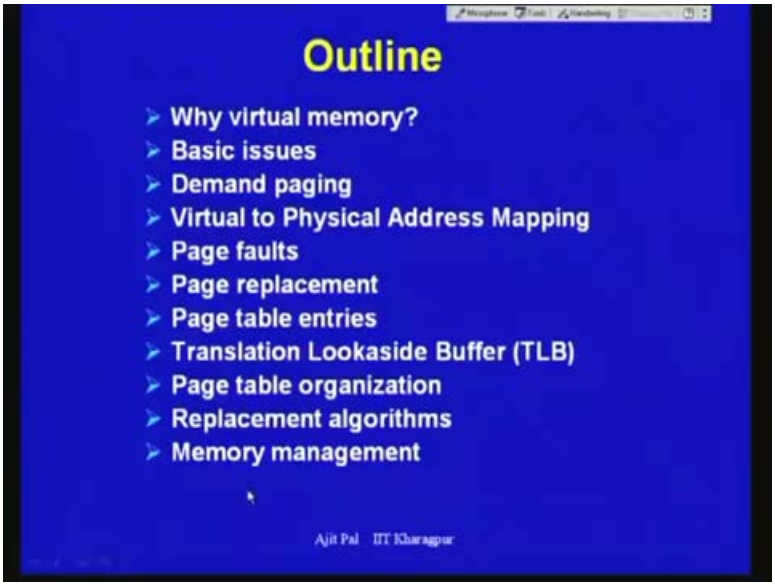
High Performance Computer Architecture
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 29
Virtual Memory

Hello and welcome to today's lecture on Virtual Memory, so we have discussed how the performance of memory system can be improved by organizing memory in a hieratical manner, and your focused our attention on cache memory, and also we have focus our attention on main memory. So, we have discuss the hierarchy between cache memory and main memory, now we shall go for another level of hierarchy that is between main memory and the disk memory.

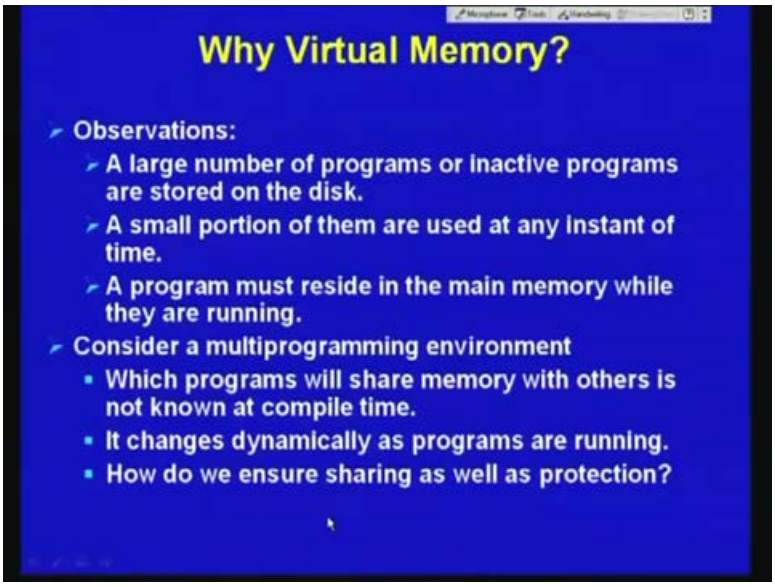
(Refer Slide Time: 01:35)



So, this is the outline of the lecture, first I shall discuss about why do you really require virtual memory, then the important issues related to virtual memory we shall discuss. And as we shall see it will involve a particular approach known as demand paging, and we shall discuss about virtual to physical address mapping, it will require some address translation mapping of virtual address psychical address. Then they will be page faults, page replacement and page table entries, possibly I shall able to cover all this topics in today's lecture. The remaining topics translation look a side buffer, page table

organization, replacement algorithm, memory management these topics I shall cover in the next lecture.

(Refer Slide Time: 02:30)



So, let us first try to address the question why virtual memory, we know that our observation is that a large numbers of programs or inactive programs are stored on the disk. That means, disk can be consider as kind of repository where you store all for different types of programs, but some of them very few of them are active at a particular instant of time that is our observation. And program must reside in the main memory while they are running, this is another important restriction that is that whenever you are trying to executive a program, it must be main memory resident.

That means, we have large programs large number of programs store in the hard disk, very few of them are active which needs to be present in a main memory, this is our first observation. Second is a let us consider the multi programming scenario which is very common in present day a environment; that means, most of the computers are running in this multiprogramming mode. And whenever we have we are running in multiprogramming mode, then we shall will be having large number of programs by different users.

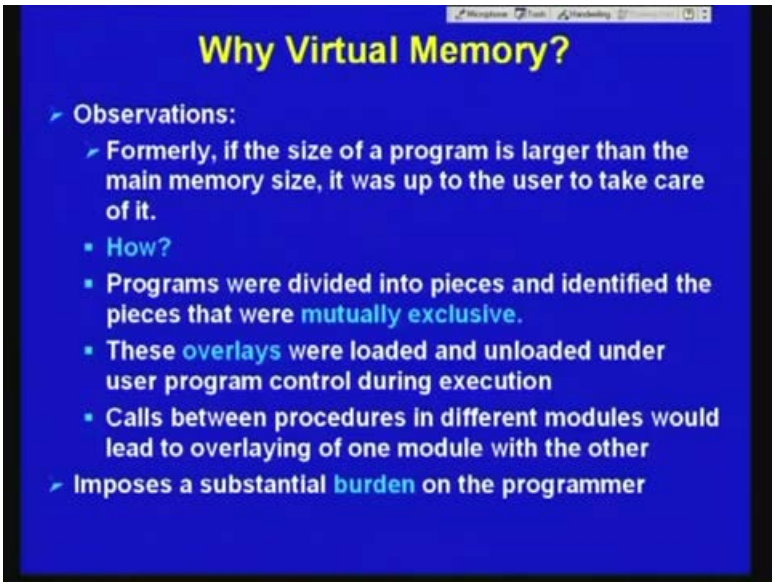
Now, the question is which programs will share memory with other is not known at compile time. So, at compile time whenever we are compiling a program particular user we do not know, which part of the memory will be shared, and how it will be shared with

other programs. That means, that the it is not known time only it is known at wrong time at compile time this is not know because it changes dynamically as programs are running.

So, programs are running I mean there will be context switching a program will be taken out of I mean that processer will not execute, will you suspended then again it will be a program will execution will resume. Because, we are using in a time sharing I mean time division multiplexes manner, so question is how do we ensure sharing as well as protection.

So, this are the two conflicting things, we want to protect one is this program from another research program, at the same time we also want that there will be kind of sharing if necessary if required. So, these two conflicting things are to be satisfied with the help of this virtual memory.

(Refer Slide Time: 05:43)

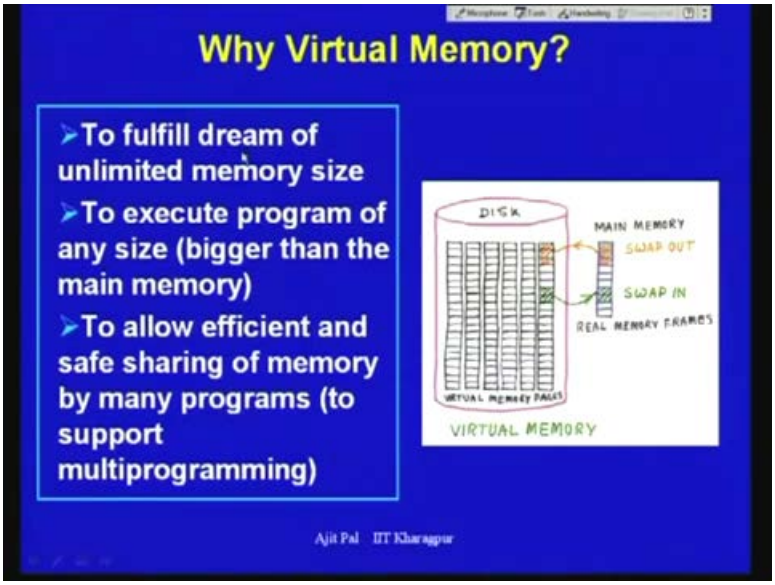


Second observation is formally the size of the programs is larger than the main memory it was up to the user to take care of it. So, if you go back to the history of the development of computers, you will find that earlier it was the responsibility of the program are user to load the program and memory and you to fit it in the main memory. So, in such a case there is a restriction; that means, if the size of the program is larger than the main memory, than what will happen cannot be executed.

So, what was done the programs were divided into pieces and identified the pieces that was mutually exclusive. That means, the programs which were mutual exclusive I mean the programs which are they only some of the programs will be loaded in the main memory not all. So, this overlays while loaded and unloaded under user program control during execution; that means, it was the user was responsible to swap in and swap out of their programs from the hard disk to the main memory, as when required a during the execution.

So, this calls then particularly when calls between procedure and different modules should be lead to I mean whenever you have got a calls between procedure in different module. Actually those programs that which you divide into mutual exclusive manner those whose are called modules, so if these calls between procedure in different module that would lead to overlaying of one module with another. And obviously, this imposes a substantial burden on the programmer, so this was a scenario in 80's and you can imagine what was burden on the programmer fortunately, now the burden mean overcome.

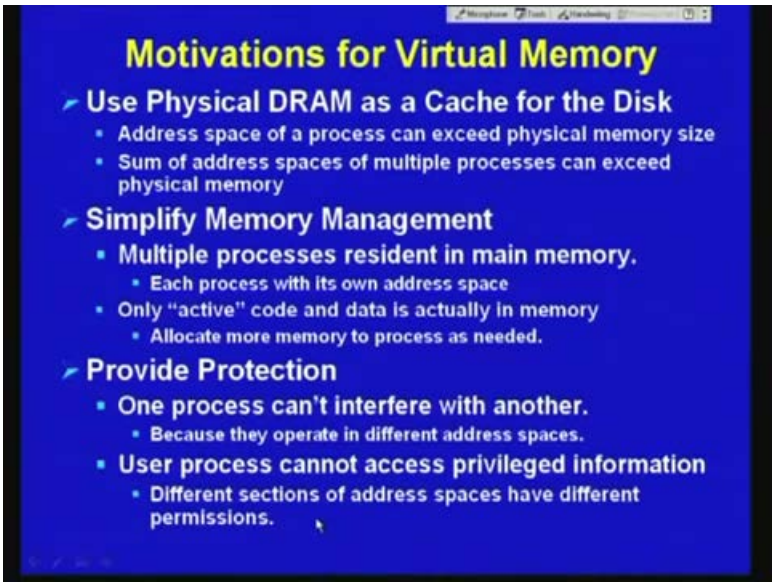
(Refer Slide Time: 08:01)



So and third virtual memory as has been now available in all computer systems, and it satisfied two important requirements; number one is to fulfill the dream of unlimited memory size. That means, to the programmer or user it gives a elusion as it the size of the main memory that it has got or size of the memory that it has got is unlimited, in other words there is no restriction on the size of the program.

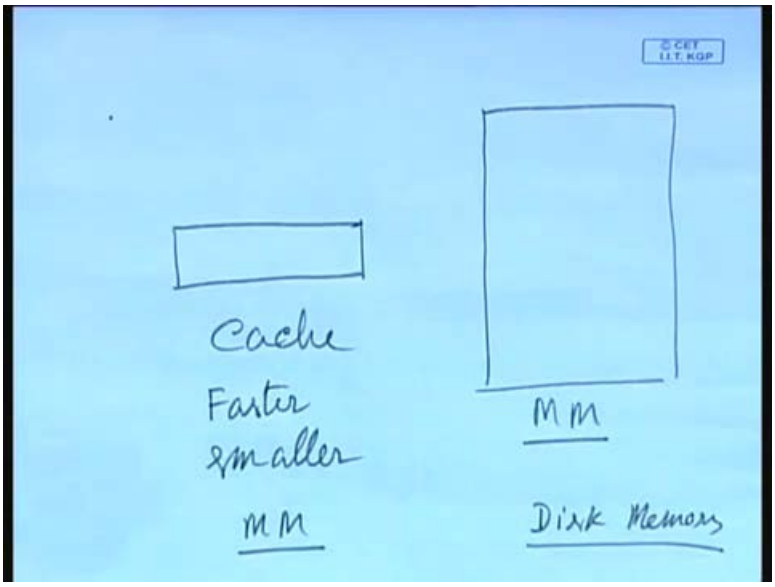
So, it allows you to execute programs of any sizes bigger than the main memory, and to allow efficient to safe sharing memory by many programs. Essentially to support multi programming this is essential, and this can be achieved as I told by swapping in and swapping out a programs. But, that should be transparent to the user, so user will not do it the virtual memory system will do it automatically.

(Refer Slide Time: 09:14)



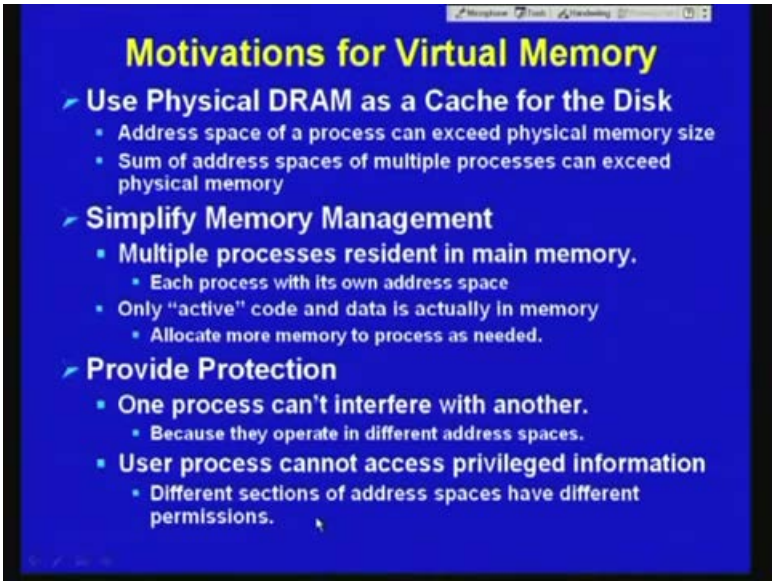
So, we can say that I mean how really the virtual memories implemented, first thing is does is it uses physical DRAM as cache of the disk cache memory of disk.

(Refer Slide Time: 09:36)



We have seen that the cache memory and main memory hierarchy you have got main memory and a part of that was stored in the cache memory, and which is; obviously, faster and smaller then the main memory. So, that is how the cache memory, main memory hierarchy goes implemented, now you are a going for another level where in this place of main memory you shall be having disk memory. And main memory will be consider as the cache of the disk memory, so that is the basic idea.

(Refer Slide Time: 10:23)



And address space of process can exceed physical memory size so; that means, that restriction that particular program need not be bigger than the main memory that restriction is no longer arises. And some of address spaces of multiple processors can exceed physical memory, so we are taken care of two things, number one each users program size can be bigger than the main memory size.

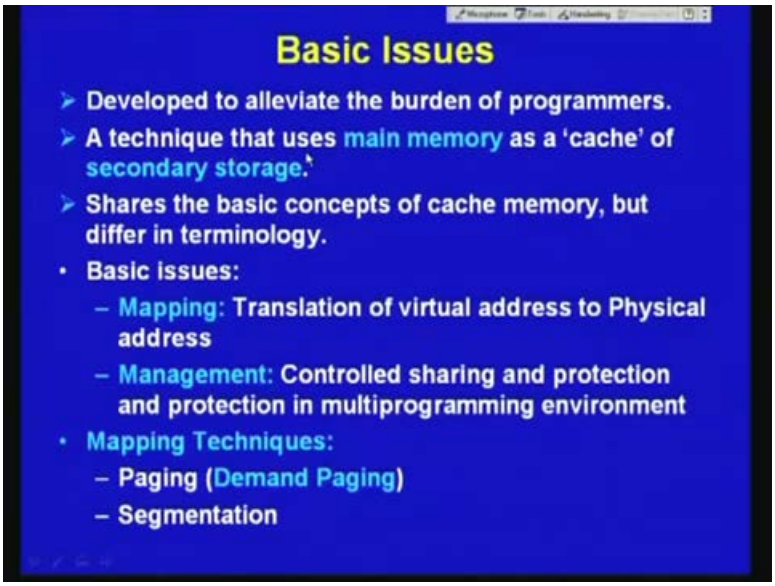
Moreover the some totals of multiple users programs size will definitely will be bigger than the size of the main memory. And it will simplify memory management, multiple processes resident in the main memory; that means, simultaneously multiple process will be kept in the main memory. And each process with it is own address space; that means, each of this process coming from different user will have it is own address space, and that we shall call virtual address space as we shall see.

And only active code and data is actually in memory; that means, we are not loading all parts of a different users program in a main memory, only active code and data are

actually in the memory, as we do in case of cache memory. So, allocate more memory to process as needed; that means, since it is dynamic as we need more and more main memory. Whenever we are executing program that can be dynamically increased, more memory can be assigned.

And it will also allow you provide you protection one process cannot interfere with the other; that means, it will be through the virtual memory user will not do directly. Because, they operate into different address spaces, and user process cannot access privileged information, so this is another important aspect, so different sections of address spaces have different permission you will see that later on we shall discuss about it and permissions for different parts can be different like read only access only like that.

(Refer Slide Time: 12:59)

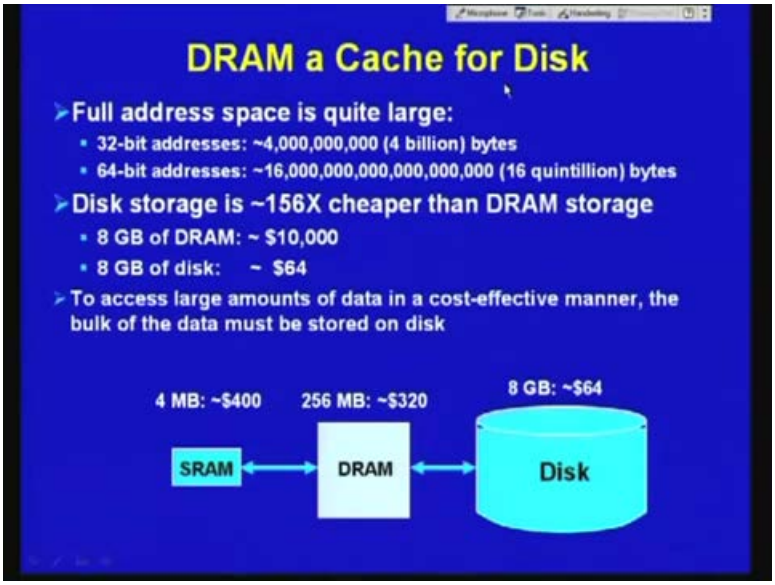


So, we cannot say that this virtual memory was developed the alleviate the burden of programmers, and a technique that uses main memory as a cache of secondary storage. And share the and it will indeed shares the basic concepts of cache memory, but using different terminology because of historical reasons, the different terminology were used in the context of virtual memory, and those are different from the terminology that are used in a cache memory we shall see.

However it will involve to important steps or issues mapping, translation of virtual address to physical address, and management control sharing and protection and protection in multiple programming and environment. So, these are the two basic issues

to be del to it, and as we shall see mapping techniques can be broadly divided into two types paging, in fact special type of paging known as demand paging and segmentation, which we shall discuss in my next lecture.

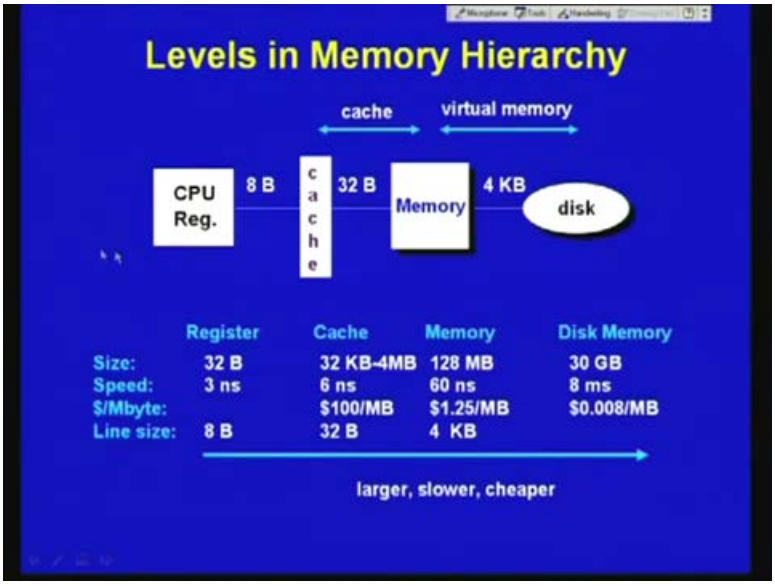
(Refer Slide Time: 14:12)



Now, let us have a fill about the differences sizes as we use the main memory as cache of disk. So, full address space is quite large if we use 32 bit address, then we have got a 5 4 billion bytes; that means, 2 to the power 32, 4 billion bytes are to be stored, and whenever we go for 64 bit address as it is true for 64 bit processers, then the size of the full address is quite large 16 quintal bytes. So, you can imagine the size of the address I mean with the help of 32 or 62 bit address that can be generated, and another observation, which I mentioned earlier that the disk storage is much cheaper than the dynamic RAM storage. So, an particularly 150 times cheaper then dynamic RAM for example, 8 GB of dynamic RAM will cost may cost 10,000 dollar, and 8 GB hard disc may cost only 64 dollar. So, these are the typical prizes with times this are this prices are going down; however, that ratio remaining more or less same.

So, two access large amounts of data in a cost effective manner, the bulk of data most be store in memory. So, this gives a motivation that we can have very large disk, but relatively must smaller dynamic RAM, and most of a our data and program should decide in the disk because we can store in the disk in a much more cost effective manner.

(Refer Slide Time: 16:09)



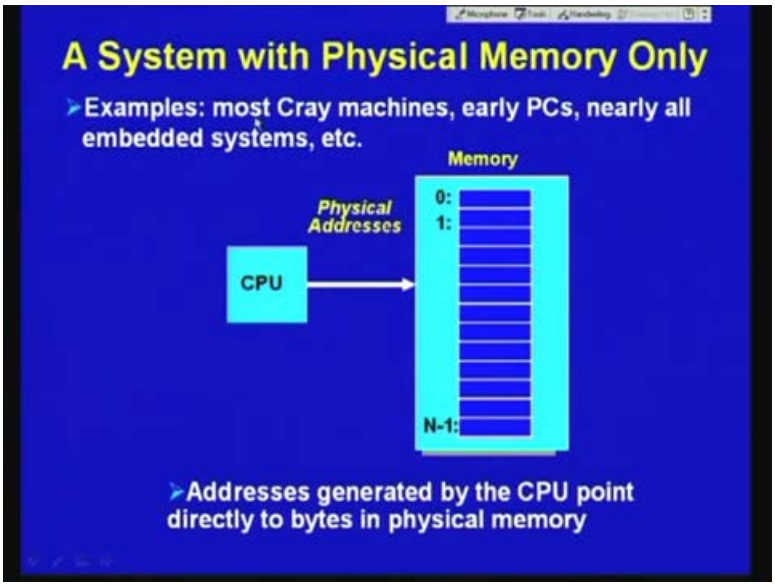
So, this are let two levels on memory hierarchy, one is between cache and main memory, another is between main memory and hard disk. And as you can see the two levels of hierarchy is deducted here, you have got CPU which primarily access from cache memory. And the transfer that can take place is maybe 8 byte transfer between CPU and cache. And on the other hand between cache and maim memory as you have seen a multiple words a get transferred, so maybe 32 4 watts, so 32 bytes are getting transferred.

On the other hand, if you look at the hierarchy that is we are discussing today virtual memory. The transfer will be 4 kilo byte; that means, later on we shall see we shall be using at a term call page, a page will be transfer not a block; that means, page is consider to a block in the virtual memory hierarchy. And this are the typical values this size of register as we know 32 byte it can be I mean sizes of register, than 32 bit it has to be bit, then cache memory size is 32 kilo bit to 4 mega bit when memory can be 128 mega bit this can be 32 GIGA bit.

So, speed varies 32 3 nanosecond whenever the processer is a accessing from register, whenever it is accessing from cache memory it will take 1 r 2 cycles 6 nanosecond from main memory 60 nanosecond. But, from this it takes the norm mask time 8 nanosecond, so it is very large, so you may say that the miss penalty for this virtual memory is two large. And cost as I already told is becoming smaller and smaller, as we go from cache memory to disk. And line size is also changing as we can see 8 byte, 32 byte and 4 kilo

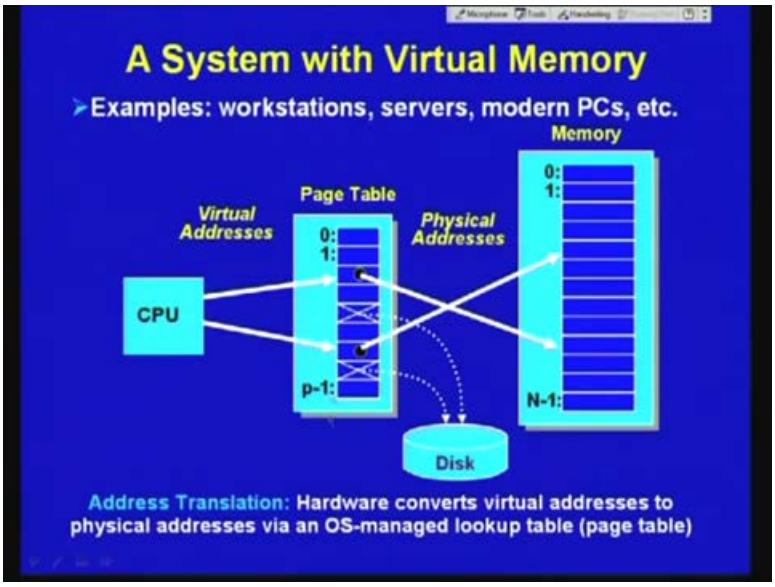
byte; that means, block size and page size. So, as we are moving from one hierarchy to another hierarchy going for larger, slower and cheaper memory.

(Refer Slide Time: 18:40)



In the earlier when virtual memory was not used, was not prevalent in those days the process the CPU was directly generating the physical address. So, example is most of the Cray machines, early PC's, and nearly all embedded system gadget, so virtual memory concept where not present in early systems. So, addresses is generated by the CPU point directly to the 2 bytes input physical memory.

(Refer Slide Time: 19:14)

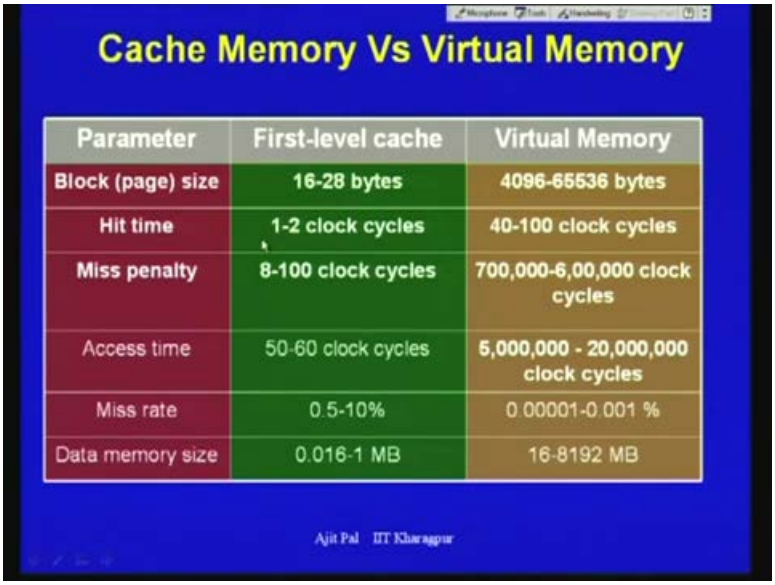


However, in the present the system as we move to virtual memory system, and nowadays all the workstation, server, moderns PC's, modern PC's all use virtual memory system. Where you have to go through and intermediate steps that intermediate steps is known as address translation, so CPU will generate virtual address and before that address is presented corresponding to the main memory, we have to go through transitions process that is known as address translation.

And that hardware come wards virtual address to physical address, via on operating system manage look up table, and that table is known as page table. So, a page table is shown which really maps virtual address to physical address, and that physical address will be presented to the main memory of course, they are can have another hierarchy as you have seen cache memory. But, I mean we can have, but I mean we shell refer it to main memory only.

And the address in the page table not only it will refer to main memory, it may refer to the disk whenever it has not been transferred to the main memory. So, in the page table there is province for keeping information about the main memory address, as well as the disk address where it is kept in the disk.

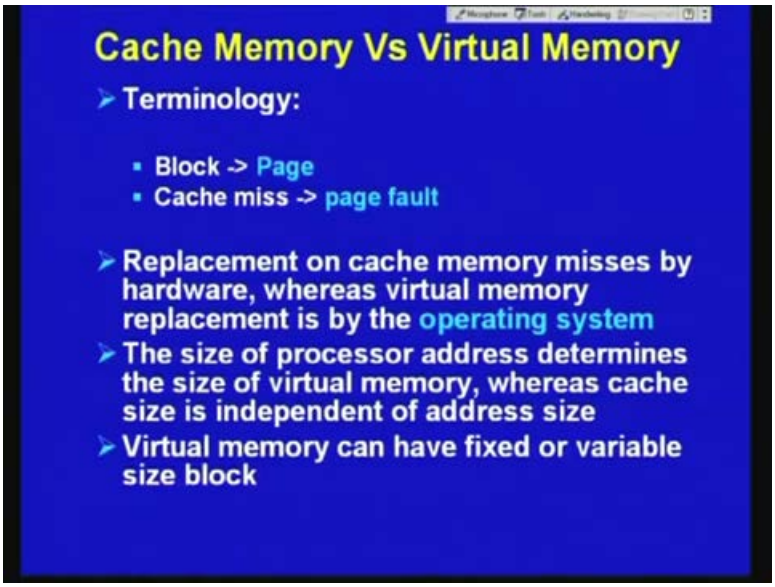
(Refer Slide Time: 20:51)



And this are those typical comparison parameters I have already mention block size, first level cache 16 to 28 byte, virtual memories it will large 4 k to 64 k hit time as I said 1 to 2 cycles, and virtual memory may take for 40 to 100 clock cycles. And miss penalty as a

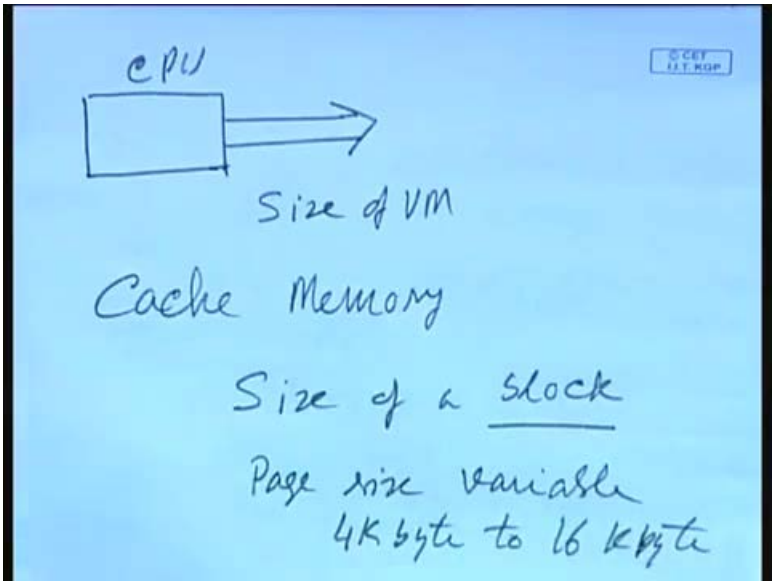
you can see is very large in a virtual memory 100, 1000 times larger than the cache memory. And access time is also quite large; however, only good thing is that miss rate for virtual memory correspondingly much smaller, and data memory size as I have already told in pieces in case of virtual memory this are for the purpose of comparison.

(Refer Slide Time: 21:45)



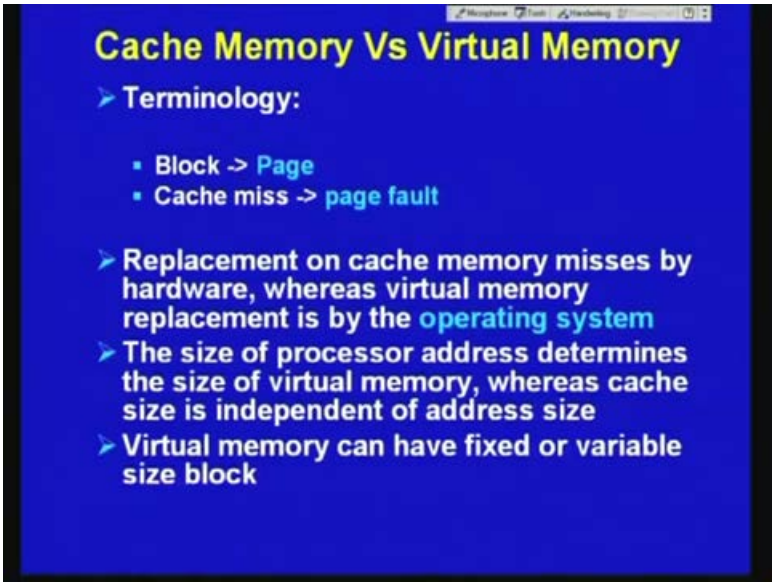
And as I mention there will be difference in terminology, we shall using page of block, and we shall refer to a cache miss by page fault in the context of virtual memory. And other difference are placement on cache memory misses by hardware that is done in case of a cache memory as you have seen. Whereas, in cache of virtual memory it will handle by software operating system, so the size of processer addressed determines the size of virtual memory, where as cache size independent of address size.

(Refer Slide Time: 22:28)



So, we have seen that you know that CPU will be generating address and that address size will determine the size of the virtual memory. Because, that address will be I mean you cannot really change the size because CPU will keep on generating the address and bigger then that you cannot have virtual memory.

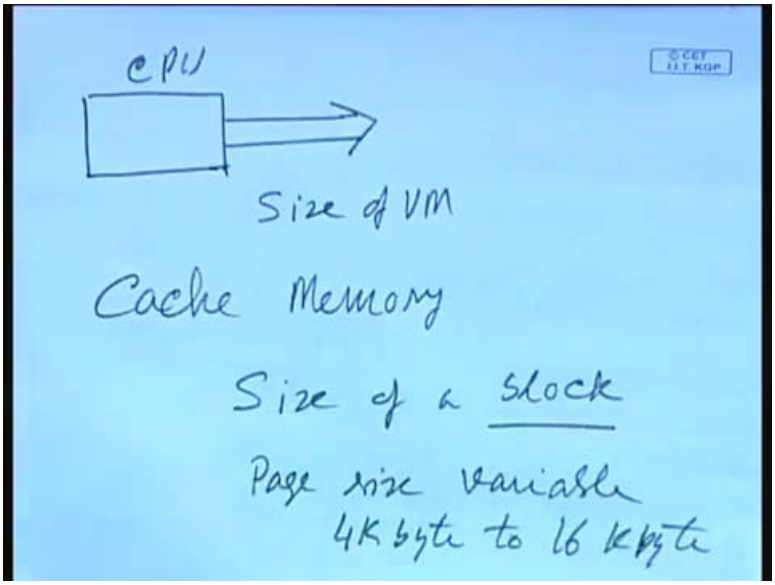
(Refer Slide Time: 22:59)



But, you know in case of cache memory we have seen, it is independent of the address size. Because, the address size in case of a physical memory can be smaller can be different of course, in the earlier systems there was no distinction, the virtual that this

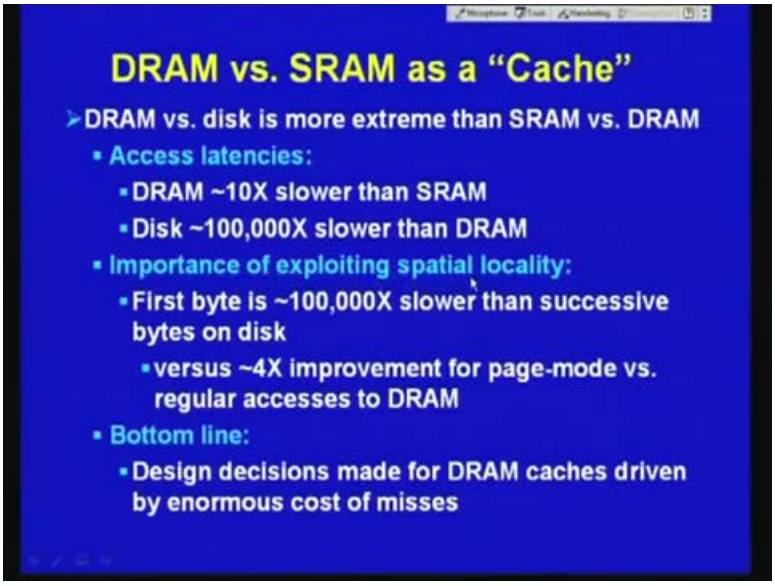
that address that was generated by the processer was directly apply to the main memory. But, in the present the context it will be different, so virtual memory can have fixed or variable size blocks that we shall see that page size can be user defined can be variable.

(Refer Slide Time: 22:28)



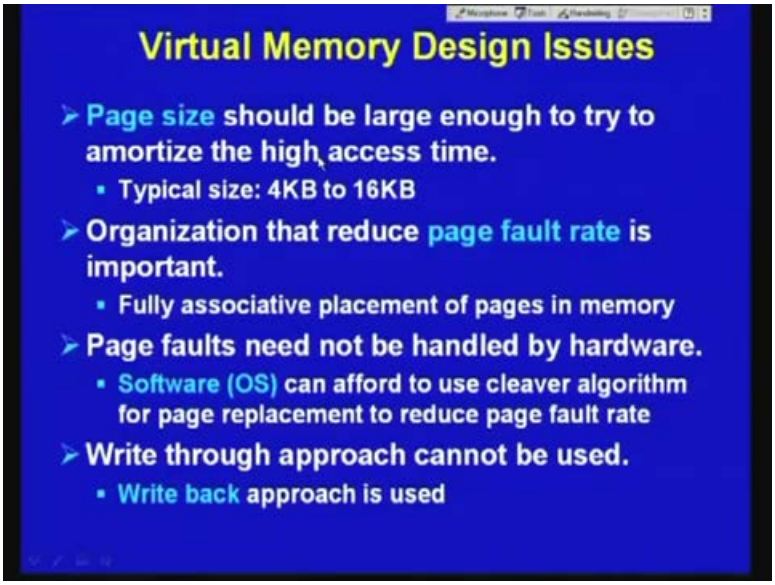
We have seen in case of cache memory the size of a block whose decided by hardware; that means, hardware is responsible for the deciding the size of a block and accordingly the cache memory is implemented. But, in case of virtual memory that size of a block that this is your page can be variable, user or program can changes and it can vary from say 4 kilo byte to may be 16 kilo byte. And nowadays people are still using bigger a page sizes, it can be 32 kilo byte or even 64 kilobyte.

(Refer Slide Time: 24:36)



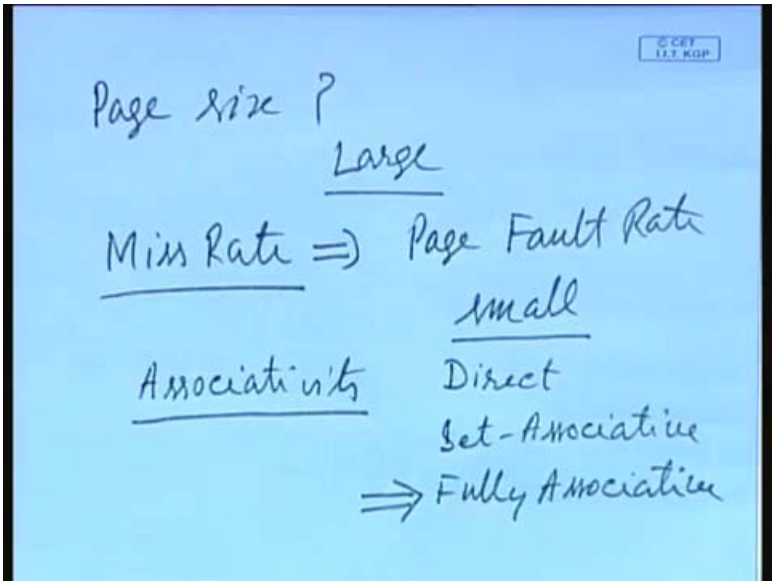
So, I have already discus all this issues.

(Refer Slide Time: 24:42)



Now, let us consider what are the important designing issues based on what you have discuss, so far. Number one is page size, what should be the page, the page size should be larger enough to try to amortize the high access time.

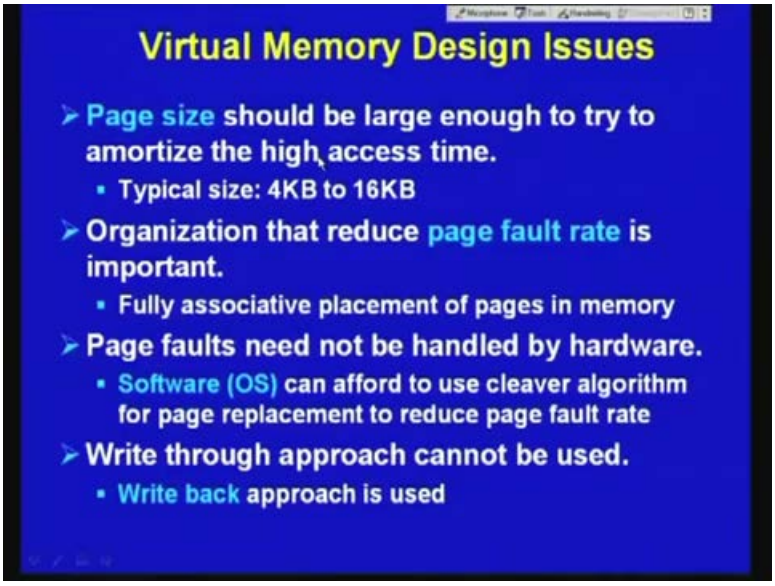
(Refer Slide Time: 25:05)



We have sensitizes in the hard disk that size of a page can be variable, but what should be the size. It should be large, why it should be large, the reason for that is it should amortize the size of the high access time, and as I mention this size can be vary from 4 kilo byte to 16 kilo byte. And organization that reduces page fault rate is important and; that means, the miss rate which we know we call at page fall rate that should be small, and you have to design such a way that page fall rate is reduced. How it can be reduce it, will be depended on the design of the virtual memory.

And one important parameter as we know we have seen that associatively is a important parameter that can effect the miss rate. And as we know there are three possibilities, direct mapping or it can be set associative mapping, and fully associative mapping, in the context of virtual memory as we shall see we shall always use fully associative mapping. So, that the miss rate is reduced; that means, fully associative placement in pages of the memory. What does it really mean; that means, that the page that you transfer from the hard disk to the main memory can be placed anywhere. So, that is your fully associative placement in pages.

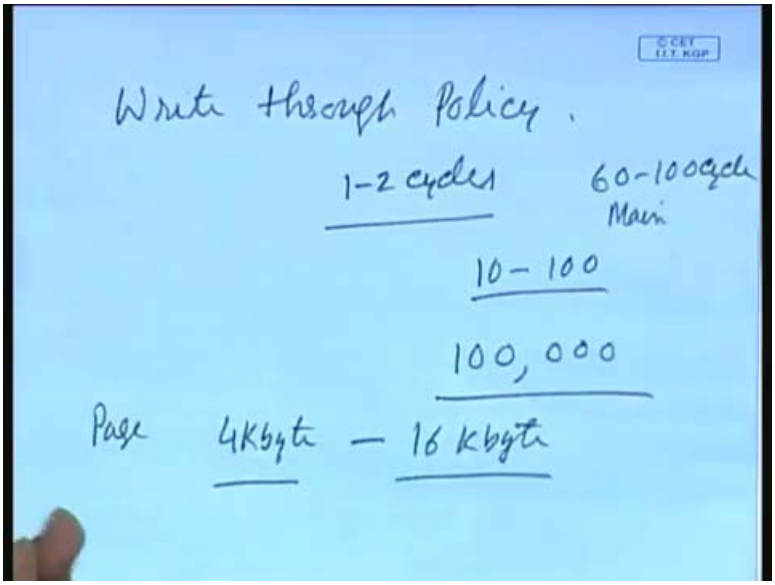
(Refer Slide Time: 27:18)



So, that saw the virtual memory will be organized, then page faults need not be handling in hardware. We have seen in case in context of cache memory, the whenever there is a cache miss it is handled by hardware, but it cannot be handle by hardware, the reason for that is if we use software, we can use cleaver algorithm for page replacement to reduce the page fault rate.

That means, whenever we are using hardware it has to be simple, we cannot use complicated algorithm say sophisticated algorithm for page replacement. So, page replacement has to be done in such a way because we are using fully associative mapping that it will involve replacement of pages and page replacement has be done. So, that a particular page which is replaced is not required in near feature that can be achieved by with the help of sophisticated algorithm. And that will help to reduce page fault rate, another important design issue is write through approach cannot be used.

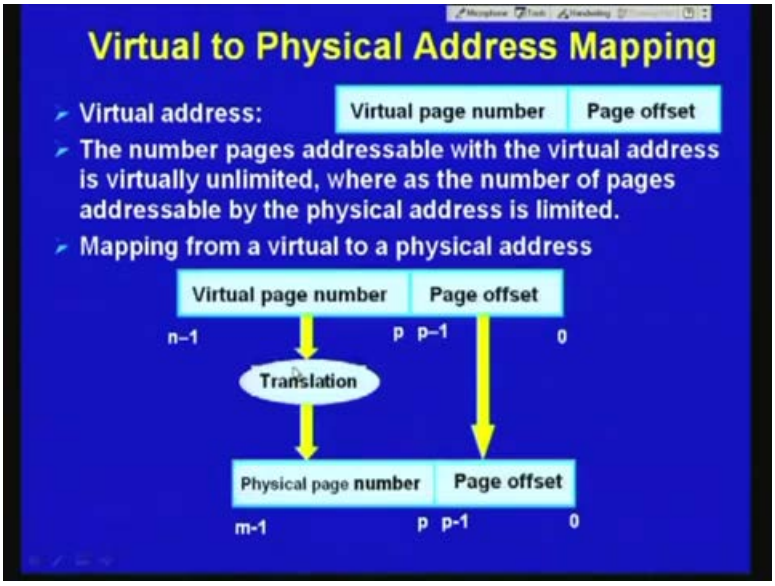
(Refer Slide Time: 28:52)



You have seen in the context of your cache memory we normally use write through policy, why write through policy can be use in the context of cache memory, the reason for that is the difference in speed is very small. So, instead of 1 or 2 cycles for reading from cache memory it may take maybe say 60 to 100 cycles, whenever it is rate from main memory. So, the difference in speed is not very large may be 10 to 100 times, but in case of you are a virtual memory we have seen it is 100,000 times slower.

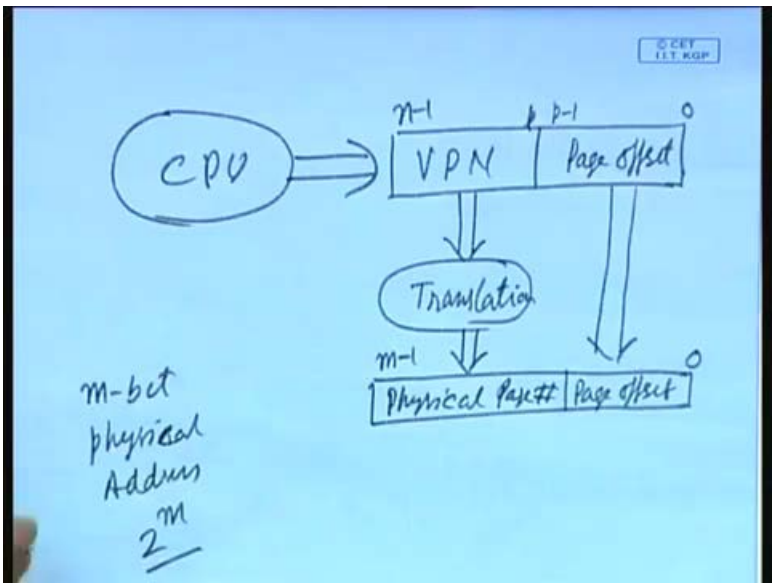
So, that requires that you have to use write back approach, moreover particular page is quite big it can be 4 kilobyte to 16 kilobyte let us assume. So, there is a possibility that you will be performing many writes in a single page before it is replaced. So, write back policy will allow you that; that means, you being in a page keep on writing into it many times, there is no need to write in a hard disk, if you keep on writing in main memory as long as it is present it is not replaced. So, that is why write back policy is the suitable approach for virtual memory. So, these are the basic design issues, and we have got answer an what type of things to be done in virtual memory that you have discussed.

(Refer Slide Time: 30:43)



Now, let us focus on the mapping aspect, we shall be we have to do virtual address mapping to physical address how exactly it is been done. So, a processor generates virtual addressed it has got two components virtual page number and page offset.

(Refer Slide Time: 31:07)



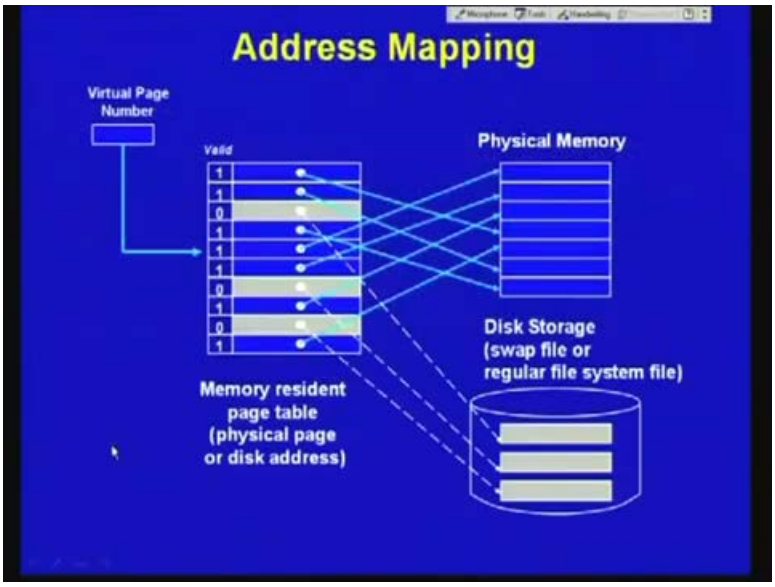
So, this is what is generate by the CPU, CPU is generate that this is the CPU, CPU is generate virtual page number VPN plus page offset. ((Refer Time: 31:31)) And then the number pages addressable with the virtual addresses is virtual unlimited because that number of the present is quite large. Whereas, the number of pages addressable by the

physical address is limited because you have got a very limited number of search pages in the physical memory.

So, mapping from virtual to physical address is necessary and you will do the kind of translation. So, this virtual page number will go through a translation, suppose you are using n bit address out of each which p bit corresponds to page offset, and remain p 2 n minus 1 correspond to virtual page number. So, this is the virtual address that is generated by the CPU, and the page offset directly goes to the physical page number.

So, physical page number is divided again into two parts, one is your page offset that directly comes from that virtual pages number, and here we perform a translation. So, it need not be same as your n it can be smaller than that, so let us give an a number m minus 1. So, you have got m bit physical address I mean m bit physical address size and having 2 to the power m bits of main memory you can say. And this is your physical page number, so virtual page number is generated by the CPU there is a translation that translation generating the physical page number, and this is concatenated with the page offset to generate the physical address, so this is the physical address.

(Refer Slide Time: 33:43)

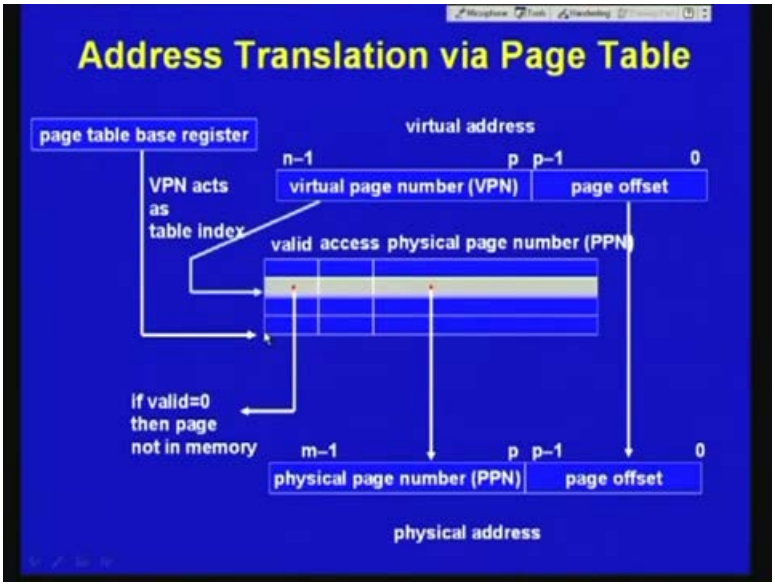


And this has to be done with the help of a table, memory resident page table, and as you can see virtual page number generated by the CPU, which is applied to a page table. That page table does the translation from the virtual page number to physical page number, and here as you can see we have added bit known at I mean valid bit, why this valid bit is

necessary the valid bit is necessary because only those pages which are required by a program will be transfer to the physical memory or main memory.

So, when the valid bit is one they will that that page table will point to the physical address. So, here you will get the physical address, on the other hand those addresses corresponding to I mean when the valid bit is 0, they will pointing to a particular location in the hard disk. So, you can see the main memory resident page table can have I mean it will either point to physical page or at the disk address, so it will have two different possibilities that is present in the page table.

(Refer Slide Time: 35:15)

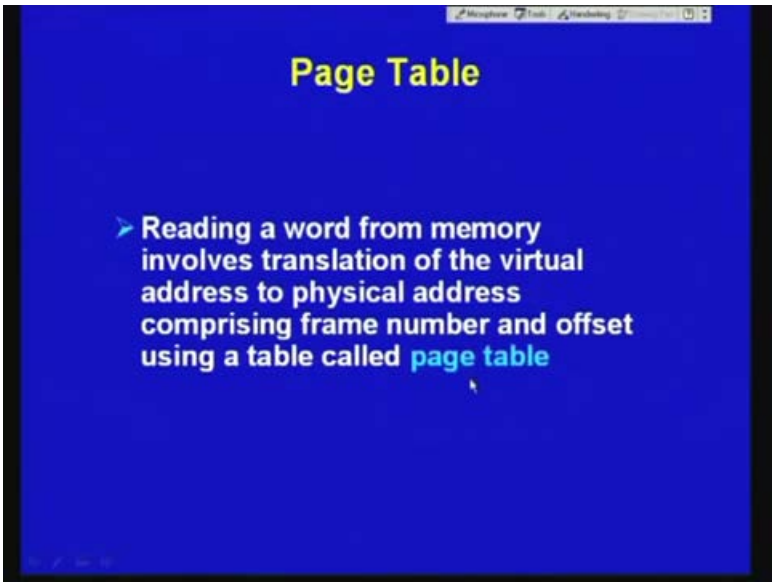


And this is how actually it is being done, you can see this is the page table resister, so as the you are storing the page table in the memory, there is the page table base register which points to the base address of the page table. So, with respect to that actual page number is a provides you a kind of index, so this virtual page number will act as an index to that a page table, and with the help of that you will get the physical page number.

And you can see in addition to the physical page number, it has got two more fields valid bit. So, if the valid bit is 0 then the page is not in the memory; that means, it have not been transfer to the main memory it is still present in the hard disk, you are assuming that the program is always present in the hard disk. And from this table we shall be getting the virtual that physical page number, and with that the page of offset will be

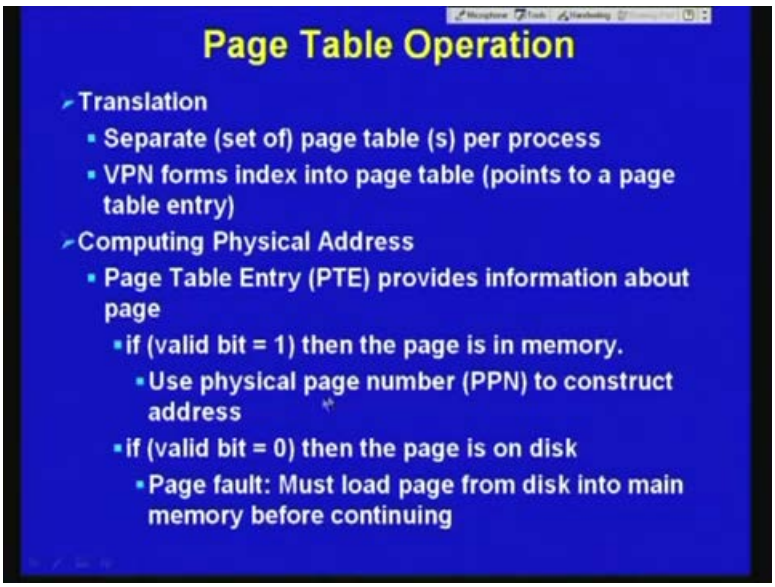
concatenated operated. And this physical address will be presented to the main memory and of course, cache memory will be there as a part of the main memory system.

(Refer Slide Time: 36:41)



So, reading a word from memory involves translation of the virtual address to physical address comprising frame number and offset using a table called page table. Actually this is called the page frame number this that physical page number is called the page frame number.

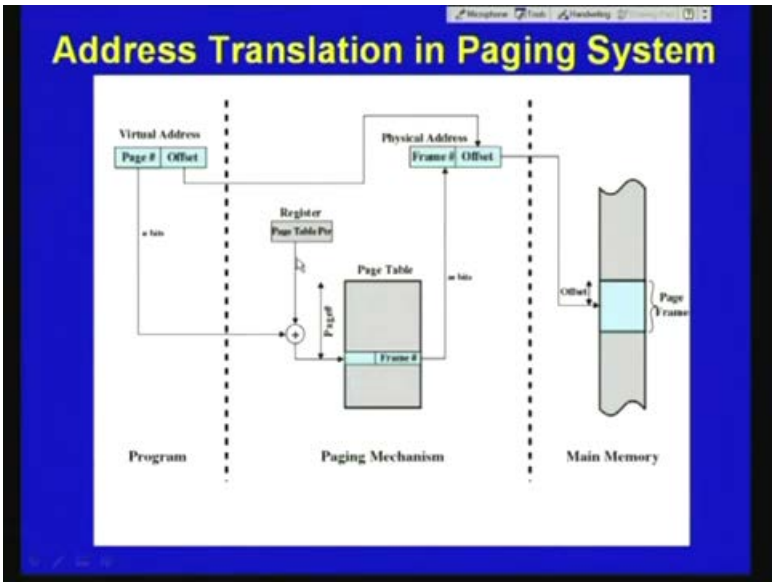
(Refer Slide Time: 37:07)



How the page table operation tacks place easily is illustrated here, first it goes it translation operation, you have got separate page table for process. As we know you can have different process that is present in I mean simultaneously getting executed by CPU of course, in a time division multiplex manner. So, for each process there will be separate page tables, and actual page number form index into the pages table points to a page table entry as you have seen.

Then it will to the compute physical address page table entry provide information about page and of course, it is valid bit is 1. Then the page in the memory and that will use physical page number to construct address, as you have seen with the help of the previous diagram, and the valid bit is 0 then the page is not on disk. So, page fault will occur and then you must load the page from the disk into main memory before continuing, so this is how the page table operation is will continue.

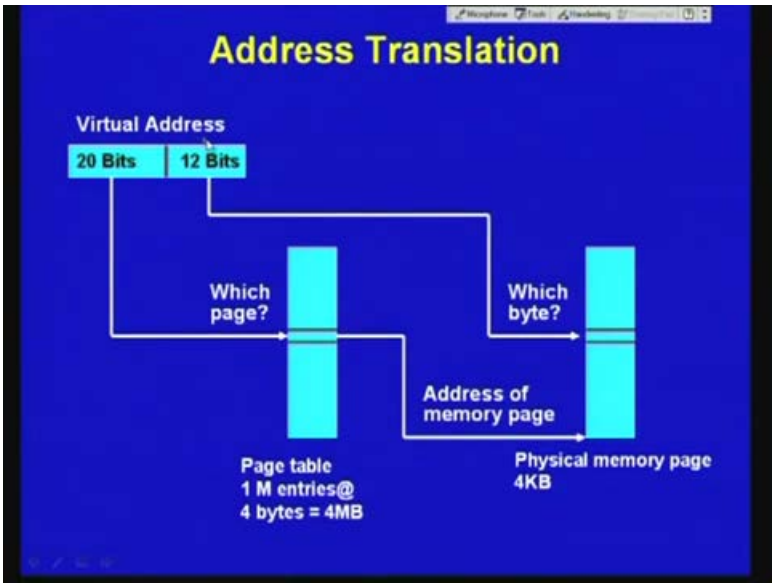
(Refer Slide Time: 38:23)



And this is illustrated with the help of diagram, so here you can see the virtual address designated by the processor comprising n bits, and this is that page table pointer that based addressed is providing. So, then this virtual page number with the help of that it is getting index, and we are getting entry in the page table where you have got a page frame number. And that page frame number is the physical address page number were the physical page number is stored, and that physical address, and that page off set is concatenated together and that is presented to the main memory.

So, this is how the address translation in the paging system works, as you can see it involves I mean program is generating, the virtual address then the paging mechanism we will generate the physical address, and that will ultimately go to the main memory this is how it will work.

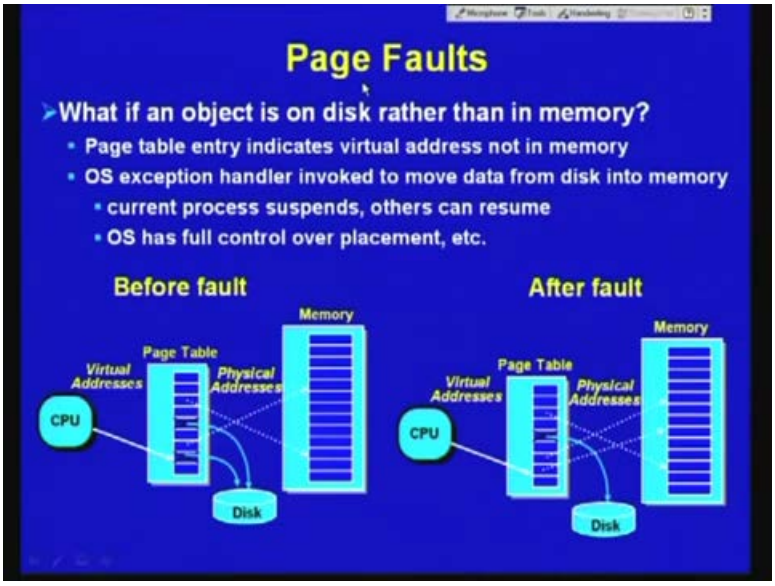
(Refer Slide Time: 39:33)



So, you can see here that virtual address is 20 bit let us assume and the offset is 12 bits, so this 20 bit virtual address is telling which page. And of course, that which page of this page table and you can have 1 M entry in the page table because you are using 20 bits. So, that page table is quite large as you can see, and that 4 byte which entry, so 4 mega byte is the size of the page table itself, so this tells you that whenever we are using virtual addressing that you have to store the page table of each user.

So, you can imagine that you will require 4 mega bit page table for each user, not for one process for all the processors that has been created will be having such a page table. And then you will having a physical memory page that will be in your main memory, and this page off set will give you which bit of this page. And of course, address of the memory page will be obtained from the page table that where the page frame number is stored that will give you the address of the memory page. And this essentially shows you a particular page and that particular page will be having the data that is requested for that is your which bit that is available in the main memory.

(Refer Slide Time: 41:27)

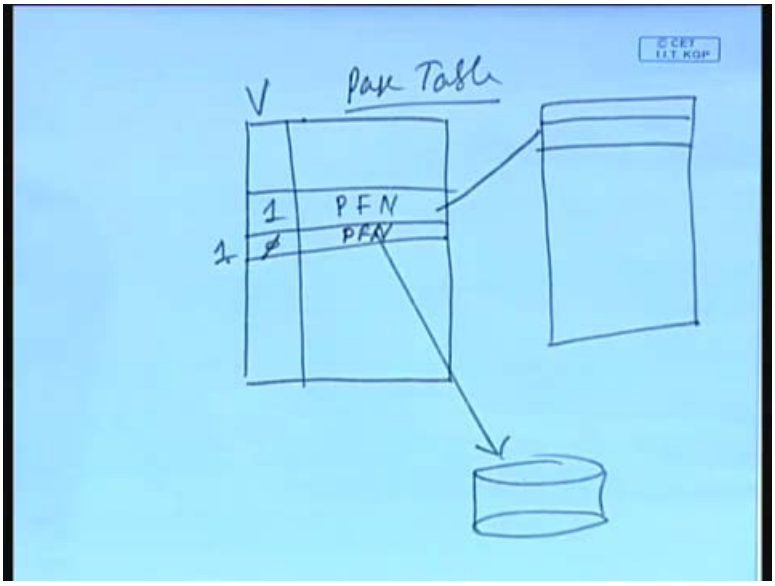


Now, we shall discuss about one important aspect that is your page fault, we have seen that a particular data may be present in the main memory or may not be present. Whenever it is not present in the main memory, this will lead to page fault; that means, as you have seen what is if an object is on disk rather than in memory, so if it is not in the main memory that will lead to page fault, and page table entry indicates virtual address not in memory.

We have already seen that in the page table we shall be having an entry valid bit that valid bit will tell that page table that particular page is not present in the main memory. So, operating system exception handler involved to move data from disk into memory, so here it will be taken care by the operating system, and current process is suspended, and others program other process can resume, and operating system has full control over placement. So, you can see we are using a combination of hardware and software, operating system and some software and hardware.

And this is the situation when before fault you can see these two are mapping to hard disk they address corresponds to hard disk. Now, this CPU was pointing to this particular address this was the virtual address and this was since it was not in the main memory; that means, it was 0 and virtual address I mean the disk was stored here. Now, after the control goes to the operating system, and it handles the page fault then after the fault these senior changes.

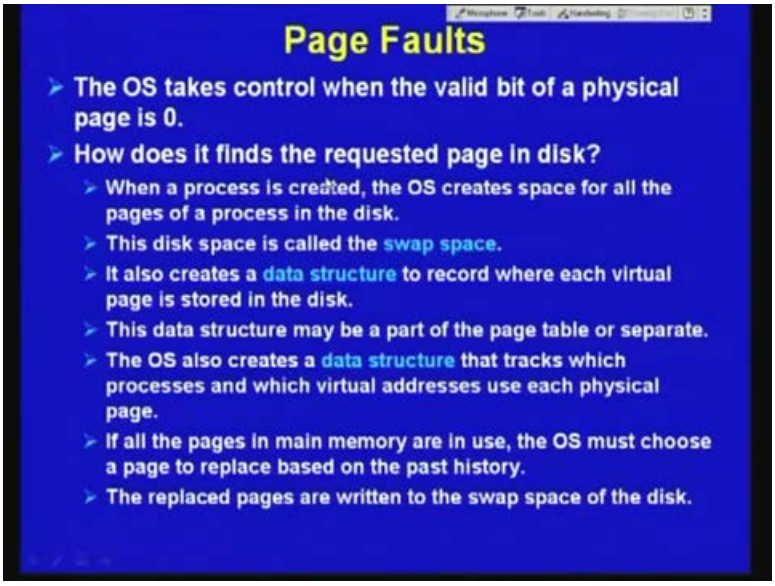
(Refer Slide Time: 43:45)



That means, you are in the page table you are having valid bit, and this is the page frame number this can point to main memory. And whenever it is 0 it will point to hard disk, and whenever it is serviced this will become 1 because if this these corresponding page has been transferred from the main memory to the hard disk to the main memory. And it will be converted to 1 and corresponding that page frame number corresponding to the physical memory will be present in the page table.

So, page table will be updated and this is what is being shown here; that means, that page table has been updated and 0 has been converted to 1, and that particular address is no longer pointing to hard disk, but it is pointing to the main memory.

(Refer Slide Time: 45:03)

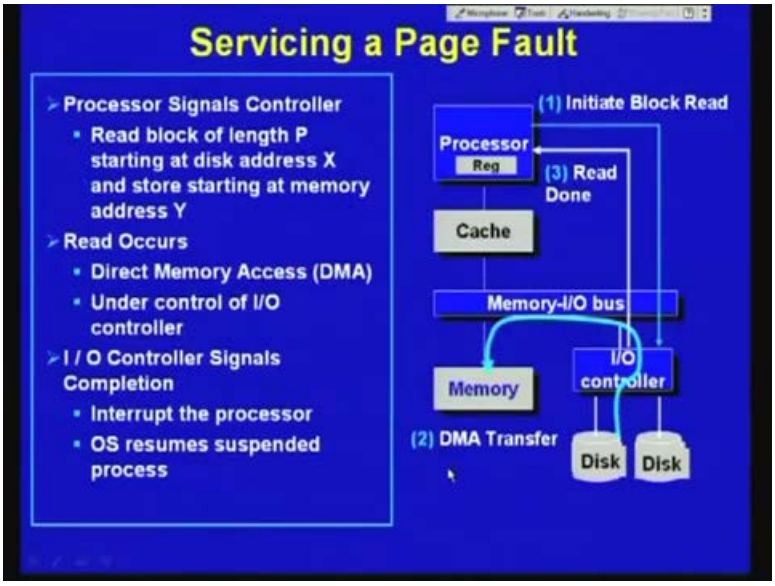


So, this is the detail how exactly it happens the operating system takes control when the valid bit of the physical page this is 0. And how does it finds the requested page on the disk that is another question, when a process is created, the operating system creates space for all the pages of a processing in the disk. So, in the disk all are present this disk space I called swap space, and it also creates a data structure to record where each virtual page is stored in the disk.

The data structure that you have referring to is essentially the page table, and this data structure may be a part of the page table or it may be separate. But, whatever what we shown right now is essentially the page table, and operating system also creates a data structure that tracks which processes and which virtual addresses use physical page. So, as you have seen in this is your data structure that text care of it, and if all the pages in main memory are in use, then operating system must choose a page to replace the based on the past history.

So, this is here it will be necessary to change the; that means, if this particular page which is being used. This page has to replaced and new page has to be taken from the hard disk to the main memory, so the replace pages are written to the swap space of the disk. So, what we are doing we are transferring between main memory and cache memory, main memory and hard disk this is how the swapping is taking place.

(Refer Slide Time: 47:01)



So, this the particular slide elaborates how exactly text place, so number 1 step is initiate block read, processors signals controller read block of length P starting at disk address X and store starting at memory address Y. And read text place whenever there is a page fault, then you have to read it from the hard disk and as you know we normally use direct memory access DMA. So, DMA technique is used that DMA approach transfers from the hard disk to the main memory.

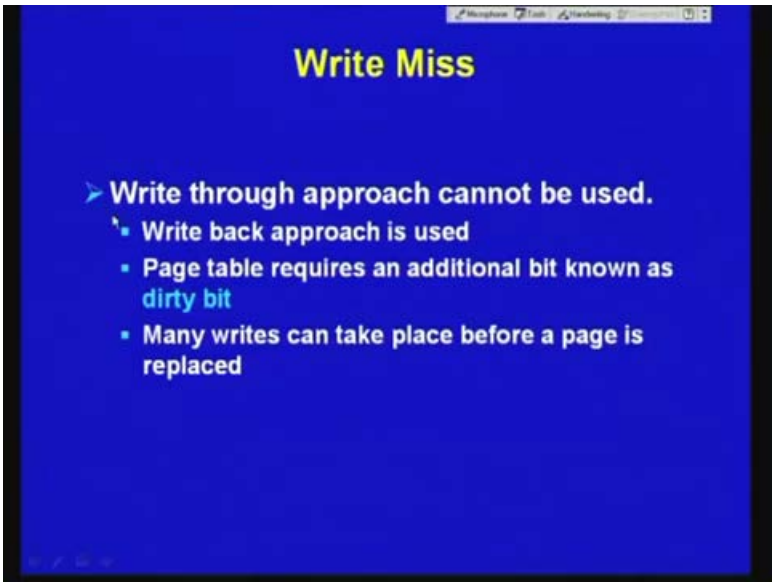
So, in step 2 the data to hard disk is getting transfer to the main memory with the help of the Input Output controller. So, the first step that was initiated, but since there was page fault we found that it is no longer present in the main memory, so from the hard disc it was read, and it was transfer to the hard disk. And then the third step is Input Output controller signals completion, that means after the DMA is over DMA Direct Memory Access is over.

Then the it interrupt the processor indicating that the Input Output controller interrupt the processer indicating that. So, here is a interrupt signal going to the processor indicating that the page fault has been serviced by the operating system, and then operating system resumes the suspended process. So, in the third step since the data is now available in the main memory, then it resumes it is operation, so this is how exactly the page fault servicing takes place with the help of the operating system.

(Refer Slide Time: 48:57)

Coming to the page replacement find a free frame to store a desired page, if there is a page frame use it. So; that means, it has to find out whether there is a page frame already available in the main memory or there is no page frame, so if is not their then use page replacement algorithm to select a victim frame. So, write the victim page to the secondary storage, and change the page and page table accordingly. So, you have to modify both the page on the page table entry, so read the desire page into the free frame, and change the page and page table, and you will restart the user process this is how the page replacement will takes place with the help of the operating system.

(Refer Slide Time: 49:54)

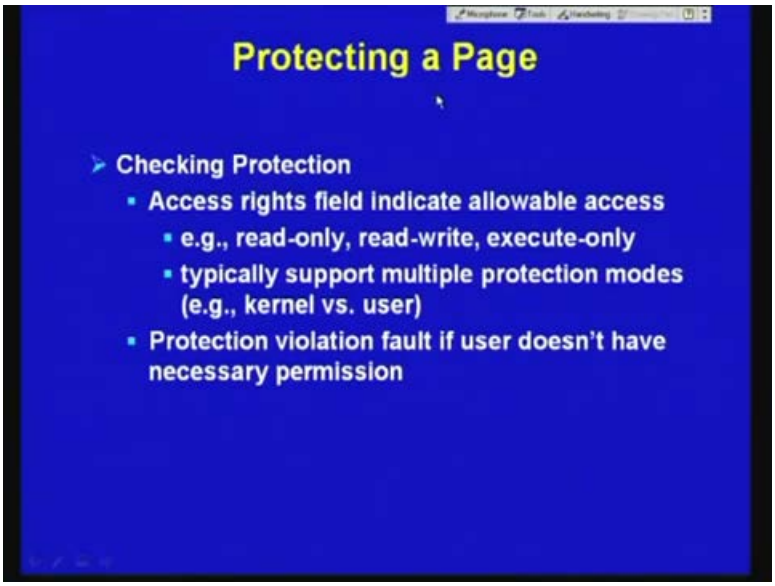


Now, the another important aspect is write miss, so as I already mentioned we cannot use the write through approach because size of the page quite is large. And because of the you know locality of references both that temporal and special because of temporal and special locality it is very likely that subsequent access by the processor are will take place from nearby memory locations. So, the page that has been transferred will be accessed may be in near future that is a reason why we use write back policy.

However, we will requires an additional bit known as dirty bit, so page table requires an additional bit known as dirty bit. ((Refer Time: 50:47)) So, in addition to the virtual bit that you already mentioned, you will requires a dirty bit in the page table, so dirty bit means whenever you have modify you will make it 1. That means, you have to written into it you will 1 if it is only read in then it will remains 0.

That means, initially whenever you transfer it then the dirty bit is 0, and whenever you perform writing into the main memory, then it is made 1 converted into 1. So, as I already mentioned many writes can be take place before a page is replaced, so once dirty bit is set you can perform many writes without modifying anything.

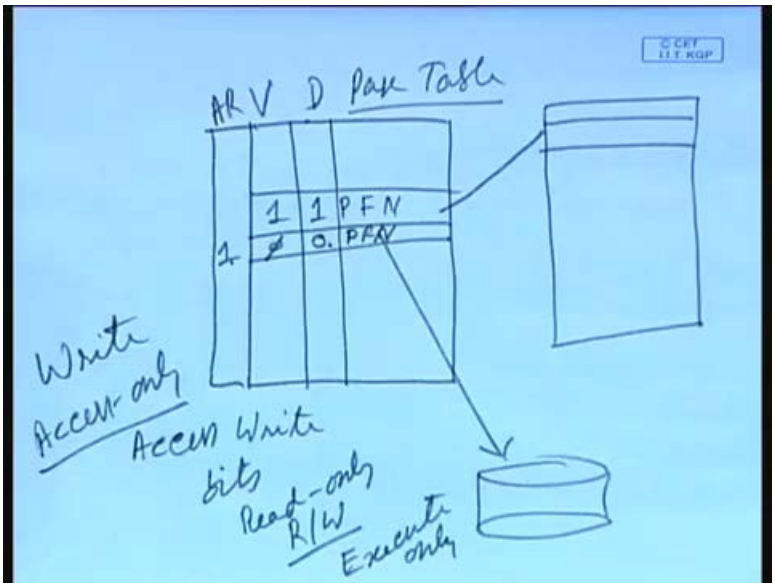
(Refer Slide Time: 51:35)



Coming to the protection as I mentioned, you have got many users, many process present I mean programs are many users are present or many processes are concurrently getting executed by the processor in a time division multiplex manner. So, many processes will be present in the main memory, how one is a program will protect from other user

program that is achieve with the help of protection bits. So, access right field indicates allowable access; that means, read only, read write execute only, so typically support multiple protection modes. That means, that kernels verses users we will see that this modifications can be only be done by the operating system, not done by the user.

(Refer Slide Time: 52:41)

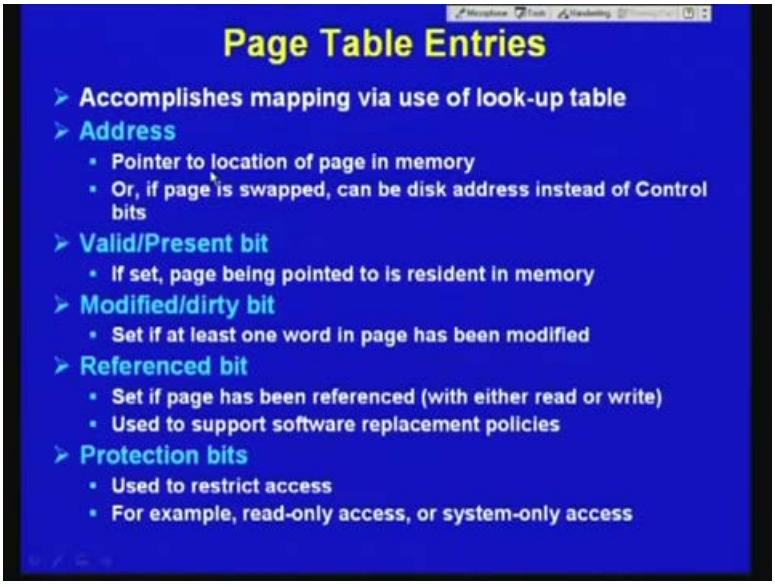


And that means here in addition to this virtual bits, you will be having those access write bits, which can be say read only or it can be read write if allow read write or it can be execute only this type of flag bits can be stored access write fields. So, you can several bits present there, so you can say this is your access right, so you can see the page table will have several other bits.

So, protection violation fault if users does not have necessary permission; that means, whenever you are trying to read a data from the main memory not only you have to check whether it is present in the main memory or not you have to also check whether the correspond that; that means, the corresponding permission is available.

Suppose your trying to write, but the access that is provided only access only; that means, you cannot perform write, so in fact that particular page is present in the main memory, you cannot perform write, because it is highlighting the access write, so it is reading to additional complication.

(Refer Slide Time: 54:19)



So, based on what we have discuss, so far we can see that, the page table will have the following entries number 1 is address, pointer to location of page in memory or if page in swapped can be disk address instead of control bits. Then valid or present bit if set page being pointed to is resident in memory, we have already mentioned about it, modified or dirty bit set if at least one word in page has been modified. So, in page have large number of bytes, but even if single byte has been modified, you have to modified then dirty bit will be set to 1.

Then reference bit set if page has been referenced with either read or write, so used to support software replacement. Later on we shall discuss I mean it is essentially implementing kind of LR use it recently used with help of these presently use a simplified version of LR being used implementation. Then you will be having protection bits used to restrict access, as I have already mentioned read access, system only access that type of production bits can be added in the production field.

So, with this we come to end of today's lecture, we have discussed about various aspects of virtual memory, particularly why virtual memory, and then the different issues related to virtual memory. And particularly we have elaborate about the translation process, which involves translation of the virtual address to physical address, then we have discuss also issues. Like, whenever you have to perform replacement, then whenever you go for replacement what type of technique can be used, later on we shall use more

details. And in case of write operation, what are the additional complication that can arise that we can discuss.

Thank you.