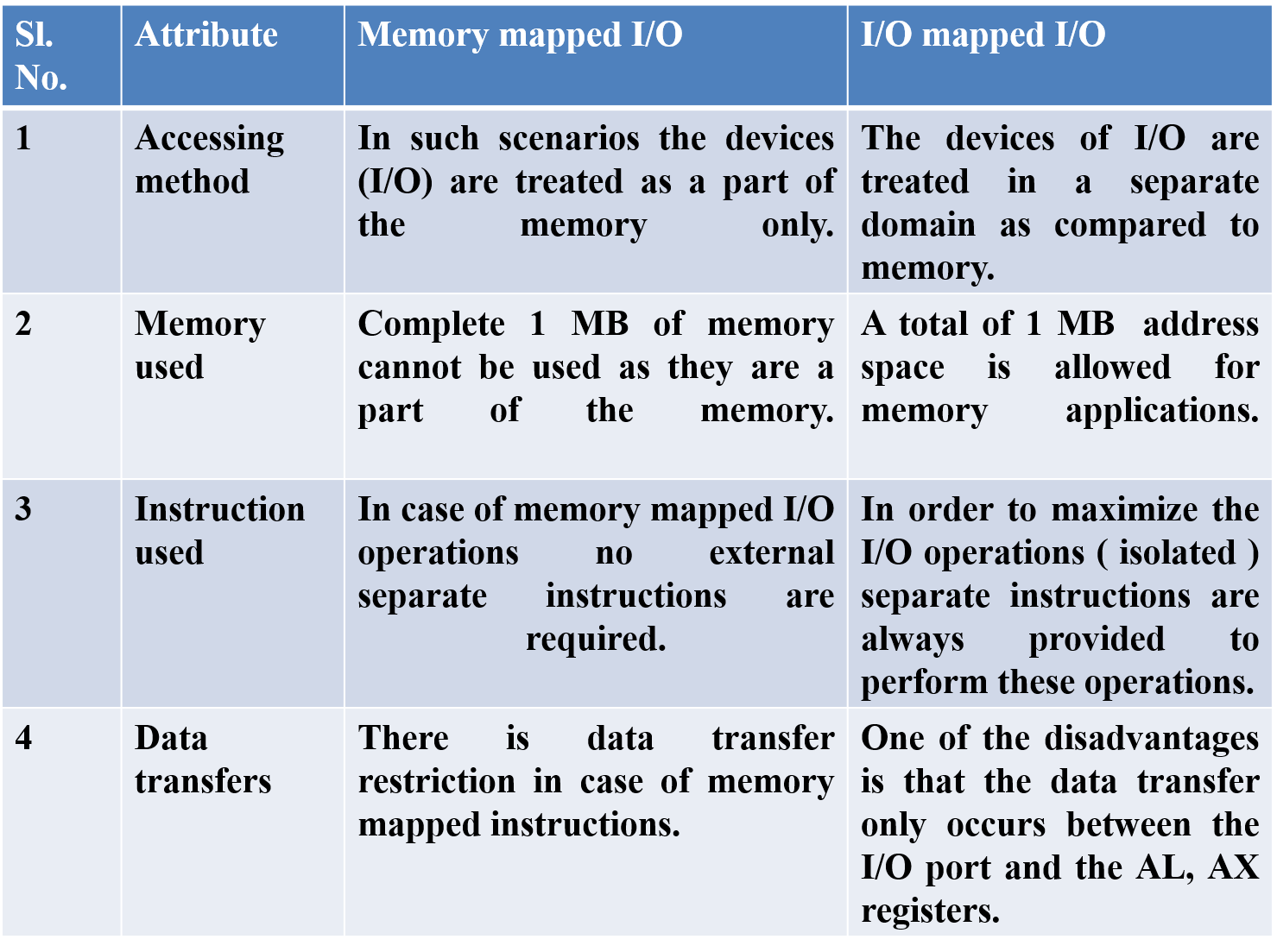


**Advantages and Disadvantages:**

1. Less logic is needed to decode a discrete address and therefore less cost to add hardware devices to a machine hardware

2. More instruction needed to perform single task because first the data sent to registers and then it performs the operation.

**Difference between memory mapped I/O and I/O mapped I/O:**



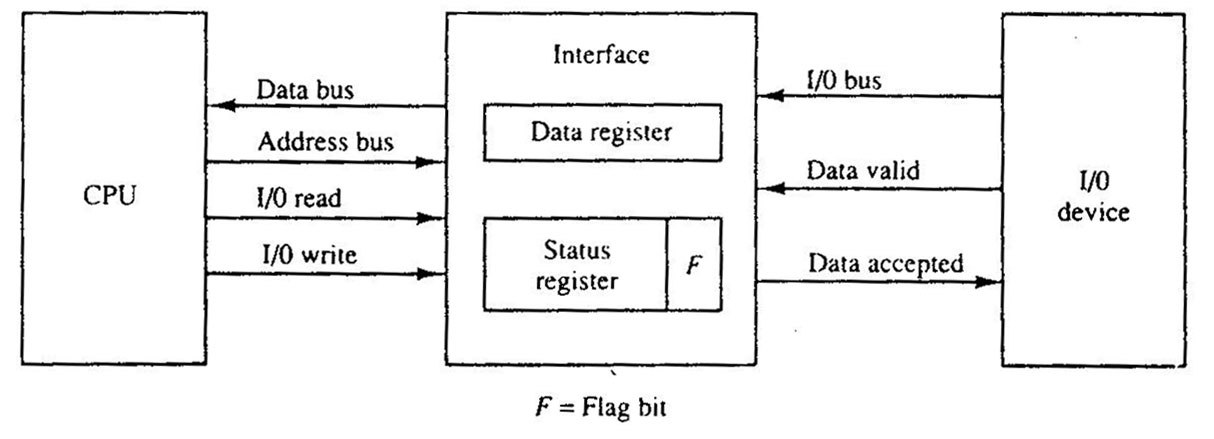
**Modes of transfer:**

* For later processing, binary information received from an external device is usually stored in memory.
* Information transferred from the central computer into an external device, initiates in the memory unit.
* The CPU only executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit.
* Data transfer between the central computer and I/O devices may be handled in a variety of modes. Some modes use the CPU as an intermediate path; others transfer the data directly to and from the memory unit.
* Data transfer to and from peripherals may be done in either of three possible modes:
  + 1. Programmed I/O
    2. Interrupt-initiated I/O
    3. Direct memory access (DMA)

**Programmed I/O:**

* In this method, the I/O device does not have direct access to memory.
* A transfer from an I/O device to memory needs the execution of several instruction by the CPU,
  + including an input instruction to transfer the data from the device to the CPU and
  + a store instruction to transfer the data from the CPU to memory.
* Other instruction may be needed to verify that the data are available from the device and to count the numbers of words transferred.
* Fig. below shows data transfer from an I/O device through an interface into the CPU.
* The device transfers bytes of data one at a time, as they are available. When a byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line.
* The interface sets a bit in the status register that we will refer to as a "flag" bit. The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled.

**Data transfer from I/O device to CPU:**



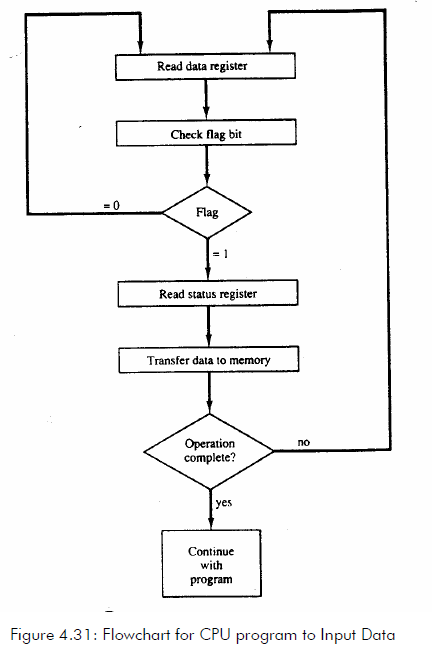
* A program is written to check the flag in the status register to determine if a byte has been placed in the data register by the I/O device.
* This can be done by reading the status register into a CPU register and checking the value of the flag bit.
* If the flag is equal to 1, the CPU reads the data from the data register.
* The flag-bit is then cleared to 0 by either the CPU or by the interface, depending on how the interface circuits are designed.
* Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.
* A flowchart of the program is shown in Fig. It is assumed that the device is sending a sequence of bytes that must be stored in memory. The transfer of each byte needs three instructions:

1. Read the status register.

2. Check the status of the flag bit and branch to step I if not set or to step if set.

3. Read the data register.

* CPU register read each byte and then transferred to memory with a store instruction. I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.



**Application of programmed I/O:**

* This method is used in small low-speed computers or in systems that are dedicated to monitor a device continuously.
* The difference in information transfer rate between the CPU and the I/O device makes this type of transfer inefficient.

**Interrupt-initiated I/O:**

* Another way of constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility.
* While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
* The CPU deviates from what it is doing to take care of the input or output transfer.
* The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.
* The way that the processor chooses the branch address of the service routine varies from one unit to another.
* In principle, there are two methods for accomplishing this.
  + One is called vectored interrupt and the other,
  + non vectored interrupt.
* In a non-vectored interrupt, the branch address is assigned to a fixed location in memory.
* Vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector. In some computers the interrupt vector is the first address of the I/O service routine. In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the I/O service routine is stored.

**Interrupt:**

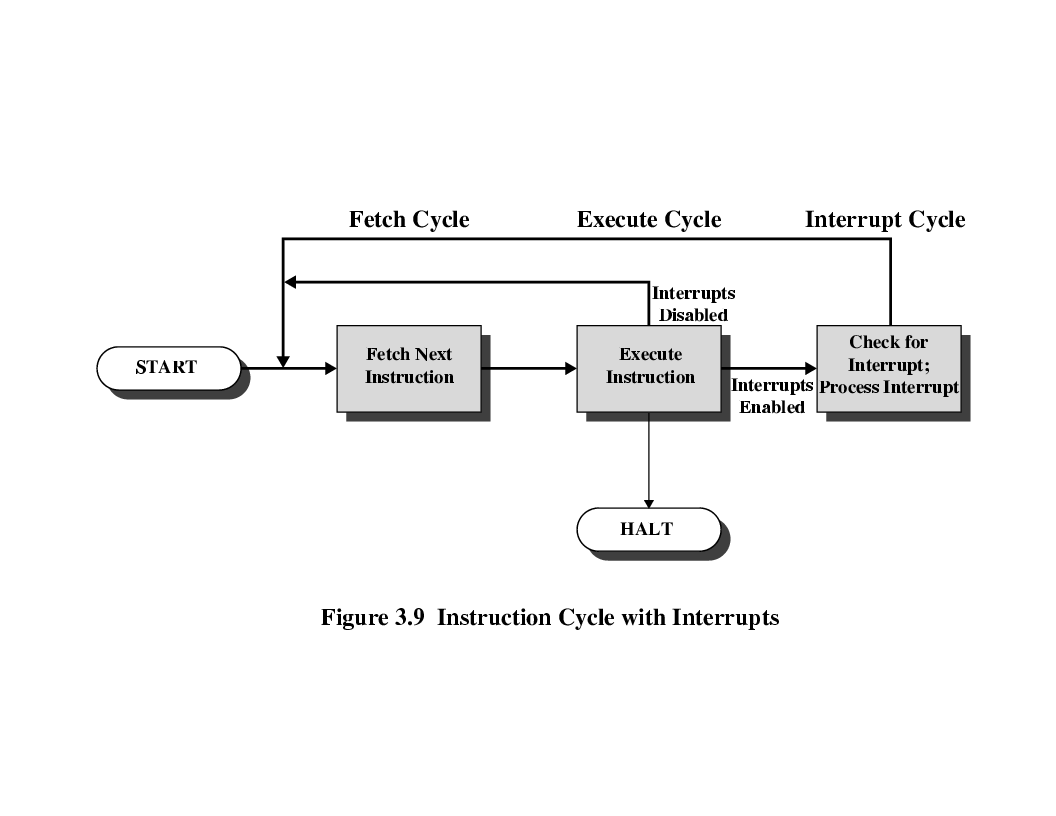
* **An interrupt is just that an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.**
* **Interrupts are provided primarily as a way to improve processing efficiency.**
* **For example, most external devices are much slower than the processor. With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.**
* **For I/O operation, say an output operation, like printing some information by a printer. Printer is much slower device than the CPU. The CPU puts some information on the output buffer. While printer is busy printing this information from output buffer, CPU is lying idle. During this time CPU can perform some other task which does not involve the memory bus. When the external device is ready to accept more data from the processor, the I/O module for that external device sends an interrupt request signal to the processor. The processor responds by suspending operation of the current program, branching off to a program to service the particular I/O device, and resuming the original execution after the device is serviced.**

**Interrupt cycle:**

**In the interrupt cycle, the processor checks if any interrupt have occurred, indicated by the presence of an interrupt signals.**

* **If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction of the current program.**
* **If an interrupt is pending, the processor does the following:**
  + **It suspends the execution of the current program being executed and saves its contents. This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the processor's current activity.**
  + **It sets the program counter to the starting address of an interrupt handler routine.**

**Instruction Cycle with Interrupts:**



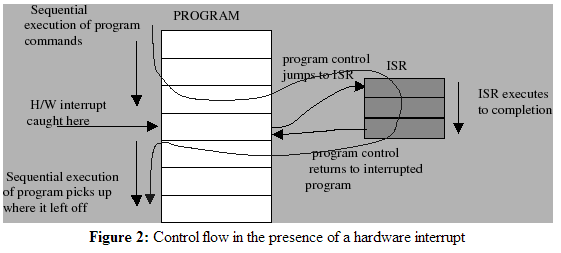
**Types of interrupts:**

**Virtually all computers provide a mechanism by which other module (I/O, memory etc.) may interrupt the normal processing of the processor. The most common classes of interrupts are:**

1. **Software: Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside the user's allowed memory space.**
2. **Internal or I/O: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.**

**Hardware/ External: If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts**

**Control flow in the presence of interrupt:**



**Handling multiple interrupts :**

**While handling multiple devices, the issues concerned are:**

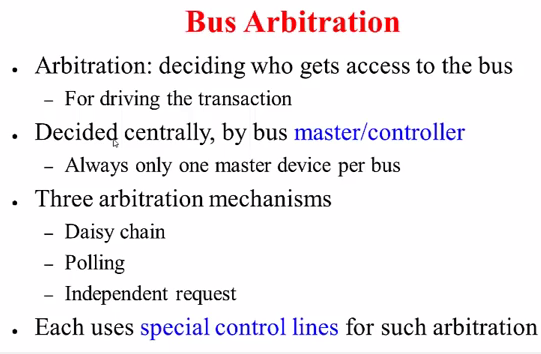
* **How can the processor recognize the device requesting an interrupt?**
* **How can the processor obtain the starting address of the appropriate ISR (Interrupt Service Routine)?**
* **Should a device be allowed to interrupt the processor while another interrupt is being serviced?**
* **How should 2 or more simultaneous interrupt-requests be handled?**

**An interrupt handler or interrupt service routine (ISR) is a software routine that hardware invokes in response to an interrupt. ISRs examine an interrupt and determine how to handle it. ISRs handle the interrupt, and then return a logical interrupt value.**

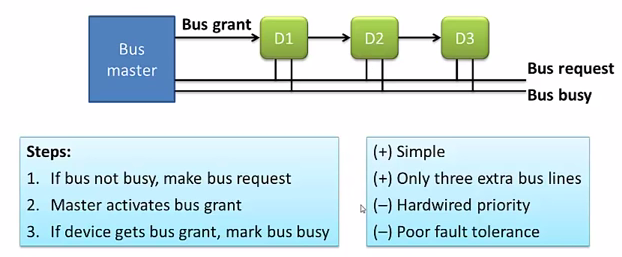
1. **Multiple interrupts-sequential**
2. **Multiple interrupts-nested**

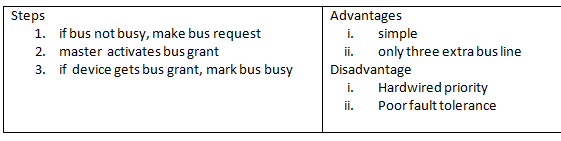
**Multiple interrupts-sequential:**

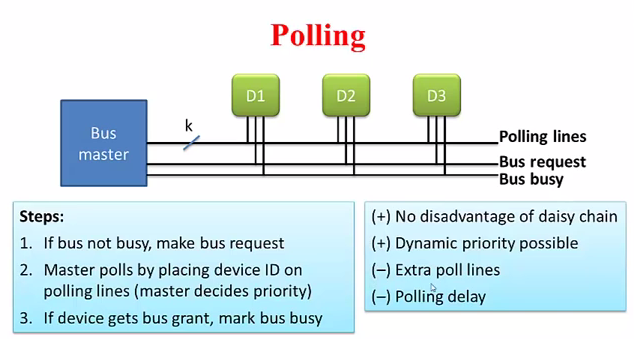
* To handle multiple interrupts sequential bus arbitration is used

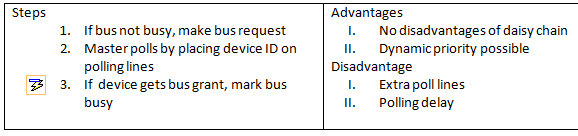


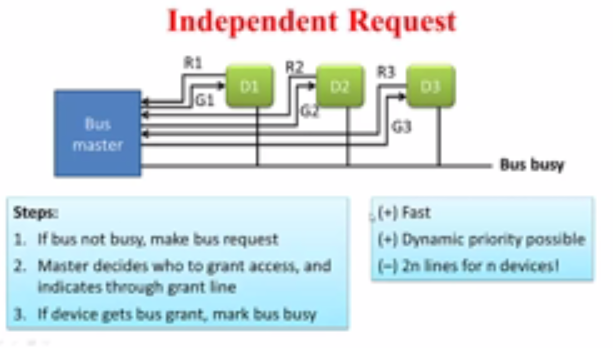
**Daisy chain:**

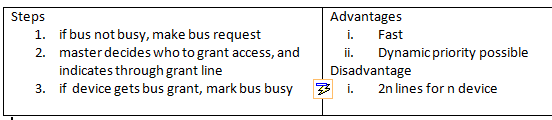












**Multiple interrupts-nested:**

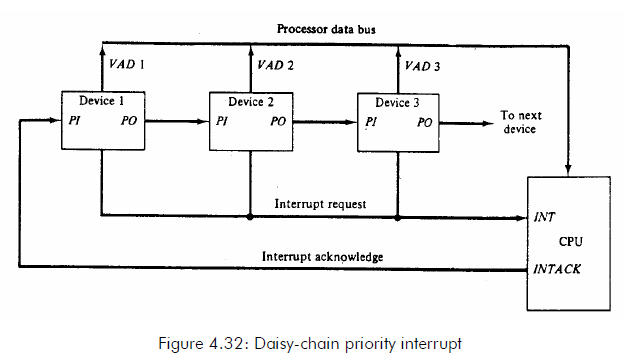
* To handle multiple nested interrupts priorities are used.
* Methods to establish priorities
  + **Logical way:** time sensitive situation has a high priority
  + **Hardware:** according to the physical location of device
  + **Software:** assigning a priority number to each interrupts

**Priority Interrupt**:

* A priority interrupt establishes a priority to decide which condition is to be serviced first when two or more requests arrive simultaneously.
* The system may also determine which conditions are permitted to interrupt the computer, while another interrupt is being serviced.
* Devices with high- speed transfers are given high priority, and slow devices receive low priority.
* When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.
* Establishing the priority of simultaneous interrupts can be done by software or hardware.

**Daisy-Chaining Priority:**

* The daisy-chaining method has a serial connection of all devices that request an interrupt.
* The device with the highest priority is kept in the first position, followed by lower-priority devices and so on. This method of connection is shown in Fig.4.32.
* The interrupt request line is common to all devices.
* When no interrupts are pending, the interrupt line stays in the high-level state and as a result CPU does not recognize any interrupt.
* The CPU responds to an interrupt request by enabling the interrupt acknowledge line.
* This signal is received by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.
* If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.
* It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.



**Parallel priority interrupt:**

* The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device.
* Priority is established according to the position of the bits in the register.
* In addition to the interrupt register, the circuit may include a **mask register whose purpose is to control the status of each interrupt request.**
* The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced.
* It can also provide a facility that allows a high-priority device to interrupt, while a lower priority device is being serviced.

**Synchronous vs. Asynchronous data transfer:**

**Synchronous data transfer:**

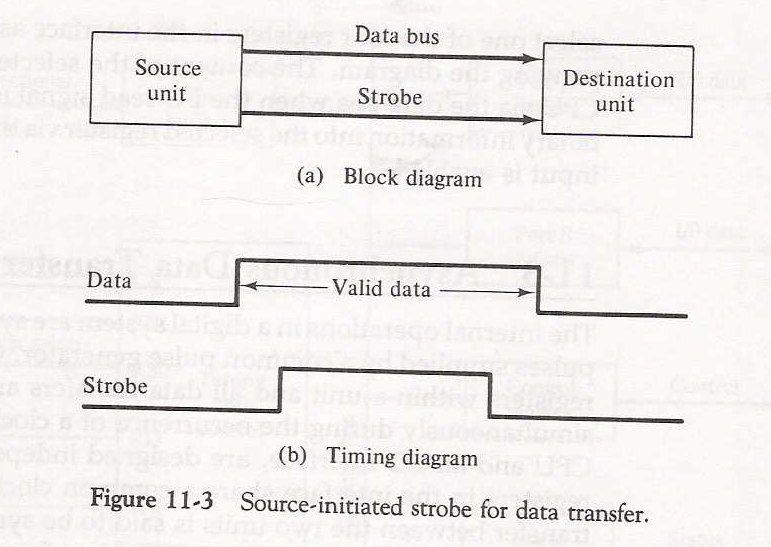
* The internal operations in a digital system are synchronized using a common pulse generator.
* Clock pulses are used by all registers within a unit and all data transfers among internal registers occur simultaneously during the occurrence of a clock pulse.
* Two units, such as a CPU and an I/O interface, are independent of each other.
* If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be synchronous.

**Asynchronous data transfer:**

* Asynchronous data transfer between two independent units requires that control signals be transmitted between the communication units to indicate the time at which data is being transmitted.
* Two methods to accomplish asynchronous data transfer
  + Strobe Control
  + Handshaking

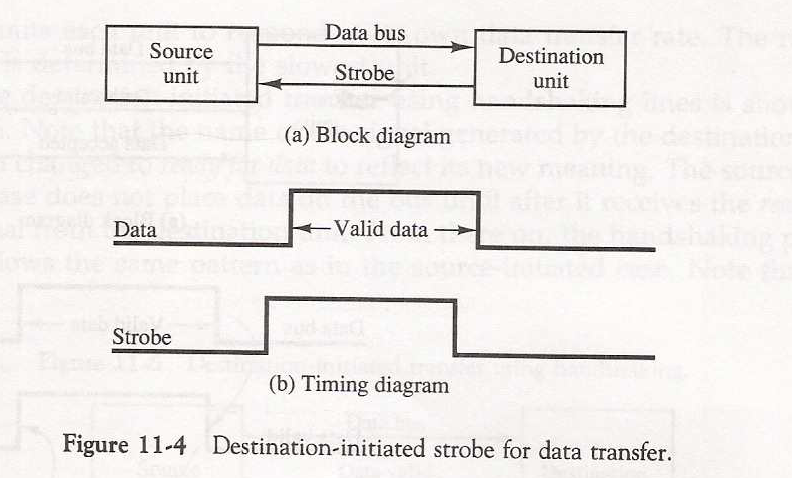
**Strobe Control:**

* The strobe control method of asynchronous data transfer employs a single control line to time each transfer.
* The strobe may be activated by either the source or the destination unit.
* The **data bus** carries the binary information from source unit to the destination unit.
* Typically, the bus has multiple lines to transfer an entire byte or word.
* The **strobe is a single** line that informs the destination unit when a valid data word is available in the bus.



**Source initiated strobe:**

* The source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse.
* The information on the data bus and the strobe signal do not change in the active state for a sufficient time period to allow the destination unit to receive the data.
* Often, the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers.
* The source removes the data from the bus a brief period after it disables its strobe pulse.
* Actually, the source does not have to change the information in the data bus. The fact that the strobe signal is disabled indicates that the data bus does not have valid data.
* New valid data will be available only after the strobe is enabled again.



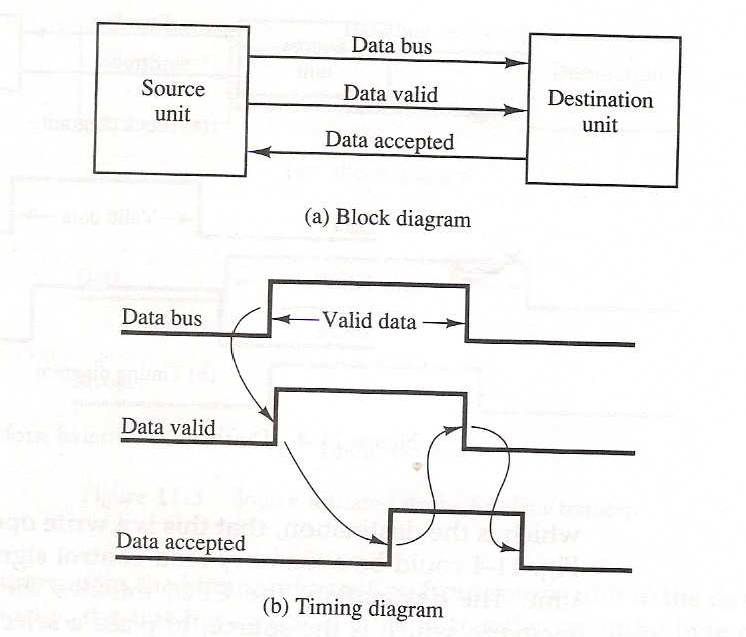
**Destination initiated strobe:**

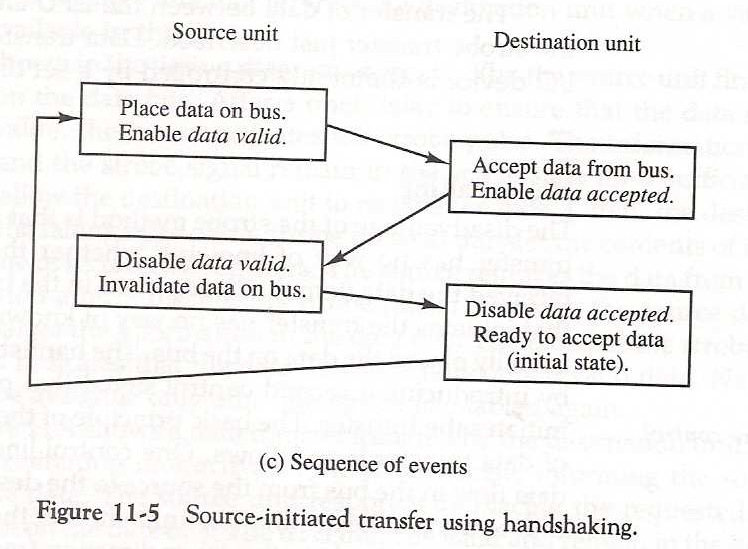
* Figure 11.4 describe a data transfer initiated by the destination unit. In this case the destination unit activates the strobe pulse, informing the source to provide the data.
* The source unit responds by putting the requested binary information on the data bus. The data should be valid and remain in the bus long enough for the destination unit to accept it.
* We can use the falling edge of the strobe pulse again to trigger a destination register. The destination unit then disables the strobe.
* The source removes the data from the bus after a predetermined time interval.

**Handshaking:**

* The strobe method has a disadvantage that the source unit that initiates the transfer has no method of knowing whether the destination unit has actually received the data item that was placed in the bus.
* Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.
* The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.
* The principle of the two-wire handshaking method of data transfer is as follows.
  + One control line is in the same direction as the data flow in the bus from the source to the destination. It is used by the source unit to inform the destination unit whether there are valid data in the bus.
  + The other control line is in the other direction from the destination to the source. It is used by the destination unit to inform the source whether it can accept data.
  + The sequence of control during the transfer depends on the unit that initiates the transfer.

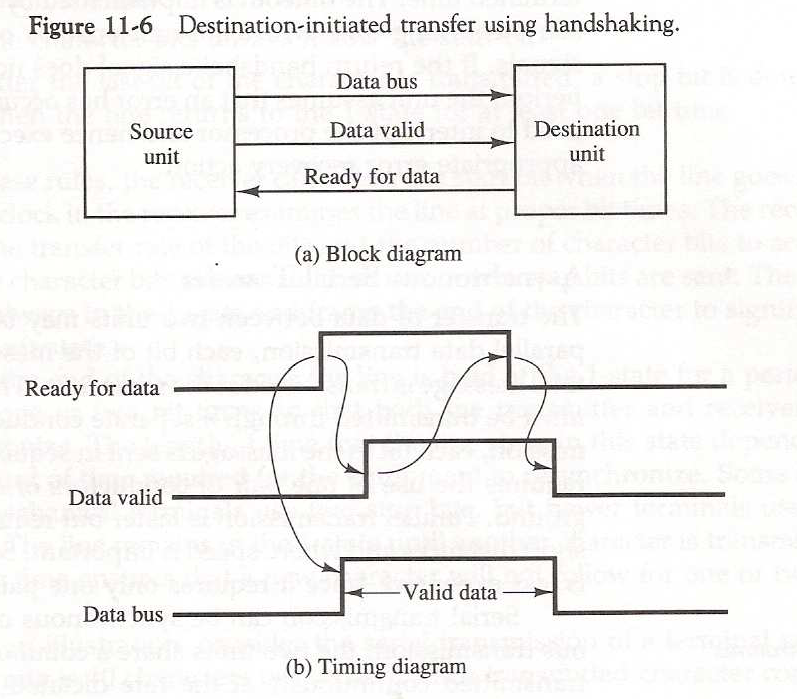
**Source initiated transfer using handshaking**:

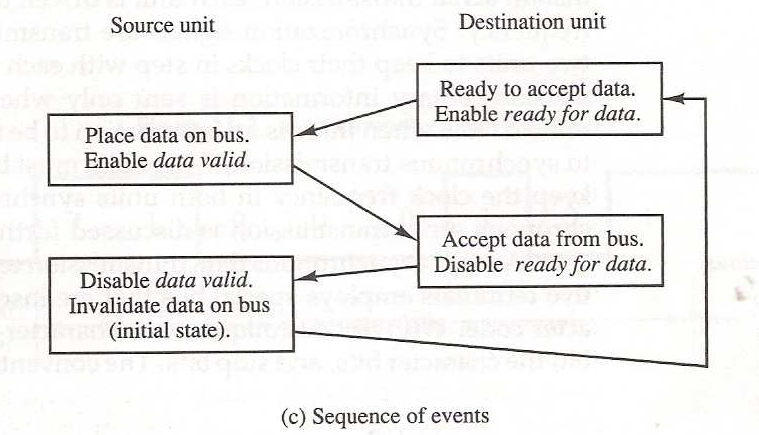




**Source initiated transfer using handshaking:**

* Figure 11.5 shows the data transfer procedure when source begins it. The two handshaking-lines are data valid, which is generated by the source unit, and data-accepted, generated by the destination unit.
* The timing-diagram describes the exchange of signals between the two units. The sequence of events listed in part (c) shows the four possible states that the system can be at any given time.
* The source unit initiates the transfer by placing the data on the bus and enabling its data-valid signal.
* The data-accepted signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its data valid signal, which invalidates the data on the bus.
* The destination unit then disables its data accepted signal and the system goes into its initial state.
* The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.





**Destination initiated transfer using handshaking:**

* The name of the signal generated by the destination unit has been changed to ready for data.
* The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.

**Handshaking advantages:**

* The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units.
* If one unit is faulty, the data transfer will not be completed.
* Such an error can be detected by means of a timeout mechanism, which produces an alarm if the data transfer is not completed within a predetermined time. The timeout is implemented by means of an internal clock time.

**Synchronous vs. Asynchronous:**

* **Synchronous data transfer: sender and receiver use the same clock signal**
  + supports high data transfer rate
  + needs clock signal between the sender and the receiver
  + requires master/slave configuration
* **Asynchronous data transfer: sender provides a synchronization signal to the receiver before starting the transfer of each message**
  + does not need clock signal between the sender and the receiver
  + slower data transfer rate

**DMA:**

**Introduction:**

* Processor, memory and I/O all are connected through system bus.
* We can transfer data between processor and memory or between processor and I/O.
* We can not transfer data between I/O and memory directly. Because memory and I/O cannot generate control signals.
* All control signals are generated by CPU only.
* For example, if you want to print something then first data is transferred from memory to processorand then processor to I/O. So, requires two cycles. If you want to transfer small amount of data nobody cares. But if you transfer large data for example copying into pen drive or printing bulk data requires millions of cycle.
* So, to speedup such big transfer, there is a concept used called DMA. DMA is used to transfer data directly between memory and I/O. DMA controller is used to generate the control signal.

**DMA (Direct Memory Access)**:

* The transfer of data between I/O and memory is often limited by the speed of the CPU.
* Removing CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called Direct Memory Access.
* **Direct memory access** (**DMA**) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main **memory**, by passing the CPU to speed up **memory** operations.
* The process is managed by a chip known as a **DMA** controller (DMAC).
* During DMA transfer, the CPU is idle and has no control of the memory buses.
* A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

**CPU bus signals for DMA transfer:**

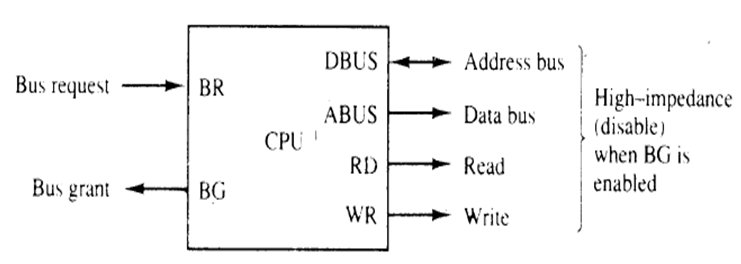
The CPU may be placed in an idle state in a variety of ways. One common method is to disable the buses through special control signals.

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

* **BG (Bus Grant):** the CPU activates the bus grant (BG) to inform the external DMA that the buses are in the high impedance state.

The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

**CPU bus signals for DMA transfer:**

**Types of data transfer:**

* When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways:
* **Burst transfer**: In this transfer, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred.
* **Cycle stealing**: it allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU.

**DMA controller:**

* The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.
* In addition, it needs three registers
  + **Address registers**: It contains an address to specify the desired location in memory.
  + **Word count registers**: It holds the number of words to be transferred. This register is decremented by one after each word transfer and internally tested for zero.
  + **Control register**: It specifies the mode of transfer.

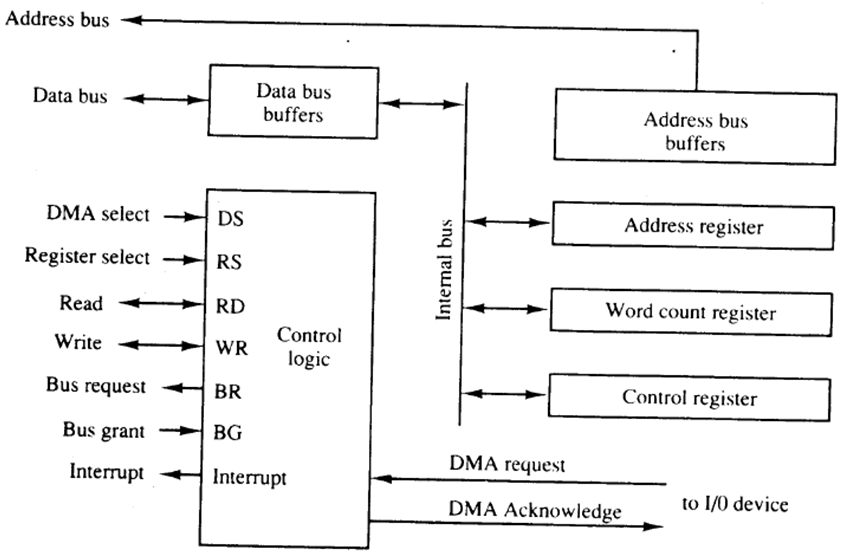
The DMAC unit communicates with the CPU via the data bus and control lines. Control lines are

1. **DS (DMA select) and RS (Register select)**: The registers in the DMA are selected by the CPU through the address bus by enabling the DS and RS inputs.
2. **RD (read) and WR (write):** Both inputs are bidirectional.
3. **BG (Bus Grant):** When BG=0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG=1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.
4. **BR (Bus Request):** This input is used by the DMA controller to request the CPU to give up control of the buses.
5. **Interrupt:** An **interrupt** is a signal to the CPU emitted by DMA indicating an event that needs immediate attention.

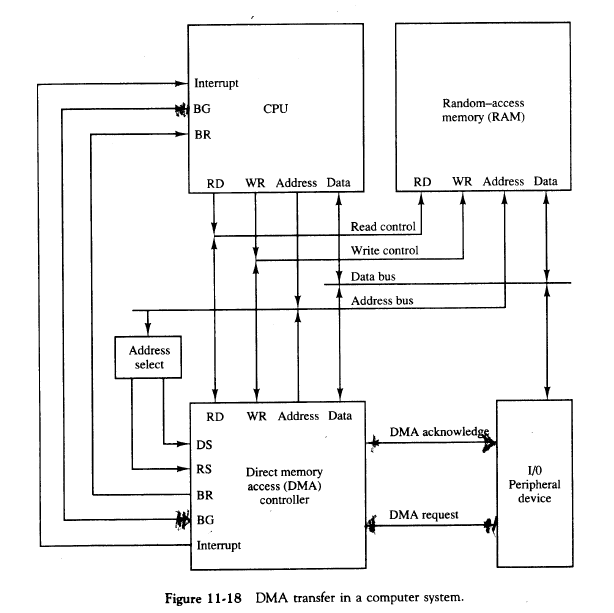
**DMA initialization:**

* The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred. The initialization process is essentially a program consisting of I/O instructions that include the address for selecting particular DMA registers.
* The CPU initializes the DMA by sending the following information through the data bus:
  1. The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
  2. The word count, which is the number of words in the memory block
  3. Control to specify the mode of transfer such as read or write
  4. A control to start the DMA transfer

**Block diagram of DMA controller:**



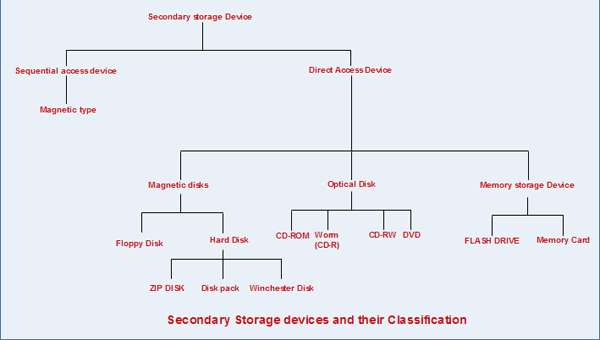
**DMA transfer:**



* The position of the DMA controller among the other components in a computer system is given in the figure.
* The CPU communicates with the DMA through the address and data buses as with any interface unit. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.
* When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses.
* The CPU responds with the its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device.
* When BG=0, the RD or WR are input lines allowing the CPU to communicate with the internal DMA registers.
* When BG=1, the RD or WR are output lines from the DMA controller to the RAM to specify the read or write operation for the data.
* When the peripheral device receives a DMA acknowledge, it puts a word in the data bus or receives a word from the data bus. Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.
* For each word that is transferred, the DMA increments its address registers and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred.
* If the peripheral speed is slower, the DMA request line may come somewhat later. In this case the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.
* If the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by means of a interrupt.

**Peripheral Devices:**

* The input-output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.
* Programs and data must be entered into computer memory for processing and results obtained from computations must be recoded or displayed for the user.
* Input or output devices attached to the computer are also called peripherals. There are three types of peripherals such as
  + Input
  + Output
  + Input-output
* Most common peripherals are keyboards, displays units, and printers.
* Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.



**Magnetic Tape:**

* **Magnetic Tape** can be used to perform both functions -input and output. Magnetic Tapeis a **secondary storage media.**
* Access is sequential and consists of records that can be accessed one after another as the tape moves along a stationary read-write mechanism.
* Magnetic tapes are used for large computers like mainframe computers where large volume of data is stored for a longer time.
* In PC also you can use tapes in the form of cassettes. The cost of storing data in tapes is inexpensive. Tapes consist of magnetic materials that store data permanently.
* It can be 12.5 mm to 25 mm wide plastic film-type and 500 meter to 1200 meter long which is coated with magnetic material.
* The deck is connected to the central processor and information is fed into or read from the tape through the processor. It similar to cassette tape recorder.

**Advantages of Magnetic Tape:**

**Compact:** A 10-inch diameter reel of tape is 2400 feet long and is able to hold 800, 1600 or 6250 characters in each inch of its length. The maximum capacity of such tape is 180 million characters. Thus data are stored much more compactly on tape.

• **Economical**: The cost of storing characters is very less as compared to other storage devices.

• **Fast**: Copying of data is easier and fast.

• **Long term Storage and Re-usability**: Magnetic tapes can be used for long term storage and a tape can be used repeatedly with out loss of data.

**Magnetic disk:**

* Magnetic disks have high speed rotational surfaces coated with magnetic material.
* Access is achieved by moving a read- write mechanism to a track in the magnetized surface.
* Disks are used mostly for bulk storage of programs and data.
* You might have seen the gramophone record, which is circular like a disk and coated with magnetic material.
* Magnetic disks used in computer are made on the same principle. It rotates with very high speed inside the computer drive. Data is stored on both the surface of the disk. Magnetic disks are most popular for direct access storage device.
* Each disk consists of a number of invisible concentric circles called tracks.
* Information is recorded on tracks of a disk surface in the form of tiny magnetic spots.
* The presence of a magnetic spot represents one bit and its absence represents zero bit.
* The information stored in a disk can be read many times without affecting the stored data. So the reading operation is non-destructive.

But if you want to write a new data, then the existing data is erased from the disk and new data is recorded.

**Optical Disk:**

* Information is written to or read from an optical disk or tape using laser beam. Optical disks are not suitable memory storage units because their access time is more than that of hard disks. Their advantage is that they have very high storage capacity.
* Types of optical memory are: CD –[ROM](http://ecomputernotes.com/fundamental/input-output-and-memory/explain-read-only-memory-what-are-the-types-of-rom), CD-R, CD-RW, DVD-ROM, DVD-R and DVD-RW. Information on a CD-ROM is written at the time of manufacture. CD-R/W of 700 MB are available. A DVD-ROM is similar to CD-ROM. It uses shorter wave length of laser beam and hence, stores more data than CD-ROM.
* With every new application and software there is greater demand for memory capacity. It is the necessity to store large volume of data that has led to the development of optical disk storage medium.

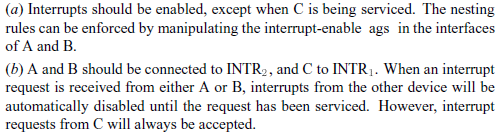
**CDROM systems:**

* CD-ROM disks are made of reflective metals.
* CD-ROM is written during the process of manufacturing by high power laser beam.
* Here the storage density is very high, storage cost is very low and access time is relatively fast.
* Each disk is approximately 4 1/2 inches in diameter and can hold over 600 MB of data.
* As the CD-ROM can be read only we cannot write or make changes into the data contained in it.

**Question 1:**

* Three devices A,B, C are connected to the bus of a computer. I/O transfer for all three devices use interrupt control. Interrupt nesting for devices A and B is not allowed, but interrupt request from C may be accepted while either A or B is being serviced.
* Suggest different ways in which this can be accomplished in each of following cases:
  + The computer has one interrupt request line
  + Two interrupt request lines, INTR1 and INTR2, are available with INTR1 having high priority.
* Specify when and how interrupt are enabled and disabled in each case

**Solution 1:**



**Question 2:**

* A disk unit has 24 recording surface. It has total of 14000 cylinders. There is an average of 400 sectors per track. Each sector contains 512 bytes of data.
  + What is maximum number of bytes that can be stored in this unit?
  + What is the data transfer rate in bytes per second at rotational speed of 7200 rpm?
  + Using a 32 bit word (A32 – A0 ) suggest a suitable scheme for specifying the disk address, assuming that there are 512 bytes per sector.

**Solution 2:**

