

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

**Lecture – 05**  
**Introduction (Contd.)**

Operating system design is often influenced by the underlying computer system architecture. So, architecture may have some features we want to exploit them for proper benefit from the system.

(Refer Slide Time: 00:40)

**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.

The diagram shows four boxes representing processors connected to a central box labeled 'Mgmt.'.

So, if the operating system does not exploit those features then of course, as an end user we cannot get facilities from that architecture. So, if you look into the computer system architecture so, they can broadly be classified into a few classes.

The first class is the single general purpose processor, so this is the most common version that we have. So, there is a single processor or single CPU and then we have got the operating system that uses that particular CPU features in terms of whether it supports a memory management whether it supports protection. So, advanced processor architecture so, they will support different features. So, operating system designer may like to exploit those features and some processors may have a number of cores.

So, we will come to that later when we discuss about the threading of a processes and all. So, basically within the same CPU so, we can have a number of cores which can do some jobs which are dependent on each other at the same time they can be made to run parallelly because there is some computational independence also between them.

So, next category of systems that we have so, they are multiprocessor system so, you have got multiple processors in the system. So, they are also known as parallel systems because they need I can do several tasks parallelly there and they are called tightly coupled system because they are part of the same system. So, what happens is that in this system. So, we have got a number of processors like this, but they are using some common memory by which the from which they get the program and data. So, they all access this common memory for getting the program and data.

So, that is a bit easy to implement because of the fact that. So, whenever there is some sharing considered between these processors so, they are ultimately accessing some memory location. So, if a particular variable  $x$  has to be shared between two programs running on two different processors. So, maybe the first processor writes a value of  $x$  at a memory location and the second processor reads it from that memory location. So, then that way we can have the same the variable values shared. So, that is why it is called a tightly coupled systems as if the all the processors are very much related to each other.

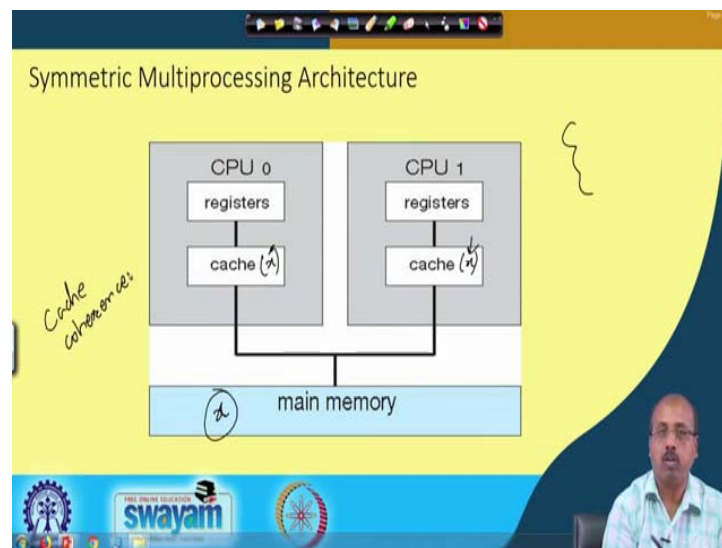
So, the advantages that we have is that increased throughput, economy of scale and increased reliability. So, increase throughput because throughput actually measures the number of jobs that we can do per unit time. So, since we have got multiple number of processors per unit time the system can compute more number of jobs. Economy of scale so, as you are increasing the number of processors then what is the rate at which your computational computation increases or the throughput increases.

So, what is the rate at which it increases ideally I should get if I increase the number of processors from 1 to  $n$  I would expect that the throughput will also increase by a factor of  $n$ , but that may or may not happen because ultimately when a number of processors are trying to access some memory. So, there will be contention in the memory access so, that way it may not be possible also. And reliability is improved definitely because if one processor fails some other processor can take up the job. So, that way it can be so, the reliability can be improved.

Then there are 2 types of multi processors; one is symmetric multiprocessing multiprocessor, where each processor they perform can perform all tasks. So, if I have to design a scheduler for the this system. So, I do not have to think like which processor a particular task we put into. So that way I have the flexibility there, but if it is asymmetric multiprocessing then each processor may be of different type maybe in my system there are say 4 CPU's and 4 DSP processors.

So, for signal processing jobs so, I would like to put it on the DSP processors or and for or general computation I will like to put on the CPU's so, that way this type of asymmetric multiprocessing may come and if the underlying platform if it is symmetric then the OS design will be of some type. If underlying design is something else the architecture is something else then the OS designer has to do the module development accordingly. So, this is the thing that I wanted to emphasize on this architecture side.

(Refer Slide Time: 05:34)



Next will look into. So, this symmetric multiprocessing so, this is the architecture like suppose I have got 2 CPU's, CPU 0 and CPU 1 each CPU has got it is own registers. It has got it is own cache memory, but whenever it is the main memory is common so, that way they are whenever some variable value is required so, they have to access this main memory. However, they have got their own cache memories so, that for repeated access, they need not go to the main memory so, they can get it from the cache itself.

Of course there are problems with this type of caches because as a variables values across the caches they should be uniform. So, it cannot be that case that there is a variable say x in my program so, this x so, this fellow has got a local copy of x and this fellow has got a local copy of x. So, in that situation it is difficult because when this CPU is modifying so, it will modify it is local copy. So, this fellow will also modify it is local copy, it will not getting reflected in the global cache. So, there is an important problem which is known as the cache coherence problem.

So, this actually addresses this issue that the value of x across all the places that is same. So, if you look into some classes on architecture so, you will find the details of about all these things. So, we do not go into that, but as an OS designer we need to solve this cache coherence problem also.

(Refer Slide Time: 07:06)

**Multicore Systems**

- Most CPU design now includes multiple computing cores on a single chip. Such multiprocessor systems are termed **multicore**.
- Multicore systems can be more efficient than multiple chips with single cores because:
  - On-chip communication is faster than between-chip communication.
  - One chip with multiple cores uses significantly less power than multiple single-core chips, an important issue for laptops as well as mobile devices.
- Note -- while multicore systems are multiprocessor systems, not all multiprocessor systems are multicore.

Then other possibility is the multi core system so, CPU design that we have now. So, they have got multiple computing cores on a single chip. So, they are called this type of multiprocessor. So, they are termed as multi core system now it is very common to have say 4 to 8 cores on this the standard CPU chips that are coming out.

So, multi core systems can be more efficient than multiple chips with single cores because of this reason. So, what happens is that in the multiprocessor system that we have looked into previously so, we have got separate chips. So, this is the CPU 0, this is

a CPU 1 so, like that we have got a number of such CPU chips. So, they are individual chips and the memory also happens to be another chip.

Now whenever this CPU wants to talk to this memory so, it has to go for an off chip communication so, that way all these communications that are happening so, they are off chip. And as we know that if you compare between off chip and on chip communication so on chip communications are much faster than the off chip communication. So, if you look into any computer system so, if you open that system box so, you will find a number of printed circuit boards or PCB's on to which these chips are all mounted.

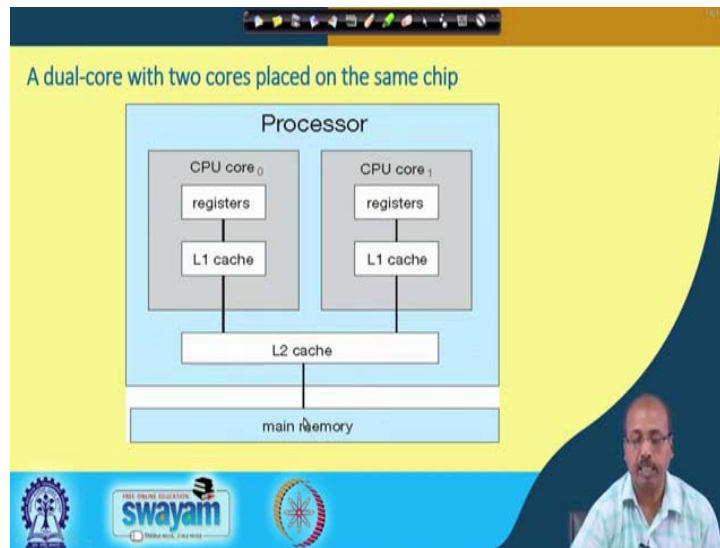
So, now if I so, there are off chip couple line connections between them. So, if I want to increase the speed further was what I need to do is that I should fabricate this whole thing onto a single silicon wafer. So, that way all the off chip communications that we have so, they will become on chip communication. So, that is so that way if I can have multiple such processors built onto a single chip so that particular system so, they will be called multi core system.

So, there are several names for that like System on Chip SoC Multi Processor System on Chip or MPSoC so, like that there are various names. But ultimately what they mean is that on the on a single silicon we have got multiple such computational platforms and the memory can also be a part of it.

So, one chip with multiple cores it uses significantly less power than multiple single core chips, because of the reason that this extra power that is consumed by these buses that we have externally so, that will be taking the power high. So, this is an important issue for laptops as well as mobile devices because the overall footprint of the system will become small. So, my mobile phone size can go down and but at the same time it is computational power does not decrease that way. So, that rather it increases so, if I have got multiple cores so, several operations can be done simultaneously.

So, multi core systems are multiprocessor systems, but all multiprocessor systems are not multi core, because multi core means within a single chip all the processors are housed. But in case of multi processor so, it may be the case that we have got separate chips for separate processors.

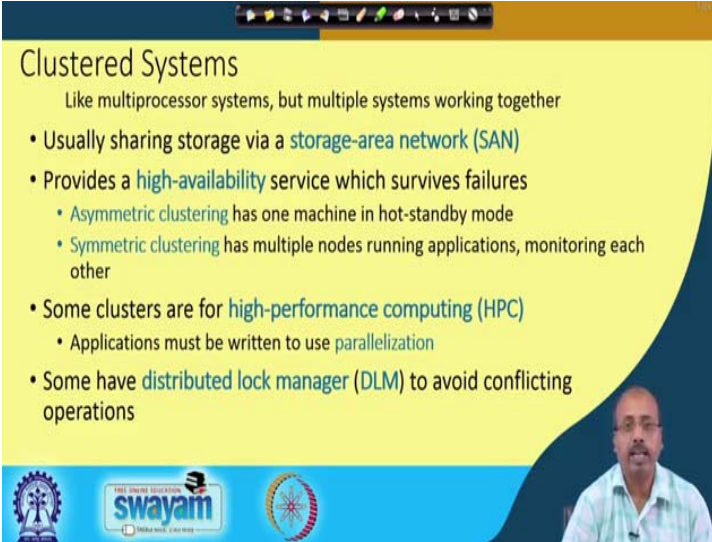
(Refer Slide Time: 10:14)



So, a dual core system there is an example is given here. So, we have got core 0 and core 1. So, we have got L1 caches. So, there are 2 levels of caches that we have. So, this L1 cache and L2 cache so, these two cores are placed on the same chip so, core 0 and core 1. So, each core has got its own L1 cache and there is an L2 cache. So, that is also on chip, but this is on the processor chip. So, for the entire processor we have got a single L2 cache, but for each core it has got an L1 cache. So, some content from main memory can be loaded onto L2 cache from L2 cache something which is relevant for core 0 may be copied into L1 cache and something relevant for core 1 may be copied into L1 cache of core 1.

So, this way we can design a system. So, when you are designing the operating system so, we have to know what is the underlying architecture and accordingly we have to do this design.

(Refer Slide Time: 11:18)



**Clustered Systems**  
Like multiprocessor systems, but multiple systems working together

- Usually sharing storage via a **storage-area network (SAN)**
- Provides a **high-availability** service which survives failures
  - **Asymmetric clustering** has one machine in hot-standby mode
  - **Symmetric clustering** has multiple nodes running applications, monitoring each other
- Some clusters are for **high-performance computing (HPC)**
  - Applications must be written to use **parallelization**
- Some have **distributed lock manager (DLM)** to avoid conflicting operations

The slide features a yellow background with a dark blue wave-like shape on the right side. At the bottom, there are logos for 'swayam' and other educational institutions, along with a small video inset of a man speaking.

Another type of systems that we have so, they are known as clustered systems, they are like multiprocessor systems, but multiple systems they are worked together. So, normally they share storage by a storage area networks and so, as you know in a computer system one part of it is the computation another part is the storage. So, accessing some variables or some data which is stored in the secondary storage is a big issue and if it is a large data size that we have lot lots of data that we have so, and that has to be shared across the systems.

So, it is important that we store it on some separate servers and that forms it is own network so, it is called storage area network or SAN. So, if we have that so they provides high availability service which survives failures like, if one computer system fails then that storage area network will not fail as a result we can get the copy of the data kept in some other server on that storage area network and can get the data from there so, that way availability of the system will increase.

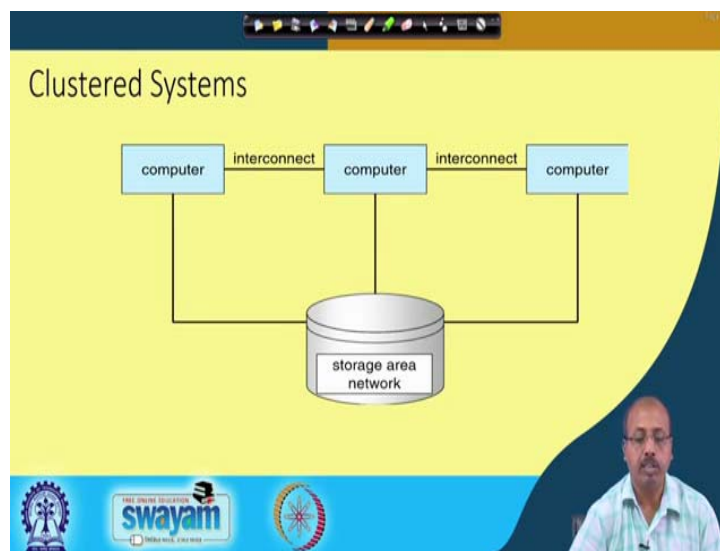
We can have asymmetric clustering so, such in which one machine is in a hot standby mode. So, hot standby mode means, it is always doing whatever the master is doing, in case master fails it can take up immediately so, that is the concept of hot standby. And we can have symmetric clustering that has got multiple nodes running applications I mean they monitor each other. So, they are they actually check what other fellow has

computed and then tries to compare between those values and that come to a decision like which one is correct.

Some clusters are for high performance computing so, they are for example, so, weather forecasting and all where amount of computation is very high and we need to do it very fast. So, some sort of clustering is there for high performance computing. Then we have got in that case the application must be written so, that the parallelism is exploited. So, own a high performance system. So, if I write a program or I develop a system that uses only sequential computation then naturally it will not be getting the benefits of the performance computing. So, we have to do some parallel programming there.

And some have Distributed Lock Manage or DLM to avoid conflicting operations, like two or two or more processors they are trying to modify the same data item. So, that way we have to have some locking facility. So, that this is particularly true if you have got this banked and banking system and all. So, whenever one account is being modified by a transaction so, no other transaction should be allowed to modify that account. So, there has to be some explicit lock put into the system and that lock will mean that this second transaction is stopped from accessing the account when in the first transaction doing it. So, this can be the thing.

(Refer Slide Time: 14:27)

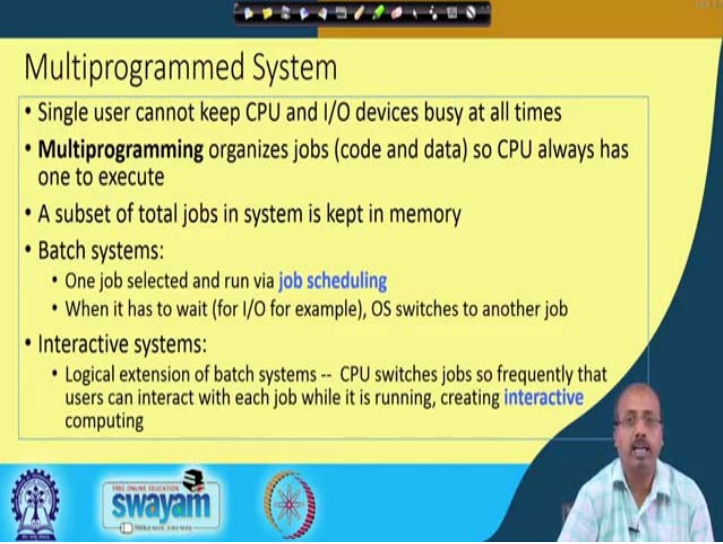


So, this is the clustered system that we have got a number of computers they are interconnected in some fashion and there is a storage area network. So, storage area



network again consists of a set of servers and these computers they can send messages to the storage area network asking for some data item. The network will determine which server it needs to access to get the corresponding data item and then send it back to the requesting computer. So, this is the clustered system.

(Refer Slide Time: 14:59)



**Multiprogrammed System**

- Single user cannot keep CPU and I/O devices busy at all times
- **Multiprogramming** organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- Batch systems:
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- Interactive systems:
  - Logical extension of batch systems -- CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive computing**

Another very important class of systems they are known as multi programmed system. So, multi program this term comes from the fact when we have got multiple users for the system or multiple programs ok. So, this term came from the multiple program. So, if I normally one user is running one program so, in that sense we have got multiple user system. So, in general what happens is that a single user cannot CPU cannot keep the CPU and IO devices busy at all times, because for keeping the CPU busy I have to give it a computational job and when it is doing the computational job. So, in between it has to get data the user has to enter some data.

For example, if you are computing roots of a quadratic equation so, you need to feed in the values of a b and c and normally this human being and these mechanical devices are much much slower compared to the electronic CPU and memory. So, naturally this process the in the process the CPU will be kept waiting when this IO is going on and when the CPU is doing the computation. So, we have user will possibly not enter something into the system by doing some IO operation.

For a single user it is very difficult to do parallel processing and keep the CPU and IO devices busy equally, but what we want is 100 percent utilization of all the system resources the CPU and IO devices. So, multi programming it organizes jobs that is code and data. So, CPU always has want to execute. So, if there are multiple users they have submitted a number of programs, then when one user is not giving the data or doing the IO operation then another users program can be taken up by the CPU for execution. Similarly, if one user is not using the IO device then some other users IO operation may be carried out at that time.

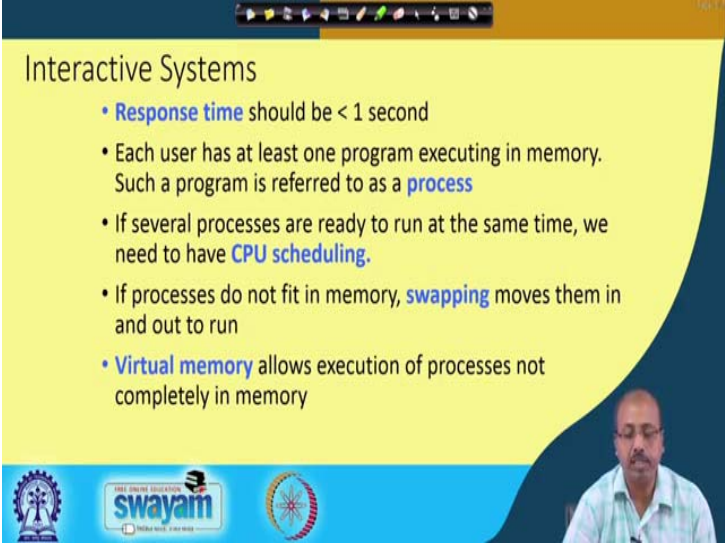
So, a subset of total jobs in the system is kept in memory and there are 2 types of systems; one is called batch system, another is called interactive system. In a batch system one job is selected and run via job scheduling. So, in a batch system what happens is that we submit a batch of jobs a set of jobs to the for to the computer for execution and the scheduler put them in a scheduling queue and one after the other the jobs will be given to the system and when it has to wait? So, OS will switch to another job so, a batch of jobs are given and one by one they have process.

So, it is like say a number of students who have submitted their jobs they are developed the program and they have given their jobs for execution. So, the computer takes up the jobs one by one and go for execution and when one job is going for IO operation another one is doing. So, this is so, there is no direct interaction. So, actually unlike say when unlike other type of examples that we took like quadratic a solving quadratic equation the values of a b c are to be input. So, in batch jobs there is no interaction with the user so, the user will give the program code and data together and this whole thing will be given to the computer for execution. So, it does not ask for data while executing the program. So, it takes from the input that is given at the beginning.

So, if we want to have interactions, then we have got interactive systems. So, this is a logical extension of batch systems because CPU switches jobs so frequently that users can interact with each job while it is running and creating an interactive computing environment. So, one job one user has submitted a job and then giving the values of the inputs that the program needs, at that time some other problem is executing and when the other program needs value again the user is prompted to enter values. Now since if the electronic system is very fast so, users will not understand that my program was not executed for some amount of time. So, that way it remains interactive in nature. So,

system is an interactive system so, we have got batch systems, we have got interactive systems.

(Refer Slide Time: 19:09)



The slide is titled "Interactive Systems" and features a yellow background with a dark blue curved border on the right side. It contains a list of five bullet points. At the bottom of the slide, there is a video inset showing a man speaking. The slide also includes logos for "swayam" and other educational institutions.

- **Response time** should be < 1 second
- Each user has at least one program executing in memory. Such a program is referred to as a **process**
- If several processes are ready to run at the same time, we need to have **CPU scheduling**.
- If processes do not fit in memory, **swapping** moves them in and out to run
- **Virtual memory** allows execution of processes not completely in memory

In case of interactive system the response time should be less than 1 second that is natural human perception. So, if I do not get a response from a computer for 1 second then I will take that the system is pretty slow. So, response time is the time by which the computer comes up with the answer it should be less until and unless it is doing some computation of course, so, it is doing. So, at that time user is ready to accept that delay, but when the program has just started executing if at the very beginning there is a welcome message to be flashed and the user has written the program in such a fashion that it first that flashes some messages at the beginning and if that message does not come immediately after the program has been started so, we will say that the system is slow.

So, that response time should be less than 1 second and each user has at least 1 program executing in memory such a program is we will call it a process. So, for every user at least 1 program must be there in the memory. So, will see later that it is called a process we look into more detailed description about the process, because this is the heart of the operating system the concept of process.

If several processes are ready to run at the same time we need to have CPU scheduling. So, here a CPU scheduling means that how the CPU is given to individual processes. So,

there are a number of ready processes, but since if I have got a single processor system then the CPU can do only one job at a time now how which job it will take so, it is decided by the scheduling policy.

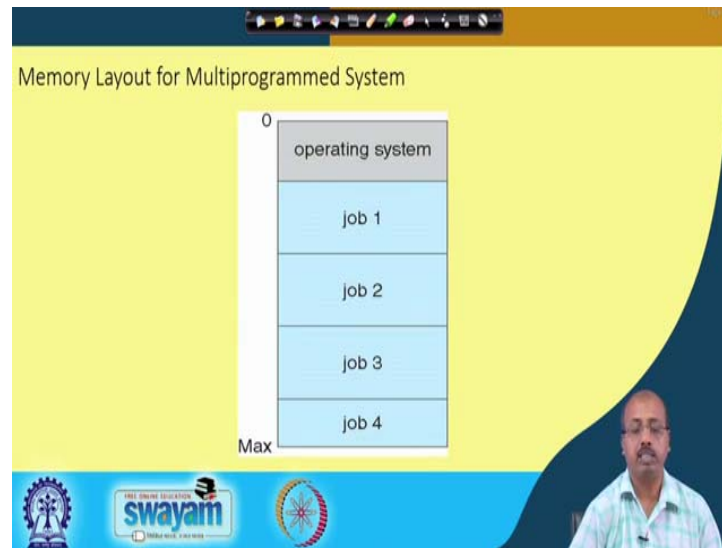
Similarly, if we have got multiple processors also then which jobs are picked up by the processors. So, definitely number of processors that we have is much less than the number of jobs that we have in the system. So, not all of them cannot be executed simultaneously all of them cannot be scheduled simultaneously, only a part of only a subset of them can be done. So, which subset of jobs should be executed at any point of time so, that is decided by the scheduling policy.

If processes do not fit in memory so, this is the other concern. So, it may so, happen that I have got say 100 different users so 100 processes must be there in the memory, but each user program has got some size. So, that way 100 programs may not be fitting into the system. So, that way we need to do something so, that the only a subset of processes are kept or for every process only we have got a few part, a small part of it available in the memory and later on when we need further parts. So, the part that was under use so, that has to be taken out of memory that is called swapping and that has to as the portion that we need to refer to now is taken in.

So, this is this process is called the swap out swap in process. So, that is very important to accommodate large programs into memory, because we cannot have a memory to achieve to be of infinite size. So, there will be limitation and this swap in swap out concept helps us in making the program logically infinite in size.

So, we have got virtual memory. So, this is another concept that we will discuss in the context of this operating system which means that the memory from the users viewpoint so, it becomes so huge that this is infinite. So, users will often consider it to be infinite, but of course, infinite memory is not possible. So, what is done internally is that a part of the disk. So, it is taken as it is also taken as a memory and since this space is very huge. So, we can say that this total memory that we have; that we have in the system is infinite. So, that is the concept of virtual memory we will come to that when we go into this virtual memory discussion.

(Refer Slide Time: 23:07)



So, on the computer system so, you can see if this is the total memory that we have a part of it is reserved for the operating system and after that we have got different jobs at different parts of the operating system.

So, this region we have got job 1, then it has job 2, job 3, job 4 like that. So, in this way in this particular case some memory is divided into a number of parts and in each part we are putting one particular job. So, this is a very simplistic type of approach that we have shown here. So, we will see that there are much more complex approaches and then we have got different situations at which by which these jobs may be done.

(Refer Slide Time: 23:52)

**Modes of Operation**

- A mechanism that allows the OS to protect itself and other system components
- Two modes:
  - User mode
  - Kernel mode
- Mode bit (0 or 1) provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - Systems call by a user asking the OS to perform some function changes from user mode to kernel mode.
  - Return from a system call resets the mode to user mode.

Handwritten notes on the slide:  
- A box with 'User mode' and 'Kernel mode' and an arrow pointing from User to Kernel.  
- Code:  $a=y+z$ ,  $p=x*x$ . A note says 'Resource access kernel mode' with an arrow pointing to the second line of code.

Now, if you look into the modes of operation of an operating system. So, this there is a mechanism that allows the OS to protect itself and other system components. So, there this is known as user mode and kernel mode. So, there is a mode bit provided by the hardware designer. So, if you look into any processor design so, there will be in the processor status register so, there will be a mode bit.

So, if it is one very simple type of implementation may be that if the mode bit is 0 so, it may be do executing the program in user mode. If the mode bit is 1 it is using the program it is executing in the kernel mode. So, what is the difference between this user mode and kernel mode is that. So, kernel mode execution is much more protected. So, there may be certain portion of memory so, which can be accessed only in the kernel mode not in the user mode.

There may be certain set of instructions that can be executed only in the kernel mode and not in the user mode. So, as a processor designer the processor designer group so, they have they provide these features like they like depend taking input from the OS group like what are the different special operations that you need to do in a protective in a protected environment. So, based on that all those operations so, they are put on to the kernel mode. So, there may be some important data structures that we do not want to modify so, they that modification will be put into the kernel mode.

So, in naturally in many of the operating system design so, what we do is that say suppose I have written a program which is doing this computation. So,  $x = y + z$ ,  $p = x \times k$  so, like that as long as I am doing these simple computations. So, this does not make any problem. So, it is simply going to these instructions are addition, multiplication operation so, they are going on like that, but as soon as I say that there is a print statement say print the value of  $x$ , now at this point I am trying to access a system resource.

Now, this is very critical because this print  $x$ . So, whether the printer is available with my program or not. So, that is an important decision may be some other program is using the printer. So, I should not be allowed to access the printer at this point of time. So, somehow this check has to be done, how do we ensure that, for ensuring it so, what is done is that whenever there is a resource access; whenever there is a system resource access so, this is done in the kernel mode. So, the thus program goes from user mode execution to a kernel mode execution. So, as if you can formally you can conceptually view it like this as if this entire operating system it is divided into two parts; one part is the user part and the other part is the kernel part.

So, when a program is running normally doing a computations and also they are are running in the user mode as soon as it needs a resource system resource for it is further execution. So, there is a door and by this door it goes into the kernel mode of execution. And I can have some checks introduced at this door so, that will stop me from getting this unrestricted access with the kernel mode. So, if everything is fine the process is allowed to enter into the kernel mode and after going to the kernel mode it does this restricted set of operations.

Like when it is when the program is allowed to go to the kernel mode so, it is allowed to access this printer and then it will do the print operation here and as soon as the operation is done it will be taken back and it will go to the user mode of execution. So, that way the programs they when they start they start in the user mode of execution as and when it needs some resources. So, it will go to the kernel mode of execution and once that resource utilization has been done the program or the process returns back to the user mode of execution. So, that is how the operating systems are generally designed so, whatever critical resources that you have in the system the OS designer so they will put it into the kernel mode of access.

So, until and unless a program is not in the kernel mode it cannot access those resources so, that is how this protection is done. So, this mode bit provided by the hardware it provides ability to distinguish when the system is running in user code or kernel code. Some instructions designated as privileged instructions are executable only in the kernel mode if you look into any processor instruction set. So, there will be a privileged set of instructions so, they are executable only in the kernel mode. System called by a user asking the ways to perform some functions changes from user mode to kernel mode.

So, as I say that whenever a system access is required so, the program makes something called a system call and when the system call is made the user the program is taken to the kernel mode of operation. After system call ends so, it will go the program will go back to the user mode of operation. So, this way we have got two modes operation; user mode and kernel mode, kernel mode uses privileged instructions provided by the architecture designer and that helps in protecting the data structures and resources that we have in the operating system, that helps in keeping the integrity of the operating system as well as saving the system resource access.