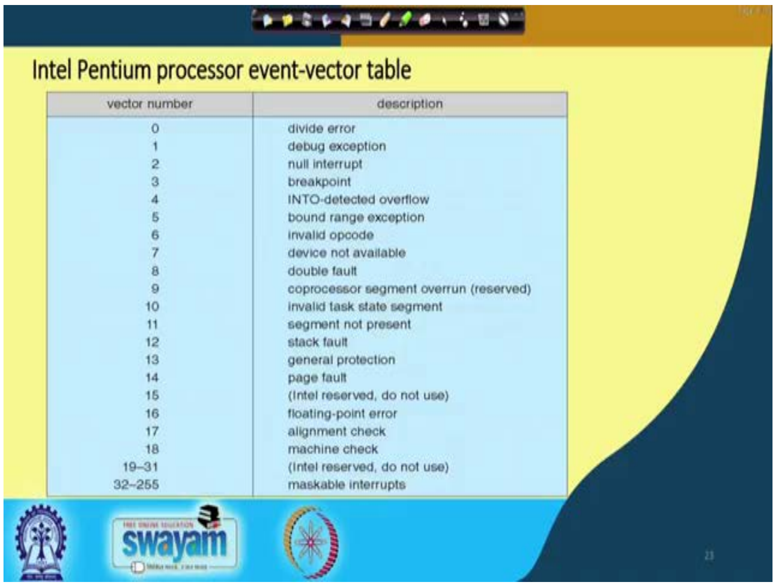
Operating System Fundamentals

Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 04
Introduction (Contd.)

(Refer Slide Time: 00:29)



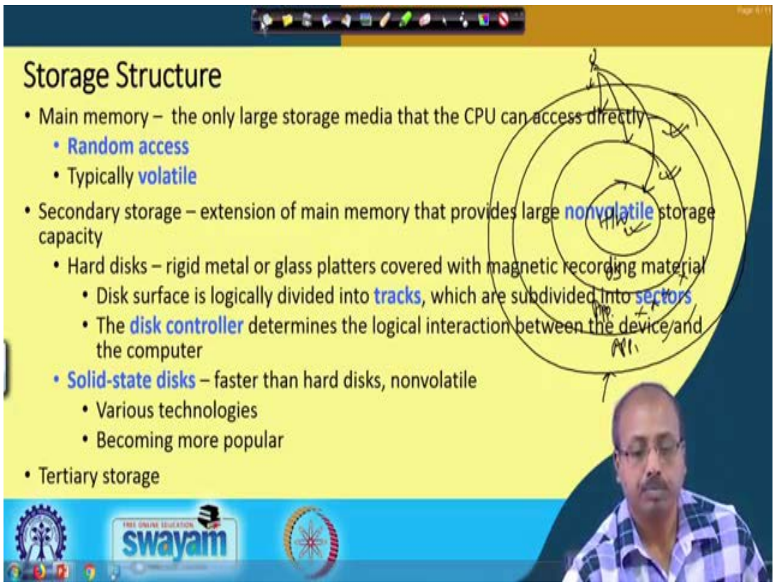
Intel Pentium processor event-vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

So, this is an typical example of this Pentium event vector table for Pentium processor. The purpose is just to tell you like what sort of interrupts may be there or the operating system services may be there. Like this inter vector number 0, it is for the reserved for divide error. So if there is a divide error, then the processor will transfer control to this particular vector and there are various designer. So we have to write the routine corresponding to that in for inter vector 0.

So, for divide by divide error, so this will be the corresponding ISR will be for this is number 0. Similarly, debug exception for one null, so this is a null. So, this is not at use this is reserved actually, some user may use this particular interrupt, then for breakpoint for debugging purpose, so this interrupt is used. In this way you can have a number of interrupts like there are 256 interrupts that I that we have in the system and different interrupts some of them may be used by the system, some of them may be used by the user for providing different services for program development.

(Refer Slide Time: 01:40)



Storage Structure

- Main memory – the only large storage media that the CPU can access directly
  - Random access
  - Typically volatile
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
  - Hard disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into tracks, which are subdivided into sectors
    - The disk controller determines the logical interaction between the device and the computer
  - Solid-state disks – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular
- Tertiary storage

Next, we will be looking into storage structure and storage structure can be broadly divided into 3 categories; Main memory, secondary storage and tertiary storage. In the main memory the so if this is the large storage media that the CPU can access directly. So as we know from our school days that processor can directly talk to the main memory and then, main memory. So, whenever CPU needs any instruction or data, it accesses the main memory and then, the content of the main memory comes to the CPU and CPU tries to understand it and execute it or use it in some instruction.

So this main memory has got the feature that it is random access. So some address is provided and that particular location can be accessed and typically volatile. If you switch off the content of the main memory will be lost. So, main memory is limited in size, though it is large, but it is not that large. So we have got the secondary storage where the extension of main memory that provides large non volatile storage. So, this is very important that it is a non volatile. So, content will not be lost even if you switch off the system.

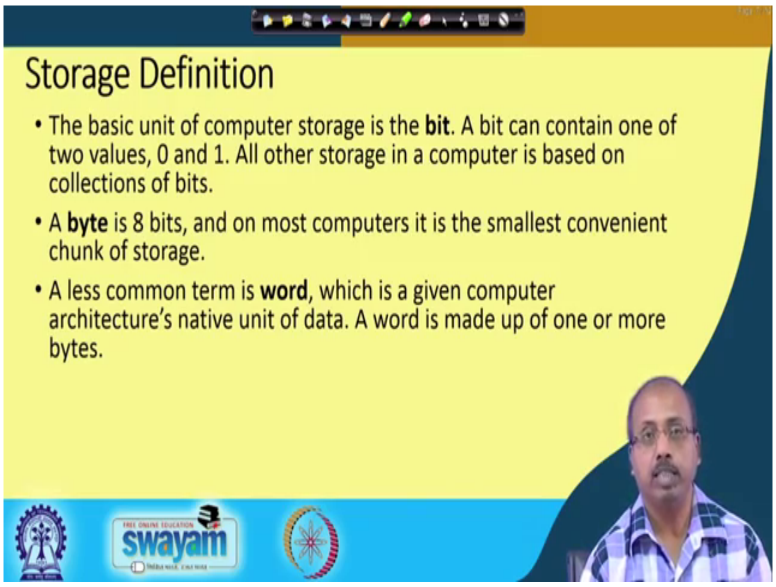
So, typical examples are hard disks, so rigid metal or glass platters covered with magnetic recording material. So this is the structure of the hard disk this surface is logically divided into tracks and which are again divided into sectors. So typically we have got a situation like this. So if this is a disk plate so this is divided into tracks. So we have got several such concentric tracks and these tracks are again divided into sectors.

So, they are divided into sectors like this. So though it appears that these sectors which are inside. So, they are of less size than the outer one, but in reality so they can all hold same amount of data.

And in this part, so each individual part of it is called a sector. So this is a sector and this individual concentric circle so that is a track. A disk controller, it will determine the logical interaction between the device and the computer. So, your individual access is in terms of sectors. So it will be reading from sector and transferring the control to the to the computer. So these are the hard disks that we have.

So next we have got the solid state disks, so they are faster than hard disk, but and non volatile. There are different technologies that are becoming more popular like so optical disk can also they are becoming more popular and there are tertiary storage. So here the storage capacity is even larger, but the access time is slow. So we will be looking into the tertiary storage, so like this tape drives, magnetic tapes particularly they fall into this tertiary storage capacity.

(Refer Slide Time: 04:41)

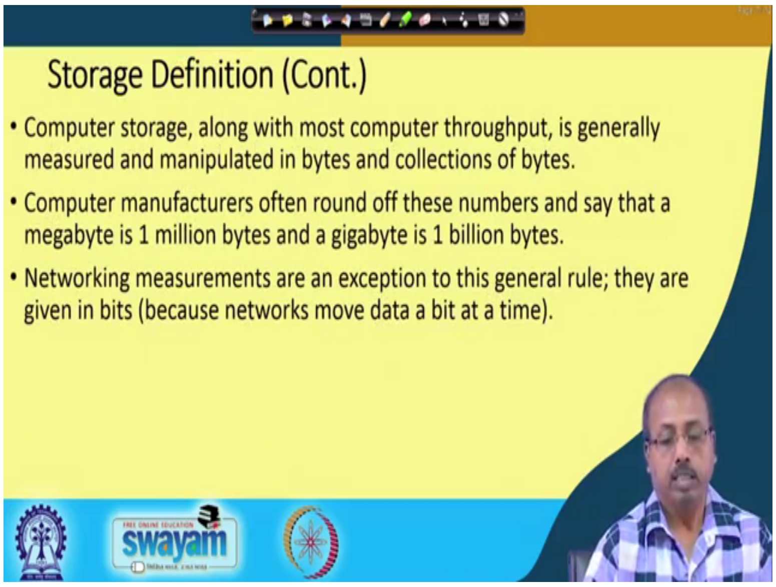


Storage Definition

- The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits.
- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.
- A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.

Definition of storage, so it varies from type of storage that we have. The basic unit of computer storage is bit. A bit can contain one of the two values. 0 and 1 and all other storage in a computer based on collection of bits. A byte is 8 bits and on most computer system it is the smallest convenient chunk of storage. A less common term is word, so this is dependent on the architecture that we have. So there, so word size is variable like a different component systems, different number of bytes can make up word. So there is no fixed size for word it is system depended.

(Refer Slide Time: 05:20)



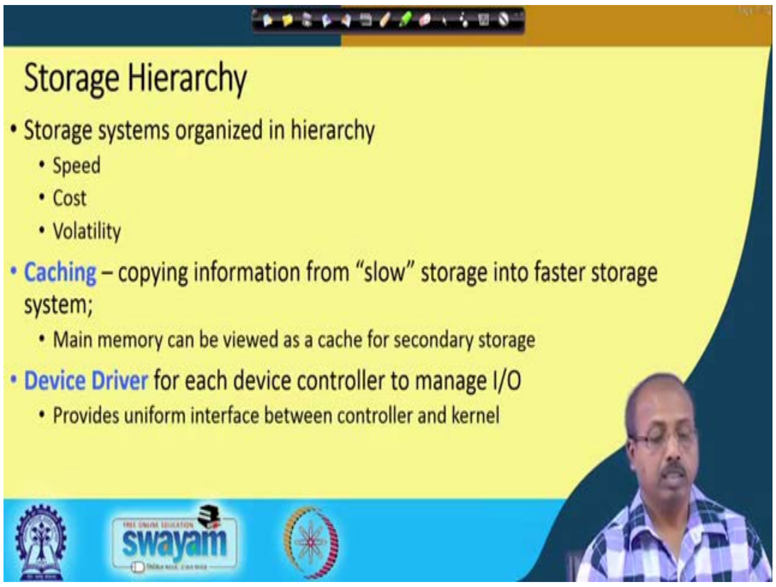
Storage Definition (Cont.)

- Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
- Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes.
- Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Then, almost along with the most computer throughput for the most of the computers a throughput is generally measured in manipulative in bytes and collections of bytes. So, these computer manufacturers often round up these numbers and say that 1 megabyte is 1 million bytes; 1 gigabyte is 1 billion bytes etcetera. But the numbers are slightly different because as you know that it is in terms of powers of 2. So, not powers of 10. So, it is a bit the values may be slightly different.

So, networking measurements are an exception to this general rule where they are given in terms of bits because it how many bits are transmitted per second. So, there it is talking in terms of a bits.

(Refer Slide Time: 06:05)



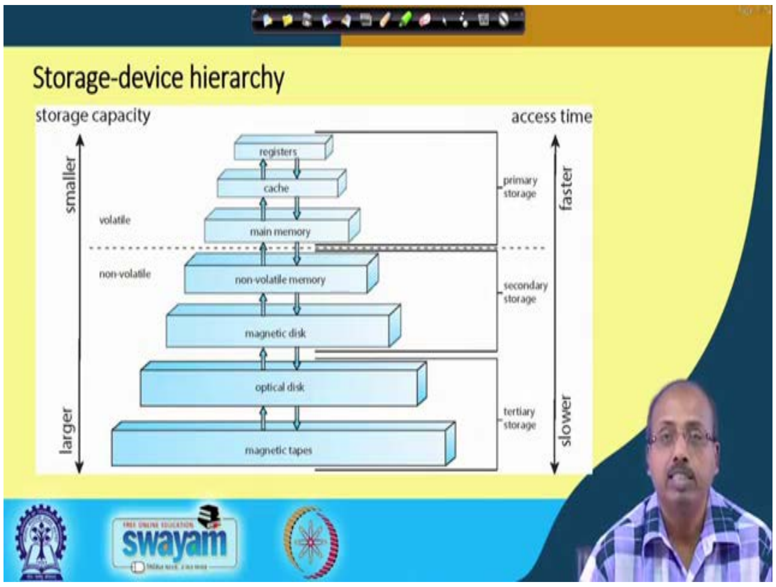
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- Caching – copying information from "slow" storage into faster storage system;
  - Main memory can be viewed as a cache for secondary storage
- Device Driver for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

If we look into the hierarchy of storage, so we have to consider this hierarchy in terms of speed, cost and volatility. So, if you are looking for very high speed device or high speed storage from the CPUs perspective, then they should be located within the CPU. And if you are looking for; naturally CPU is a semiconductor device semiconductor chips. So, if you are putting large storage there, then the cost of the chip will be significantly high. So, what is done is that there is a trade off and we go for some reduction in speed, but we want to increase the size of the system and go to the next level of memory.

And this way starting from these CPU registers if you go to the final tertiary storage, we have different values for the speed cost and volatility figures and mainly, so we will also be using the concept of caching for copying information from slow storage into fast storage.

(Refer Slide Time: 07:12)


Storage-device hierarchy

So, I will come to this concepts slightly later. So, it is like this, so this these are the are the highest level or the topmost level. So, we have got the registers for whom the access time is the minimum and at the same time capacity is also the smallest. So this is for the registers that we have inside the CPU chip. So for today, for the reduced instruction set computers or disk, so we keep a good number of registers, but still the numbers are not that big.

So, you can have say 64 or 128 registers, but not more than that because the cost of the chip will be going up. If you forget about this cache, at the next level we have got main memory. So this main memory the storage capacity is more in than the registers. But speed wise they are a bit slow compare to register because they are off-chip. Registers are within the same chip as CPU, memory is off-chips. So, this off-chip communication is slower than on-chip at least 10 times slower, so there is a gap.

Now, how to reduce this gap the speed gap? So, what has been done? Another type of memory that has been designed is the cache memory which come in between the registers and the main memory, so that there the portions of memory which are frequently referred to by a program. So, they can be kept in the cache. So, that it does not have to come to main memory; every time you want to get the content of those values.

Now, so this cache is faster than main memory because of its access technology that is different from main memory and it has got a parallel axis. So as a result it can make this access timeless compared to main memory. So if you look into any book on computer architecture you can get the details of this cache organization and all. So do not go into that, we just ah keep it in our mind that cache is going to be much faster than main memory and the portions which are frequently referred it may be kept in the cache.

After main memory we have got this non volatile memory. So maybe these are called secondary storage, so we can have magnetic disk. So they are configuring the secondary storage. Below that we have got tertiary storage, where we have got optical disk and magnetic tapes. So if you go in this direction as you are going upper and upper into the pyramid the speed is improving, but their size is decreasing and their cost is also increasing ok.

But if you are going down, so cost per bit if we consider, then the cost per bit will be high as you go towards the upper side of this pyramid and is and as you are coming to the lower side of the pyramid, the cost will be less, cost will be the minimum, but the corresponding time access time will become pretty high. So, this is how this storage device hierarchy is there in a computer system.

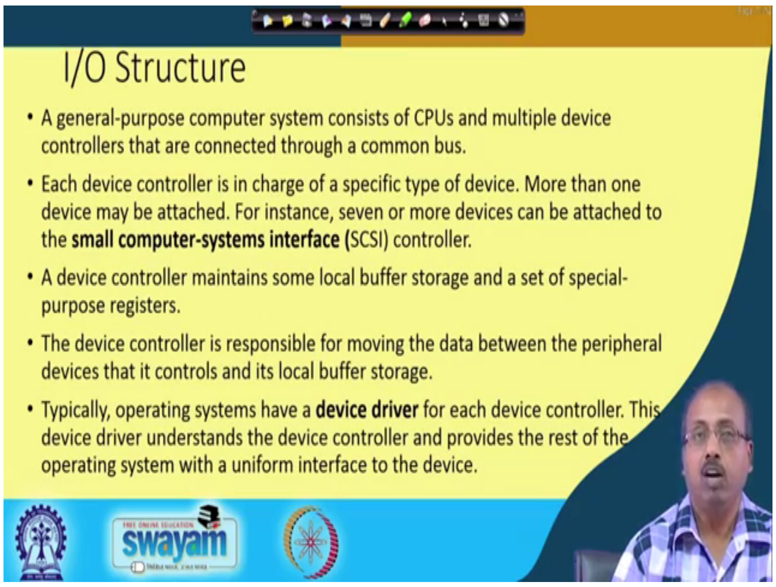
So, we have got this cashing that copies information from slow storage is to in to faster storage system and main memory can be viewed as cache for secondary storage and all. So, that way we have we can think about the cache at between every layers ok, so the some layer which is slow. So, I can take the layer above it as a cache for the lower layer, so that way it can be taken. And we have got device drivers for each device controller to manage the input output operation that provides uniform interface between controller and kernel.

So, as I said that if you look into this diagram, some of these devices are magnetic devices some of them are optical devices, some of they are magnetic devices again magnetic tape device. So, their access patterns and everything is different. So, as far as the operating system is concerned they it should not be made to take care of all of them then the design of the operating system will become very complex. So, that has to be done by the corresponding device drivers. So, whenever if industry is making an organization is making a device, they must provide the corresponding device driver by which that particular device can be operated very easily.

So, we have, so a general way out is to have everything compatible with USB because almost all the operating systems that you have now, so they are USB compatible, they support USB devices and a USB standard, we will be talking about the device classes so, and all. So, the basic functions that should be there, but definitely that is not all because if some device has got some very special features. So, that may not be provided by the device drivers that we have with USB devices.

So, that way for example, a printer may have some special type of printing feature. So that may the USB drive that you have USB device driver that you have that may not support it. So you should not to get the maximum benefit from that printer we should not use the USB device driver. But the device driver that has that has been provided by the device designer has to be used. So the purpose of this device drivers device specific drivers, they cannot be ignored.

(Refer Slide Time: 12:39)



I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device. More than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface** (SCSI) controller.
- A device controller maintains some local buffer storage and a set of special-purpose registers.
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Typically, operating systems have a **device driver** for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

Then, coming to the IO structure, a general purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus. A each device controller is in charge of a specific type of device and more than one device may be may also be attached. For instance, we can have seven or more devices attached to the small computer systems interface or a SCSI control SCSI controller. So, seven or more devices can be connected.

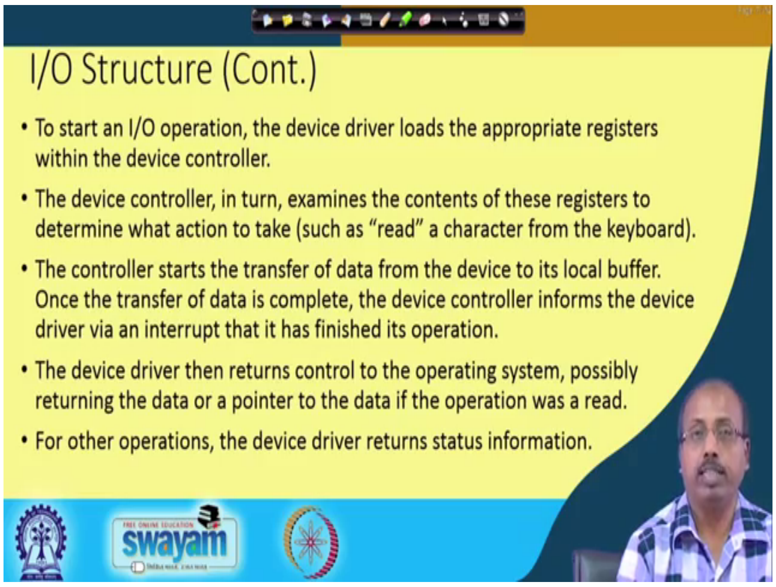
A device controller maintains some local buffer storage and a set of special purpose registers, they are mostly for control register and all. A device controller is responsible for moving the data between the peripheral devices that it controls and its local buffered storage. So this is device controller has to do this. Typically operating systems have a device driver for each device controller.

So, either the operating system will come with a previously designed device driver. For example, the operating systems that support USB devices, they have got USB drivers built in or whenever the device is installed. So that is at that time, the device driver provided by the device designer are getting attached to the operating system.

So after all when the system is operating with the device, at that point of time the device driver is available with the operating system ok. So that whenever this operating system needs to do something with the device, so it can tell the corresponding device driver to provide the facility. So this device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device, so that is very important.

So at a higher level, so it does not matter whether you are writing a file to a disk or you are writing a file to the tape or you are writing a file to the optical drive. Because operators of because they are taken care of by their own device drivers. So they the comments say it will the device driver will issue they will be specific for the corresponding device and the operating system will be relieved from all these details.

(Refer Slide Time: 14:53)



I/O Structure (Cont.)

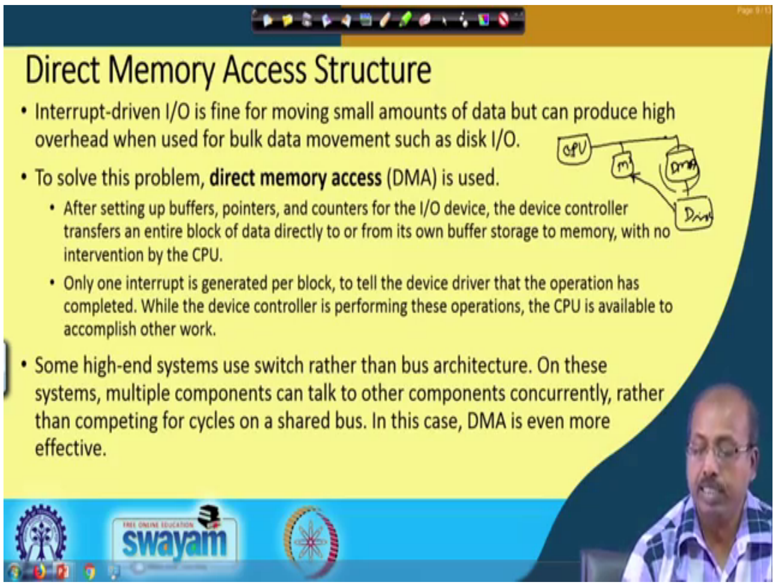
- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as "read" a character from the keyboard).
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.
- For other operations, the device driver returns status information.

So to a start an I O operation, the device driver loads the appropriate registers of the device controller. The device controller in turn will examine the content of these registers to determine what action to take. For example, read the character from the keyboard or write something onto the disk like that. The controller will start the transfer of data from the device to its local buffer and once the transfer of the data is complete, the device controller will inform the device driver via an interrupt that it has finished its operation.

So, to start with the device controllers by registers will be filled up by the device driver and then, the device controller when it finds that the registers have been written with some value it will determine the type of operation to be done. It does that operation with the device and then, it informs the device controller device driver via an interrupt that the operation is over.

Then the device driver will return control to the operating system when possibly returning the data or a pointer to the data, if the operation was a read operation and for other operation the device driver returns status information. So if it is simply a write operation, then it will be returning a status information, if it is a read operation, then it has to give the corresponding value. So, this way this I O operation will continue.

(Refer Slide Time: 16:15)



Direct Memory Access Structure

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, **direct memory access** (DMA) is used.
  - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
  - Only one interrupt is generated per block, to tell the device driver that the operation has completed. While the device controller is performing these operations, the CPU is available to accomplish other work.
- Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rather than competing for cycles on a shared bus. In this case, DMA is even more effective.

So, there is another style of data transfer which is known as Direct Memory Access or DMA. So this interrupt driven I O that we have talked about is fine, but it is for moving small amounts of data. For example, if you are transferring some if you are if your program wants to read something from the keyboard, then the program may inform the keyboard controller via the corresponding device driver that a key has to be read from the keyboard. So when the user presses the key, then the device controller will inform the device driver that a key has been pressed. So, that works well when you have got small amount of data.

But when there is a bulk amount of data that has to be moved. So, I should not interrupt the processor every time 1 byte of data has been transferred. So to solve this problem there is another transfer mechanism which is known as Direct Memory Access or DMA is used. So how does it operate? After setting the setting up buffers, pointers and counters for the I O device the device controller will transfer an entire block of data directly towards from its own buffer storage to memory with no intervention of CPU.

So, what I mean is that we have got the situation like this. So, we have got the CPU and on this the CPU is connected and then the memory chip is there, the memory is there. Then, we have got another module which is known as the DMA controller. And through this DMA controller some device or the disk is connected ok. So when this CPU finds that some amount of data has to be transferred from disk to memory or memory to disk.

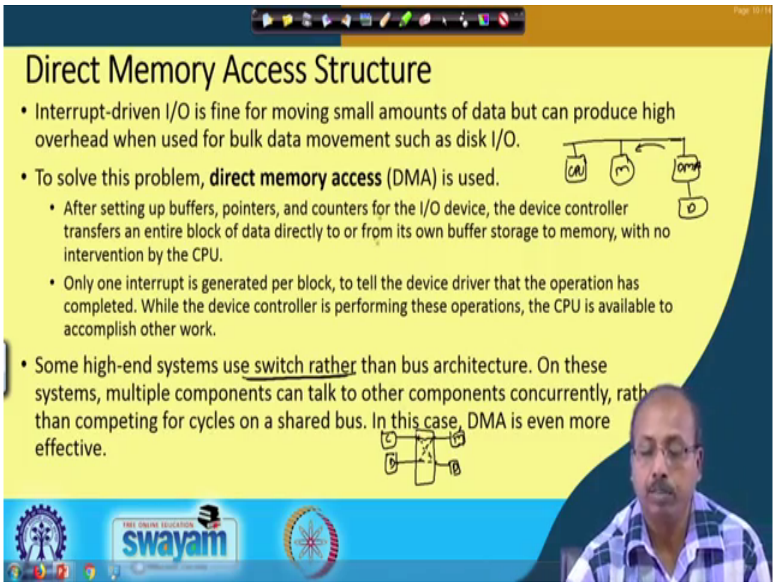
So what it will do, it will inform the DMA controller that some data transfer has to take place between memory and disk.

So what this DMA controller will do? It will be transferring this entire amount of data. So if it is a read operation, it will transfer the entire amount of this data from disk to memory without telling the CPU every time after a byte has been transferred. When this entire block of data which may be 4 kilobyte or 8 kilobyte or 10 kilobyte whatever be my block size, so depending on that when that entire amount of data has been transferred in that, after that only the DMA controller will generate an interrupt to the device driver that the operation is over one block of data has been transferred.

And while the device controller is performing this operation, the CPU can do something else because this transfer will take a good amount of time. Because this the disk that we are having, so this is a magnetic device. So, that takes time to for doing this thing. So, by this time the CPU will be doing something else.

So, CPU is free to do some the to execute some other program that way the CPU can do multi programming and the number of jobs completed per unit time can be improved that is the throughput of the system can be improved a lot. So this is how this direct memory access is going to help. Some high end systems they use switch rather than bus architecture. So some systems what they have is that, so if the type of architecture that we have used here is a bus.

(Refer Slide Time: 19:42)



Direct Memory Access Structure

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, **direct memory access** (DMA) is used.
    - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
    - Only one interrupt is generated per block, to tell the device driver that the operation has completed. While the device controller is performing these operations, the CPU is available to accomplish other work.
- Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rath than competing for cycles on a shared bus. In this case, DMA is even more effective.

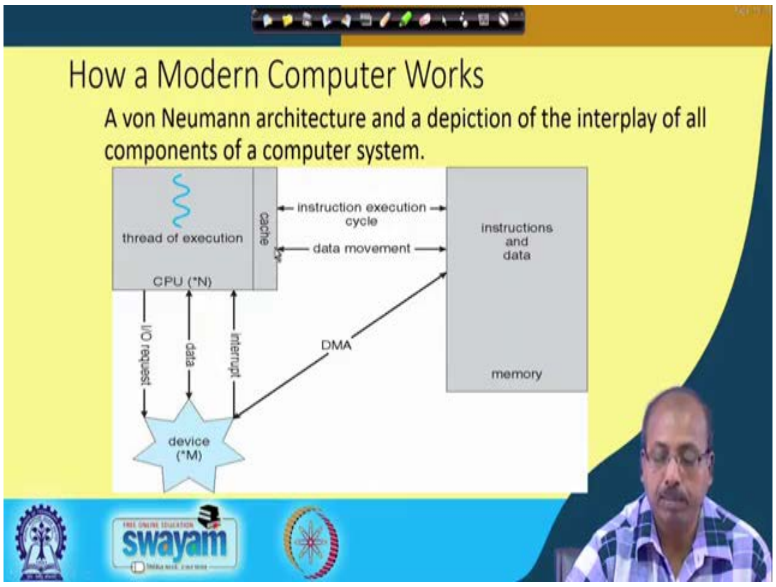
So the problem with the bus is that your, if this is a bus from where the CPU is connected, the memory is connected and my DMA controller is also connected and the devices are connected by through that DMA controller. Now, when this DMA controller is doing the transfer of data from disk to memory, so it is also using this bus and during that time if CPU is executing some program and it needs to access the content of the memory, so it becomes difficult.

So, now the CPU and DMA they start competing for the bus access and that restrict the system performance. Normally the strategy that is followed is thus DMA is given lower preference than CPU and whenever the CPU is not using the bus, then only the DMA controller will be using the bus for the transfer. So this is called something called cycle stealing and all that. However, some systems to improve the performance further, so we have got a some sort of switch connection.

So, this switch based connection. So, there we have got an interconnection switch ok, on one side, we have got all these masters like this is the CPU, this is the DMA controller like that and then other side we have got this memory we have got this disk like this and there are these switches there as there is a interconnection switch and these switches can be connected in different fashions. And depending upon the number of layers that you have in this switch, so we can establish different types of connections and we want to make multiple components talk to other components concurrently rather than competing for cycles on a shared bus.

So, like previous situation, it they were competing for getting free cycles, but in these case they do not compete for the free cycles. They just they may be doing parallel operation and doing the operation. So, DMA becomes more effective because now there the CPU is free to do some other operation rather than competing for the bus with the DMA controller.

(Refer Slide Time: 22:04)


How a Modern Computer Works

A von Neumann architecture and a depiction of the interplay of all components of a computer system.

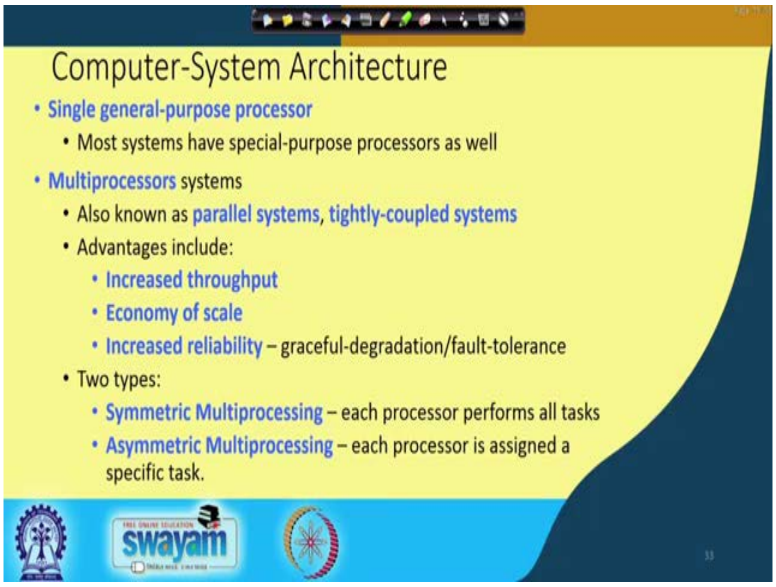
So this way DMA can help us to improve performance. So, how does this modern computers work a von Neumann architecture and a depiction of the interplay of all computer components or a computer system, so it is like this. So, CPU; so there may be several threads of execution. So, there may be several processes. So, we will come to the concept of thread later, but if there may be a process may be divided into a threads also, so we will come to that later. So, for the time being we take thread and process or program same that the some execution something that has to be executed by the CPU.

So, the CPU it may be executing N different processes simultaneously. In the sense that it does one of them at a time and if that gets suspended due to some operation or the other, it takes up the next operation next process that way it does n processes at a time simultaneously. Now, from the main memory, so it can get instruction, execution that can continue and some data movement can take place. At some point of time there may be some data movement may be necessary.

So, CPU needs some data to be transferred from device. So, it sends an I O request and as a result it initiates some DMA transfer through some either it may be a DMA transfer, where the data is transferred to the memory and from there the CPU need reads it again or it may be via an interrupt like it is sending an I O request, the device puts the data and then it sends an interrupt to the CPU telling that the data is ready.

So, that way or if it is writing something on to the device, then the CPU may put the data on to on this line and send an I O request output request here and this the device when the operation is over, it can send an interrupt telling that the operation is over, the output operation is over. So, this way this interrupt driven I O can take place and we have also discussed the direct memory access base transfer, where the transfer is made from the device to the memory directly without intervention of the CPU. So, that can be done, so that is the other way in which they can execute.

(Refer Slide Time: 24:18)



Computer-System Architecture

- Single general-purpose processor
  - Most systems have special-purpose processors as well
- Multiprocessors systems
  - Also known as parallel systems, tightly-coupled systems
  - Advantages include:
    - Increased throughput
    - Economy of scale
    - Increased reliability – graceful-degradation/fault-tolerance
  - Two types:
    - Symmetric Multiprocessing – each processor performs all tasks
    - Asymmetric Multiprocessing – each processor is assigned a specific task.

Looking into computer system architectures, so single general purpose processor, so this is the most simple type of operation that we have. So we have got a single processor and so that is executing the program. So only one program can be executed at a time; then, we can have multiprocessor systems also known as parallel systems or tightly coupled systems. So multiprocessor means we have got multiple processors. So, as a result they can be made to operate many jobs at parallelly. So, we can have many operations done together.

So, multiprocessor systems they have got the advantage of this increased throughput, economy of scale and increased reliability. So, economy of scale means so if I have got 2 processors, then I can do 2 programs simultaneously. So, if I have got say 10 processors, I can do 10 programs simultaneously. So, if the programs are totally independent, then this scalability is pretty high. If the programs are dependent, if output of one program will control the act as a input to other.

So, you may not get that tenfold in increase in the speed in the throughput. But they are at least some level of improvement will come, so that is that is the scale factor. So, that scale factor determines how much improvement in the speed can be achieved, if you increase the operation; if you increase the number of processors by some factor. Then, the reliability can be improved, so either it should do a graceful degradation or for tolerance.

Like a multiple processors are there, so if one processor become faulty the job that this processor was doing may be taken up by some other processor. Though, the overall system throughput will become less, but the system will not fail. Compared to the single processor single general purpose processor, if that processor fails, then the entire system fails. So, in a multiprocessor system we have got a graceful degradation or fault tolerance.

Again, this multiprocessor systems they can be divided into two classes; one is called symmetric multiprocessing, another is called asymmetric multiprocessing. In symmetric multiprocessing, each processor is equally capable of doing the jobs. So, all tasks can be done by every processor. So, that way as a graceful degradation that part becomes easy.

So, you can if one processor has failed, so it the job can be taken up by any other processor. Whereas, for asymmetric multiprocessing, so each processor is assigned some specific task only and depending on the type of the processor and the type of the task, so one task may not be able to be a carried out by say all the processors. Only maybe a subset of processors can do this.

So, here also I can have graceful degradation, but to the extent that a similar processor has to take up the job. So, both of them have got their advantages and disadvantages because symmetric multiprocessing means there the processors are not specialized for some operation whereas, asymmetric multiprocessing their processors are specialized for something.

For example, in a multi processing system; some of the processors may be general purpose processor; some of the processors may be signal processors. So, signal processing jobs can better be done by those their signal processing the signal processors. So, like that we can have different type of computer system architecture and that will determine the design of the operating system to a great extent. So, we look that in subsequent classes as we go into the further chapters on the operating system.