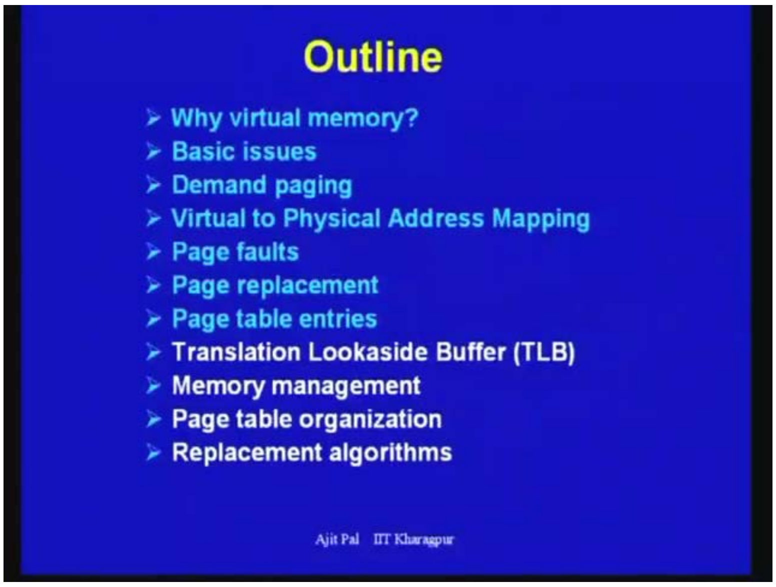
High Performance Computer Architecture

Prof. Ajit pal
Department of Computer Science and Engineering
Institute Indian of Technology, Kharagpur

Lecture - 30
Virtual Memory

Hello and welcome today's lecture, we shall continue or discussion on virtual memory. In the last lecture, I have discussed about I mean various aspects virtual memory.

(Refer Slide Time: 01:02)



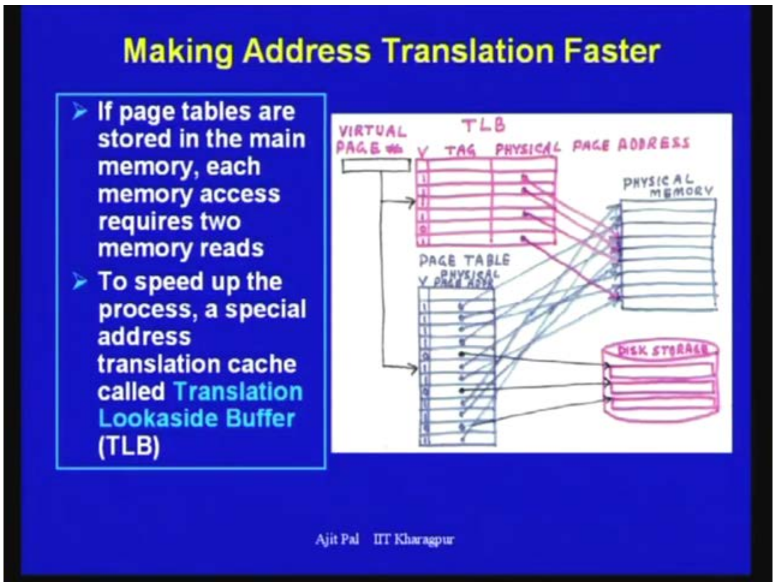
Outline

- Why virtual memory?
- Basic issues
- Demand paging
- Virtual to Physical Address Mapping
- Page faults
- Page replacement
- Page table entries
- Translation Lookaside Buffer (TLB)
- Memory management
- Page table organization
- Replacement algorithms

Ajit Pal IIT Kharagpur

Particularly, we have covered virtual to physical addressing techniques. What is page faults, how page fault is delt with the concept of page replacement, entries of a page table, which have covered in the lecture. In a lecture today, we shall go four important aspects of virtual memory. Number one is translation look aside buffer, which is used to enhance the performance of virtual memory. Then I shall briefly discuss about the memory management, what would really mean by memory management. Then I shall focus on page table organization. Apart from this is straight forward linear organization of a page table. There are other ways of doing it, we discuss, then I should focus on various replacement algorithms, that commonly used in the context of virtual memory.

(Refer Slide Time: 02:14)

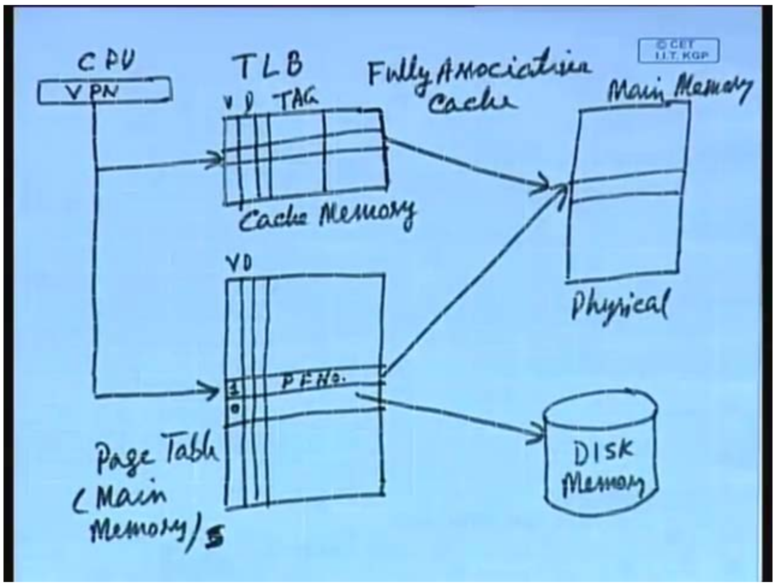

Making Address Translation Faster

- If page tables are stored in the main memory, each memory access requires two memory reads
- To speed up the process, a special address translation cache called Translation Lookaside Buffer (TLB)

Ajit Pal IIT Kharagpur

So, let as start with the translation look aside buffer. As we know we stored page table in the main memory. If a page tables are stored in the main memory. Each main memory access request two memory, we reads as a consequence that takes long time. For example, first we have to get a that page table number. Using that page table number, we have to get a physically, then you able to accept the actual data. Obviously, we take long time, because it involves several memory access, how can it be speeded up. One simple way of doing it is to have a can virtual memory. Let's assume, let see how can be done.

(Refer Slide Time: 03:08)



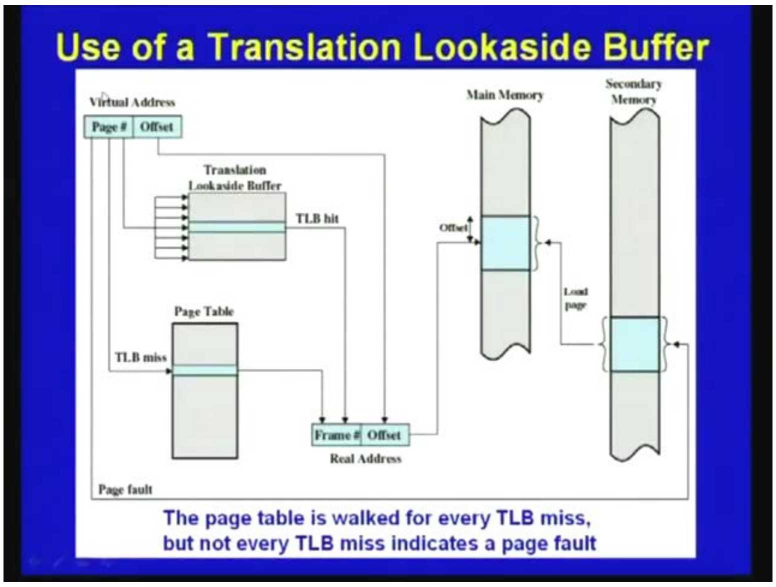
So, suppose this is your process are will generate the virtual page number V P N, which is generated by C P U. Here is the page table, which is commonly accessed and has you know the virtual management page table. I mean the page table there are several entries number one is here, it be then your durty bit. Here is axis stored there and in addition to that we have got the that physical frame number, which is stored that physical frame number is stored here. This is indexed by these particular partial page number. This is used for this is used to get the physical memories, for both these points to the physical memory. So, this is used to point to the physical memory. So, this is used physical memory in addition to that we have brought hard disk.

So, whenever these is not present in the physical memory. It is available in disk memory as you know. So, if it is the valued is one it is present in the physical memory, if it is zero then it points to the this memory. Then we have points to, because in detail in mind in the last lecture. Now, since it is stored in this page table page table is stored in the main memory. Now, instead of doing that if you have a small cache memory is a cache memory. That cache memory is known as translation look aside buffer T L B. Now, T L B will have similar fields. As it is present in the page table above the valued bit the dirty bit and so on. Addition to that it will have two more fields, one will have as you know the required tag field.

Also, you required that physical page address of the same number. That means the the this is indexed by the same virtual page number, but however has you can see, first you can check here if it is available then it points to this. So, it is gets it from here and that case we access will be reduced. We want up to read it from the page table instead of that, we get it in the T L B. So, T L B is a small portion of the page table, which are commonly currently be the used. Those page tables conditions are, I mean those entries that stored in T L B. Since, here will be it is a full availability. It is a is a fully associative cache memory, a fully associative cache.

So, it makes it faster, that means if a you get it from here, however it shown only a sub set of the page table is stored, you may have look at the page tables. Then you have individuality and of course, whenever it is if it is not present in page table. Then that data you have to get it from the this memory. That means it is not present in main memory, then you have to get in a describer. So, this is what is been done, how the translation look aside buffer is used this illustrate with the help of this diagram.

(Refer Slide Time: 07:19)



Use of a Translation Lookaside Buffer

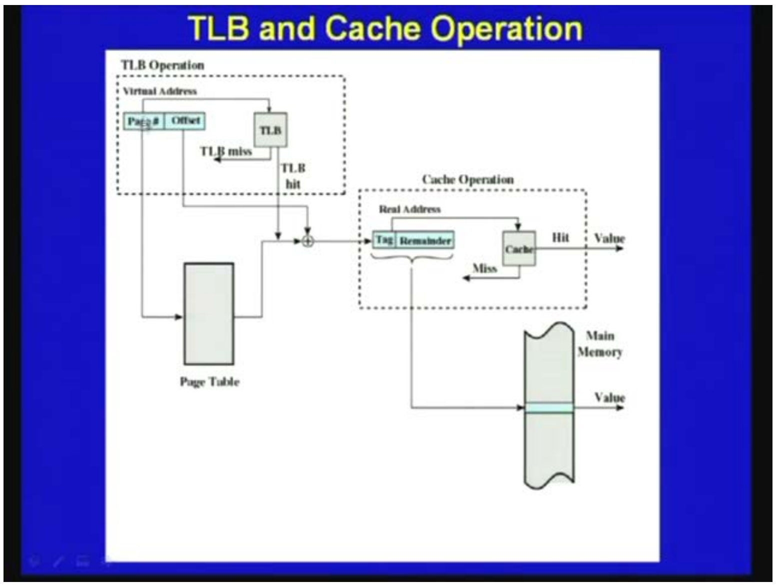
The page table is walked for every TLB miss, but not every TLB miss indicates a page fault

As you can see the virtual address has got two fields the page number and the offset. Page number is used for the purpose of indexing both the translation look aside buffer, also the page table, which is stored in a main memory. So, this is a cache memory, which is present as possible out of the processor. As consequence it will might have be faster. As you can seen first you can accessed and is parallel starts indicating that, it is a full associative cache. If it not present we call T L B will miss, in such a case then it will be indexed by page table. It will provide frame numbers. So, the frame number may come from the T L B. If it is a T L B miss it will come from the page table.

So, page table from the page table we get the same number frame number from the physical memory. This is available and along with that offset is concatenated with to get real address of the main memory. You can see this is the offset part and is the and this frame number start. Then you can load it from the we will get it here. Now, you may not get a that entry in the page table. So, such a case it will lead to a page fault and page fault will have force, you to get the information from the secondary memory. So, you can see from the secondary memory. If the load the page and off course update the various tables. So, the page table what for every T L B means, but not the every T L B means will get a page called.

That means if a T L B will occurred, does not mean it will lead to a page called. Because, that entry may be present in the page table quite natural. This is how the translation look aside buffer works. Let me show you the another diagram, where it shows that T L B and the cache all are put together.

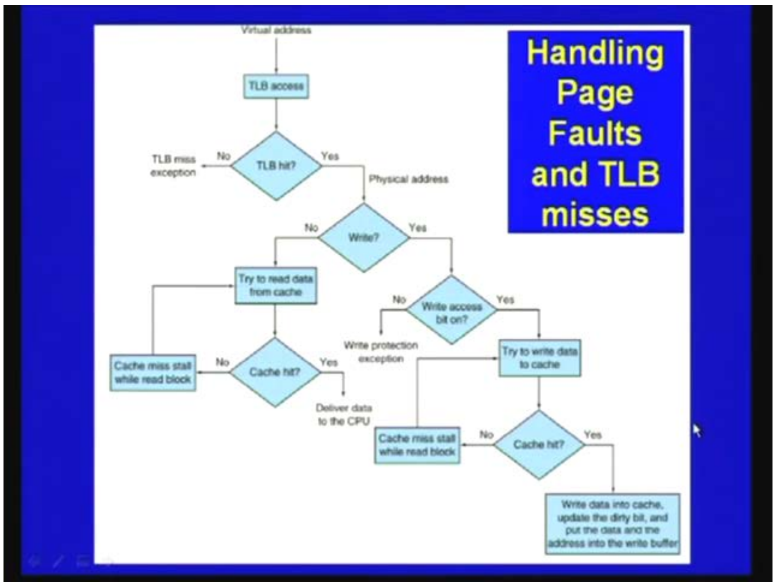
(Refer Slide Time: 09:30)



So, here you can see that partially this is generated by processor giving the page number and offset first it checks the T L B. If it is T L B heat then you get the page frame number frame number and page frame number and the offset tool, there are used to generate the physical address. You can see the and here we get the cache memory. So, cache memory where the instruction and that are stored. You already know the cache memory has got to important fields that the tag. Of course, it will have valued date and other bits will realized and it will remained is used is also available. This is used to get the information and from the cache. So, real this is used to get the information from the cache. If it is a cache head we get the allow from the cache memory.

If there instruction data is stored can if it is missed then of course, it will get it from a main memory and then from the where we get the value. That means this correspond to this cache memory corresponds to data, instruction cache data or instruction cache. It may be either of the two and this cache memory corresponds to the T L B, where a part of the page table is stored. So, we can see you got memory has part of C P U one is steered. The another is the cache, so both are these are is dotingly resident in the has part of the C P U a non chip available. So, this is how the T L B as a cache operates together. Now, this flow chart has gives you how the page faults and T L B meshes are handled.

(Refer Slide Time: 11:25)


Handling Page Faults and TLB misses
Virtual address
TLB access
TLB hit?
No
Yes
TLB miss exception
Physical address
Write?
No
Yes
Try to read data from cache
Write access bit on?
No
Yes
Write protection exception
Try to write data to cache
Cache miss stall while read block
No
Cache hit?
Yes
Deliver data to the CPU
Cache miss stall while read block
No
Cache hit?
Yes
Write data into cache, update the dirty bit, and put the data and the address into the write buffer

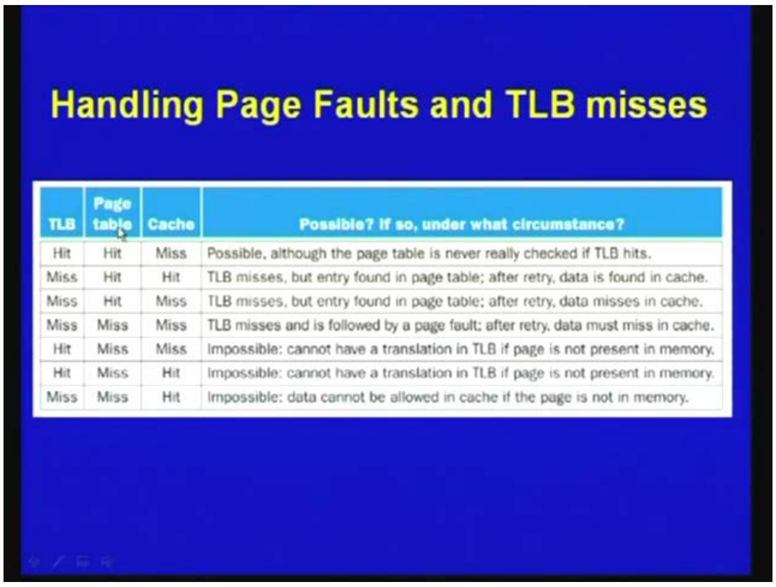
So, the first the actual at this generated by the processor, then the we get the from the T L B axis. If it T L B hit then will generate the physical address T L B means, it generates an acceptation. In such case as you already discuss. Then it will read the page table from the. It will be done by the software by the operating system not by the hard. When a for it is available in the T L B then physical address is available, then that of operation can be a read operation and write operation. If it is write operation, then a it will check the write axis bits.

As an already mention there assign a production bits. It can be read only access only or it may allowed to drive. So, depending on the those track bits, it will aside whether the write axis allowed or not if it is the write axis on, that means write is a allowed. Then it will allowed it to write data to the cash. Of course, that data may be in the cash or may not be in the cache memory. So, in a such case it will shake the cash memory. If it is yes it is present in the cash memory. It will write the data into cash and update the dirty bit and the data and the address in the write buffer.

As you already discuss the use of the write buffer, because these instead of having write through a technique that as written into the write buffer. Then sub sequentially at data is written in the main memory and the other end. If the that write axis is not allowed, then cache means tall while read block, so this happens and the other end.

If we go and the other side, if it is not a write operation in a tries to read the data from cache memory, it is a read operation and if it a hit the cache is then we delivered the data to the C P U, that data may be instruction or data whatever. It may be if it is no, then caching is installed and generated while the block is read from the main memory. Then it again goes back to cache to read a data. So, this is the flow chart, which is used for a handling page faults and T L B in saved all are shown together. Now, let as consider the different cases. The T L B is a whether it can be hit are missed the page table that entry may be present or not.

(Refer Slide Time: 14:07)



Handling Page Faults and TLB misses

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

So, it can be hit are missed and there can be cache memory, where it can be hit or missed. So, the various situations that can occurred as shown here. That means there can be a hit and page table hit and cache missed can we happen or whether it is possible or not, which is shown with the help of the page table. So, it shows that this T L B hit and page table hit it is possible to cache missed. Because, that means all though, the this is possible to also the page table never really shake in the T L B hits.

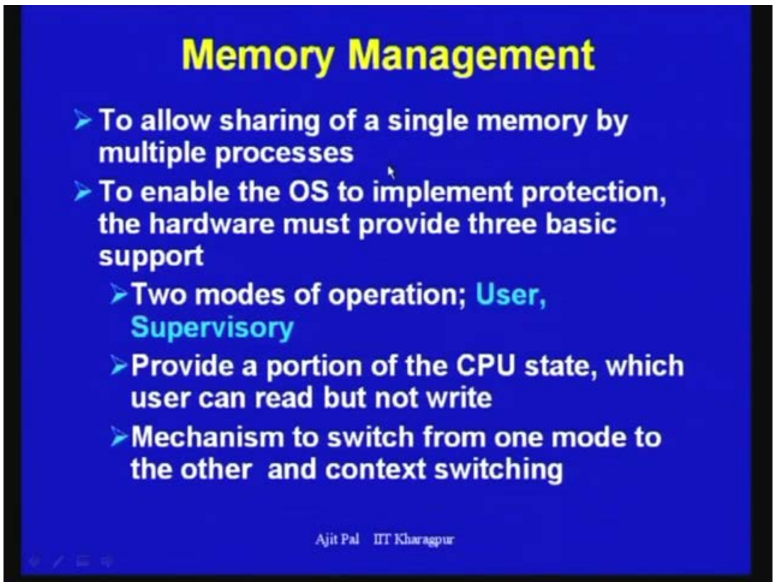
Because, it is not present in the cache memory, but T L B hit it can occurred and page table hit can occurred. Although, the data may not be present in the cache then T L B means and page table hit and cache hit. So, this will lead to T L B miss, but entry found in the page table. That means, since the page table it means occur that end is not present T L B, but it may it has been found in the page table. So, it is that hint is available. So, after retry that retry means after searching the T L B. Then it will search the page table then from the page table.

It will get the address that will data may be found in a cache, so that is signified as hit, so that end in the cache. So, this is in the possible and T L B means page table hit and cache miss so this is T L B miss, but entry found in the page table after retry throw data miss in cache. So, this can also happen now all the three misses, that means T L B miss is page table miss and cache miss all the three misses can occur mean. Such a case it will do a page fault. So, T L B misses is followed by page fault and after retry data must miss the cache. So, that means at A I was never taken from the disk storage to the main memory are to the cache. So, this is the situation where T L B miss page table miss and cache miss will occurred. Now, the let as consider at the three, where we got a T L B hit at page table miss and cache miss.

So, this cannot happen, because it can cannot have a translation on T L B page not present in the memory. So, because of the tuition property we know the data is not present in the main memory. There is no possibility that it will be present in the cache memory. So, because of that this is not possible similarly, here a T L B hit page table miss and cache hit, this is also not possible. Because, that not occur have a translation in T L B page is not present in the memory. So, if it is not present in the memory how there can it be present in the cache. So, this is situation is also not possible then the last situation there is a T L B miss followed by page table miss. Now, there is a cache memory hit it will also not possible and data cannot allowed in to the cache, if this page is not in the memory.

So, that inclusion property is highlighted. So, find that where we have seen situation that can occurred, which we are possible, which we are possible is given by in this table. Now, let as switch to memory management, so after discussing T L B let as focus on memory management.

(Refer Slide Time: 17:55)



Memory Management

- To allow sharing of a single memory by multiple processes
- To enable the OS to implement protection, the hardware must provide three basic support
  - Two modes of operation; User, Supervisory
  - Provide a portion of the CPU state, which user can read but not write
  - Mechanism to switch from one mode to the other and context switching

Ajit Pal IIT Kharagpur

So, basic purpose of memory management is to allow sharing of a single memory by multiple processes. As we already know in a multiprogramming environment or multi tasking environment process of different user, who will be present in the in the memory. That means more than one processors will be active.

So, in a such a situation to in an enable the operating system to implement protection, the hardware must provide three basic support. So, to recipient multi programming at the same time security of the data, there are three possible. I mean three basic supports to provided number one means two modes of operation user mode and supervisory mode. What do you mean by user mode and supervisory mode? C P U the processor can be either the control of the operating system, which is known as the supervisory mode or it can be control of the users program.

So, whenever the control of users program, we call it as it uses mode and the C P U is the under the control of the operating system. We call it as supervisory mode, that means whenever we put the power on the C P U will under the control of the operating system. So, it will not go to the user mode.

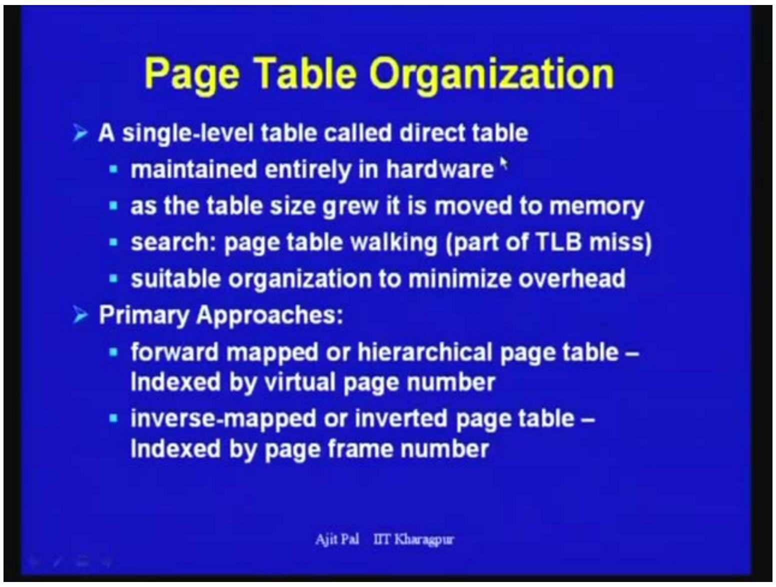
So, two separate modes are provided supervisory mode and user mode. Now, it provide a portion of the C P U state, which user can read, but not right. Some of the some information like page table. So, page table can be modified, when the processor is in the supervisory mode. So, whenever it is in the user mode that page tables cannot be modified is who are the that here will be those cannot modified. There is an fault that is that user should able to modified those table, so that they can again control over some of the pages other users or other processors. So, this is the reason, why this portion of the CPU state, which can be read, but not right. So, you can read the T L B or page table in the use mode but you cannot modified them.

So, this is this profession given in a processor, which allows user and supervisory mode. Then third mechanism will also the mechanism to switch from one mode to the other and context switching. That means as I told begin as you turn power on the processor under the control of the operating system. So, it will be in the supervisory mode, so from the supervisory mode there should be way for changing. I mean switch it over to user mode that means the program counter as to be changed would be loaded by the address of the program.

Some other that also structure at to be modified have an do this switching corresponding to to a particular user. So, this some mechanism provided is it reach from one mode to the other. This is similar to context switching, which we will have depending on multiprogramming. Already known in a multiprogramming environment have a switch from, when the processors works in a time division multiplexing, it is manner and there is a context switching among the user from one another it is a program and so on.

So, this is also supported as part of the memory management. So, these three basic features are available or provided by the system, so that we can have memory management, for control steering as well as protect and of the processor. Now, look at the page table organization, we have so for what discussed it is a single level table called direct table.

(Refer Slide Time: 22:22)



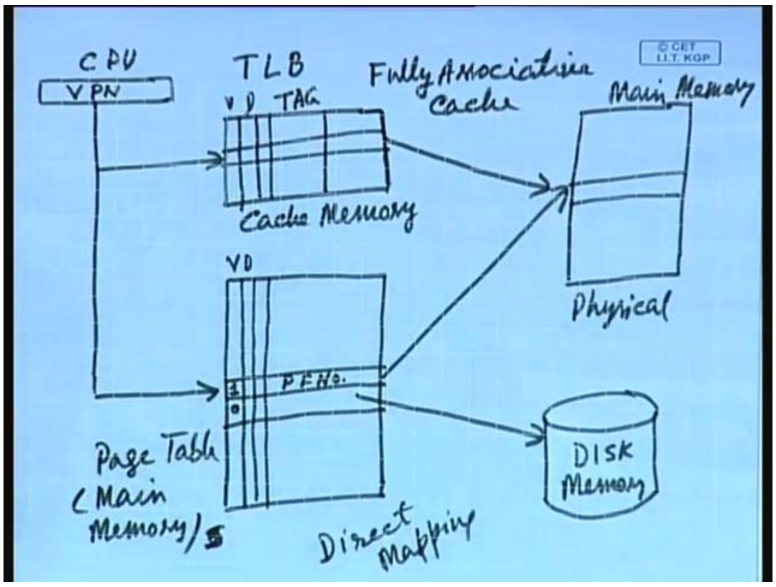
Page Table Organization

- A single-level table called direct table
  - maintained entirely in hardware
  - as the table size grew it is moved to memory
  - search: page table walking (part of TLB miss)
  - suitable organization to minimize overhead
- Primary Approaches:
  - forward mapped or hierarchical page table – Indexed by virtual page number
  - inverse-mapped or inverted page table – Indexed by page frame number

Ajit Pal IIT Kharagpur

So, here we have seen in shown a single label table. So, we may call it direct mapping, so we have got a linear organization page table.

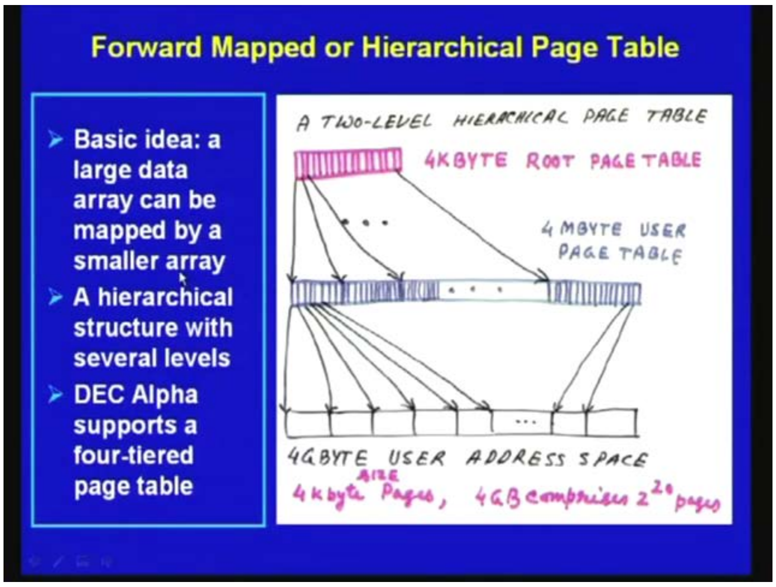
(Refer Slide Time: 22:39)



So, it is linear whenever we do the linear organization of the page table this is maintained. I mean in the beginning it was maintained by the hard ware that means these eyes of the page table was very small in the early years. It was profile data, it was maintained by, I mean it was entirely in hardware subsequently, when the page size grew it is moved to the main memory, these as it is the present situation. Then of course, as you know to enhance the search operation page table working as this is also involves searching T L B and at the T L B miss occurred.

Then you go through the page table and get the that page table number. So, how can we minimize the over head by suitable organization of the page table. For that purpose there are several approaches the first approach. Basically, this approach can be divided into two category, the first one know has forward mapped or hierarchical page table. So, in this case it is indexed by virtual page number. The second word you say second one know has inverse mapped or inverted page table, this is indexed by page frame number. So, first we shall discuss about the forward mapped or hierarchical page table indexed by virtual page. So, this is how it is done.

(Refer Slide Time: 24:38)



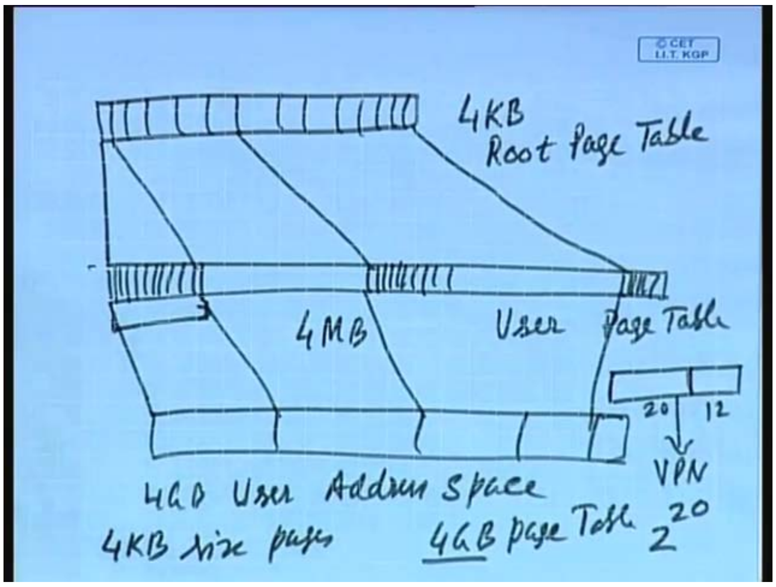
Forward Mapped or Hierarchical Page Table

- Basic idea: a large data array can be mapped by a smaller array
- A hierarchical structure with several levels
- DEC Alpha supports a four-tiered page table

A TWO-LEVEL HIERARCHICAL PAGE TABLE
4KBYTE ROOT PAGE TABLE
4 MBYTE USER PAGE TABLE
4GBYTE USER ADDRESS SPACE
4 k byte SIZE Pages, 4GB comprises 2^20 pages

So, basic idea is a large data array can be mapped by a smaller array and we go for hierarchical page table. So, in the case two level hierarchical page table is shown, what is done we divide the that search into several steps. That means, initially we got a this is know has route page table route page table. So, say 4 K byte, root page table.

(Refer Slide Time: 25:10)

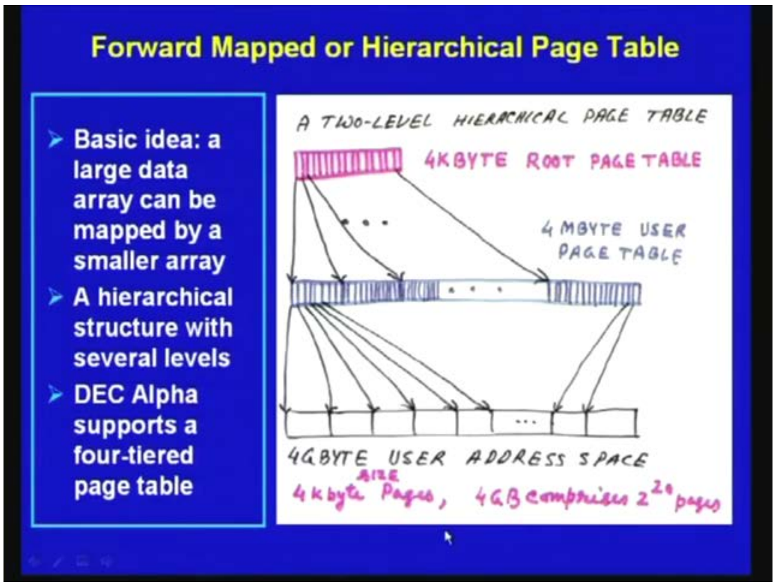


So, instead data how is single linear page table can be divided into two steps one is your root page table. Then the root page table will point to other page tables. So, this will point to and here we got the other page tables. So, this will point to another one and it is done, how shall explain.

So, this is your this is called the user page table. So, in this case 4 kilo byte page table. Then each will point to a 4 kilo byte page table user page table. So, this will together we have 4 mega byte user page table. So, we have got four mega byte user page table, so but is done in two steps instead of a single step. Then will get from here the user address page from get with. So, 4 GB user address space, so in this way this divided. Then each page table is a 4 kilobyte size pages. You have got total of 4 GB page 4 GB, I mean total table size, which corresponds to two to the power twenty pages. So, we have got two the power of twenty pages. Actually, you may call that an this is divided into three parts and earlier it was divide into two parts that frame is 20 bit and 10 bit. So, the is that offset.

This is actual page number that mean that 2 to the power 20 corresponds the, this corresponds to the actual page number V P N. So, that 4 G B is coming from this 20 bit. So, this is, but we are doing into two steps, how it will be done clear from the next diagram. In fact not only can be restricted to two level, we can go to three level and take supports of fourth tiered page table, which I discuss later.

(Refer Slide Time: 28:07



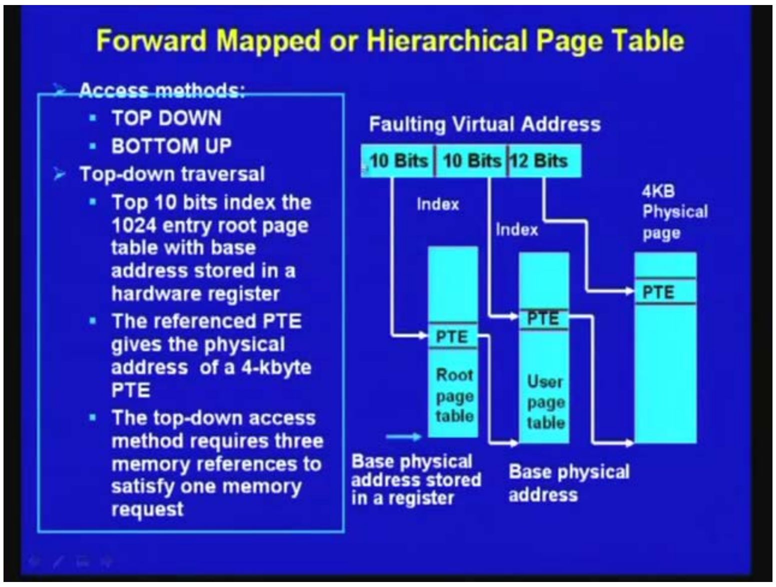
Forward Mapped or Hierarchical Page Table

- Basic idea: a large data array can be mapped by a smaller array
- A hierarchical structure with several levels
- DEC Alpha supports a four-tiered page table

A TWO-LEVEL HIERARCHICAL PAGE TABLE
4KBYTE ROOT PAGE TABLE
4 MBYTE USER PAGE TABLE
4GBYTE USER ADDRESS SPACE
4 kbyte SIZE Pages, 4GB comprises 2^{20} pages

So, how it is done is shown in this particular diagram. Here has you can see the faulting actual address has got 32 bits. It is divided into instead of two parts we have got it is divided into three parts 10 bit.

(Refer Slide Time: 28:18)



Forward Mapped or Hierarchical Page Table

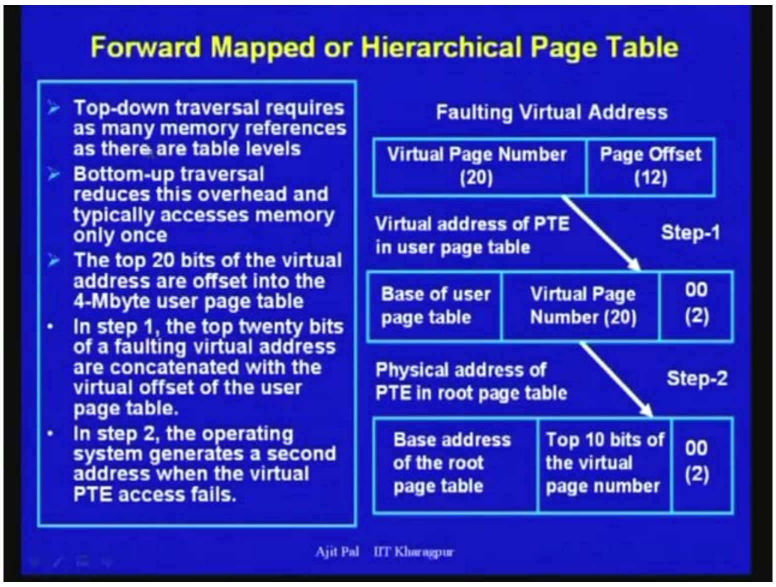
- Access methods:
  - TOP DOWN
  - BOTTOM UP
- Top-down traversal
  - Top 10 bits index the 1024 entry root page table with base address stored in a hardware register
  - The referenced PTE gives the physical address of a 4-kbyte PTE
  - The top-down access method requires three memory references to satisfy one memory request

That part actual page number will be divided into two parts 10 bit into 10 bit, 20 bit. Of course that the page offset that 12 bit shell here with the help of the iyoro dot 10 bit page table is searched. of course, that base of that root page table is stored in a physical register. That means the processor provided with a high hard ware register that base of that root page table is available from that that register. So, from that register this is the index with the help of 10 bit. The 10 bit gives you that page table entry from the root page table. Then this entry this page table entry is used to provide a base of the physical address of the user page table.

So, this will give you the base address with respect to that that 10 bit index part is used to get that page frame number of the user user page from the user page table. Here, you get the complete address that physical address. That is again used for the searching the physical page. Of course, it up to concatenate the page frame number with the page offset, you get the page table. Now, the way it is been search is known as top down access method, you get can see three memory access one for reading the root page table from the root page table. I mean this entry, then it will second time, then you have to access the main memory again to get to user page table. Then you have to get the actual data from the physical memory.

So, main memory you have to access the twice to get the requested data from the main memory. Now, we will be asking the benefit here the benefit is that, in this case you are not storing the entire page table in the main memory. Your storing the root page table and those root page table resending access. So, the total size of the page table that is stored in main memory significant is reduced. So, this is the top down traversal and this is know has, this is how at hierarchical page table organization can done searched in a top down manner. Another approach is there, which is known as bottom up traversal.

(Refer Slide Time: 31:24)

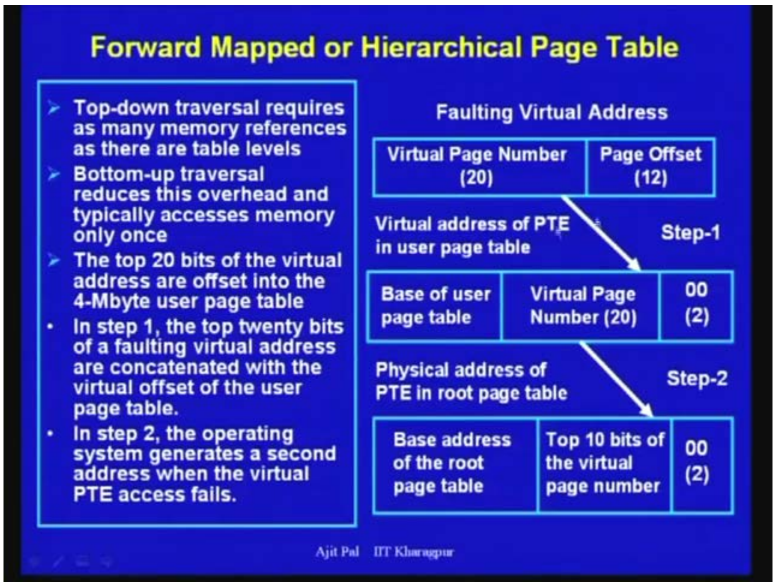


Forward Mapped or Hierarchical Page Table

- Top-down traversal requires as many memory references as there are table levels
- Bottom-up traversal reduces this overhead and typically accesses memory only once
- The top 20 bits of the virtual address are offset into the 4-Mbyte user page table
- In step 1, the top twenty bits of a faulting virtual address are concatenated with the virtual offset of the user page table.
- In step 2, the operating system generates a second address when the virtual PTE access fails.

We have seen that top down traversal request as many memory references as error table labels. As you seen in the figure case we required three memory access, that can be reduced with the help of bottom up traversal. So, it may be possible sometimes not always just having a one memory access.

(Refer Slide Time: 31:39)



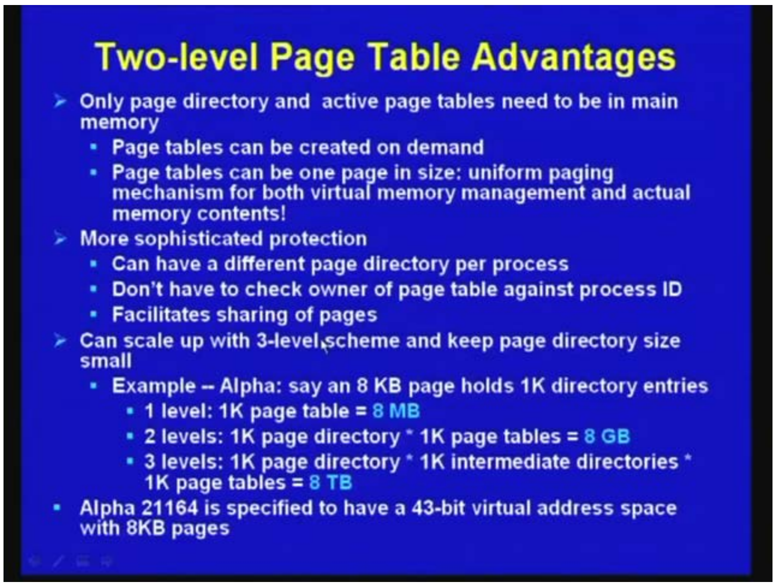
Forward Mapped or Hierarchical Page Table

- Top-down traversal requires as many memory references as there are table levels
- Bottom-up traversal reduces this overhead and typically accesses memory only once
- The top 20 bits of the virtual address are offset into the 4-Mbyte user page table
- In step 1, the top twenty bits of a faulting virtual address are concatenated with the virtual offset of the user page table.
- In step 2, the operating system generates a second address when the virtual PTE access fails.

So, what is done the top 20 bits of the actual address are offset into the 4 mega bytes, it use the page table. Of course lower order bit should be zero, because it is specified in terms of words and the number of bytes in the words is to handled zero. So, this is how if the this is successful, then will be able to do just by reading one memory access and will be able to generate a physical address. However, the step one may not be always successful. Such a case you have to go to step two in step two the operating system generates an second address.

When the virtual page table entry access fails whenever is this travels at, but in this case, somewhat similar to that down approach. Where, top 10 bits of the virtual page number used along with the base address of the root page table. That is obtained by the operating system. That is how the physical address is generated by following bottom of the traversal. So, we can see it may involved at most two steps, but in most in may most of the situations, just one step will be sufficient to get to generate the physical address. That is the reason why this bottom of the approach is convenient and faster than the top down approach.

(Refer Slide Time: 33:48)



Two-level Page Table Advantages

- Only page directory and active page tables need to be in main memory
  - Page tables can be created on demand
  - Page tables can be one page in size: uniform paging mechanism for both virtual memory management and actual memory contents!
- More sophisticated protection
  - Can have a different page directory per process
  - Don't have to check owner of page table against process ID
  - Facilitates sharing of pages
- Can scale up with 3-level scheme and keep page directory size small
  - Example -- Alpha: say an 8 KB page holds 1K directory entries
    - 1 level: 1K page table = 8 MB
    - 2 levels: 1K page directory * 1K page tables = 8 GB
    - 3 levels: 1K page directory * 1K intermediate directories * 1K page tables = 8 TB
- Alpha 21164 is specified to have a 43-bit virtual address space with 8KB pages

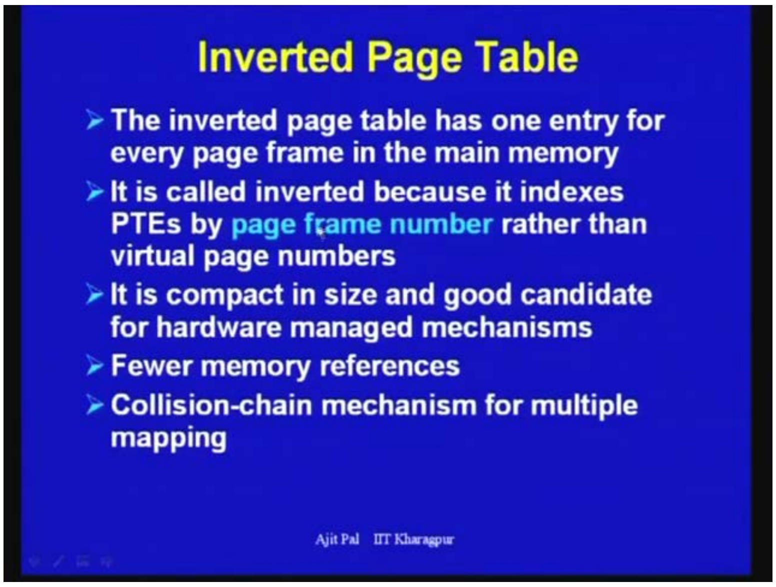
Now, here are the advantages of two level page table advantages only page directory and active page tables need to be in main memory. As I already mention instead of storing the entire page table. If it is done this way then the entire page table as to be stored, but instead of that, whenever we go for this two level page table hierarchical technique. We have to stored only page directory and the active page tables, which are presently being accessed and page tables can be created on demand. So, this is in there is no need to have the entire page table on demand. Whenever the C P U generates address accordingly, the page tables are created in the main memory. So, page tables can be a one page inside in size uniform paging mechanism, for virtual memory management and actual memory contents.

So, this is one advantage and you can use more sophisticated protection. When you use this two level page table hierarchical technique can have a different page directory per process. Do not have to check owner of page table against process I D. It also facilitates controls sharing of pages. So, this is this one of the objective of virtual memory, which is very much satisfied with help of this two page table advantage. Now, this concept was scaled up.

So, this can scale up with three level scheme and keep page directory size small. For example, that alpha take 8 kilo byte. That can take 1 kilo byte directory. So, if you got one level of hierarchy 1 k page tables. As you already discuss we will required total of 8 M B memory size to hold the page table. Whenever you go for two level hierarchy then 1 k directory page table and one k page tables you have to multiply. So, have 8 giga byte page table, but your actual we will stored in the main memory few of them. Because, whenever stored the entire page table, then most of the entry or not really empty, this case does not happen.

So, 8 G B page table can be maintained the smaller size. I mean the page directory size the small page directory size. You can go up to the three level 1 k page table directory into 1 k intermediate directories and 1 k space with 8 k byte pages. So, this is what is used by alpha. So, you see this hierarchical page table organization in there advantages. So, you can go from one or two level or three level hierarchy use. Now, there is an another technique, which is used, which is known inverted page table. As we are already seen the number of, I mean the number of a page table is size of the page table is very large, but very few of the pages are present in the main memory.

(Refer Slide Time: 37:21)


Inverted Page Table

- The inverted page table has one entry for every page frame in the main memory
- It is called inverted because it indexes PTEs by page frame number rather than virtual page numbers
- It is compact in size and good candidate for hardware managed mechanisms
- Fewer memory references
- Collision-chain mechanism for multiple mapping

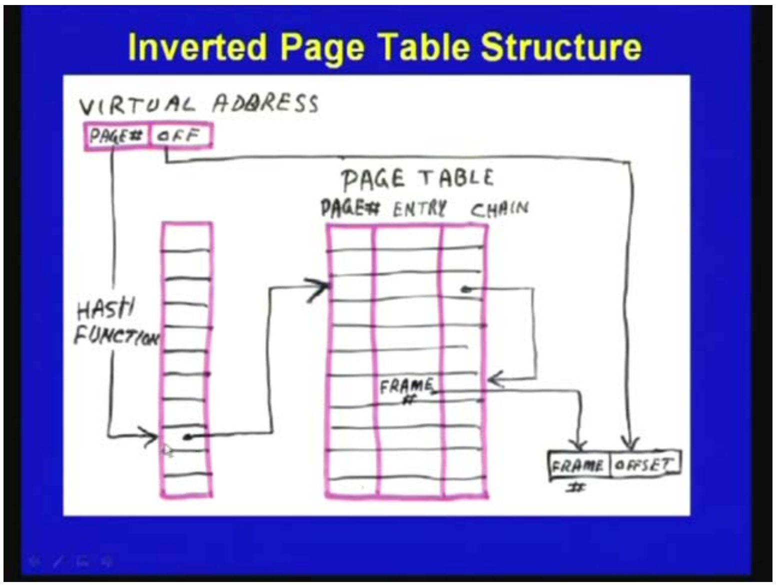
Ajit Pal IIT Kharagpur

Now, can we do something by which only those page tables, which are present in the main memory. Present in those are present in the page table. That means instead of all end page only those physical page tables, which are present in the main memory are those are present in the page table that as lead to, what is known as inverted page table. Obviously, in such a case size of the page table will be quite small.

So, in page inverted page table has one entry for every page frame that is present in the main memory, it is called inverted. Because, it indexes page table entry by page frame number. We have seen earlier it was getting indexing done by the virtual page number, but now what should be doing we shall be using the page frame number. We actually reading from the page table that page frame number will be used. So, it will be completely different from this from this page table number, that we have discuss. So, page frame number rather than the virtual page numbers. So, it is compact in size and good candidate for hardware managed mechanisms.

So, since the entire page table small there is a scope for handling with a help of hardware. Rather than the software, which is normally done whenever we got a conventional page table, this will lead to a the fewer memory references has we shall see how is done. Of course, we will required collision chain mechanism for multiple mapping, which shall I explain shortly, what is collision chain mechanism. First, let as look at the inverted page table. So, here we got the actual address.

(Refer Slide Time: 39:21)


Inverted Page Table Structure

So, you are using a hash function and that hash function is giving you, his providing way he provide he proving you that that hardware page table. As I already mention that page frame number that page frame number is stored here. So, the hashing function is used for getting page frame number, that is being used the parallel indexes.

As I already told the page table contents only those entries only those pages corresponding to. I mean feature present in the main memory. So, these are the page frame numbers of the that the hardware page frame number, that is present in the main memory. So, point of indexes done in this way. So, pushing naturally arising is since we are doing this hash in. There is a possibility that this it will go to multiple entries. That means that same that page number that we are generating, can have multiple entries present in the page table.

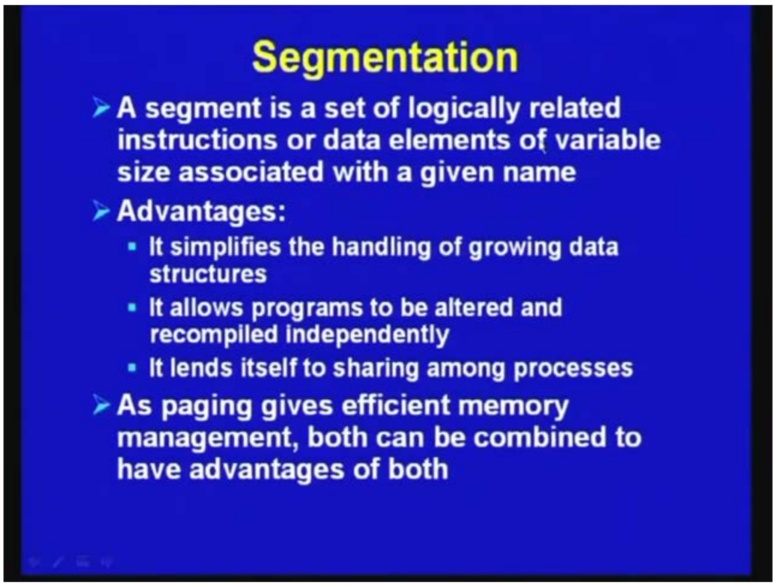
So, how that can be resolved? So, what is being done in this particular case this page number, I mean we do the hashing then you do check the page frame number. So, page frame number is compared and page frame number is compared. If the page frame number does not corresponding to what is being searched. Then it will go to another entry this is known as that collision chain mechanism. So, it will in this way it will keep up it will create a another entry. There is pointer available on the for the first entry. The first entry will you give a pointer and that pointer will stored another entry, where we will get the frame number. That frame is used for storing the that will generative the physical address.

So, in these particular has you seen the total number of entries present in this table will be equal to the total number of pages, that is present in the main memory and has consequence has much smaller. That is the reason in why inverted page table is being suitable and preferable for storing the information of page table, so after discussing that inverted page table. Now, we shall focus on another technique that is called segmentation.

So, for we have discuss about the paging paging is a technique has we have seen that virtual memory is divided into a large number of pages physical, all memories are divided into a in large number of pages. Then particular program or user process can have large number. I mean pages it consist of a large number of pages, whenever it stored in the main memory. Then there scattered depending on the availability at the time replacement depending on, wherever it is available is stored. So, in the physical memory there are stored in a scattered manner, which has know corresponding components with the users program.

So, from the user point of view, I mean user does not know, which pages correspond to this program. It is not available in contiguous memory. So, it is creates a little gap with the user and the way it is stored in the page table. So, another alternative will be used segmentation.

(Refer Slide Time: 43:32)

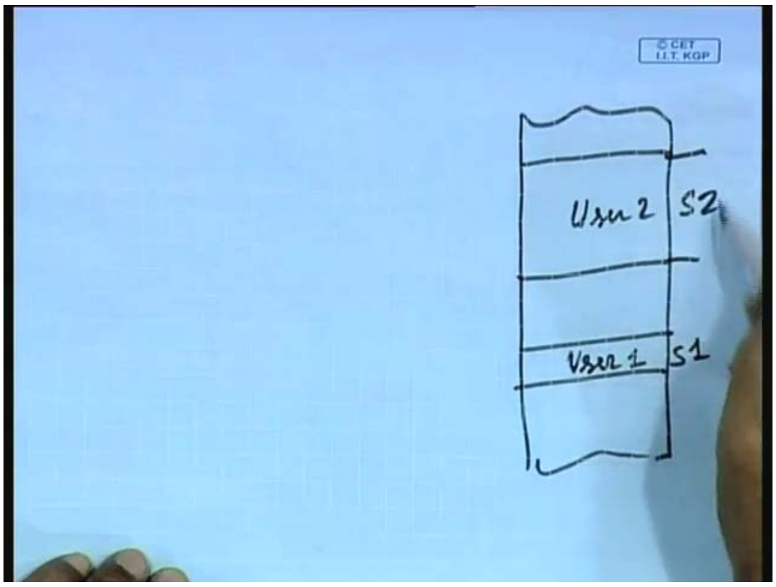


Segmentation

- A segment is a set of logically related instructions or data elements of variable size associated with a given name
- Advantages:
  - It simplifies the handling of growing data structures
  - It allows programs to be altered and recompiled independently
  - It lends itself to sharing among processes
- As paging gives efficient memory management, both can be combined to have advantages of both

So, segment is a set of logically related instructions or data elements of variable size associated with a given name. So, the given name it call may it correspond to the users I D. So, advantages it simplifies the handling of growing data structures. So, what we are actually trying to tell that a particular whenever we go for a segmentation, then this is your main memory, you can have one segment corresponds to one user.

(Refer Slide Time: 44:04)

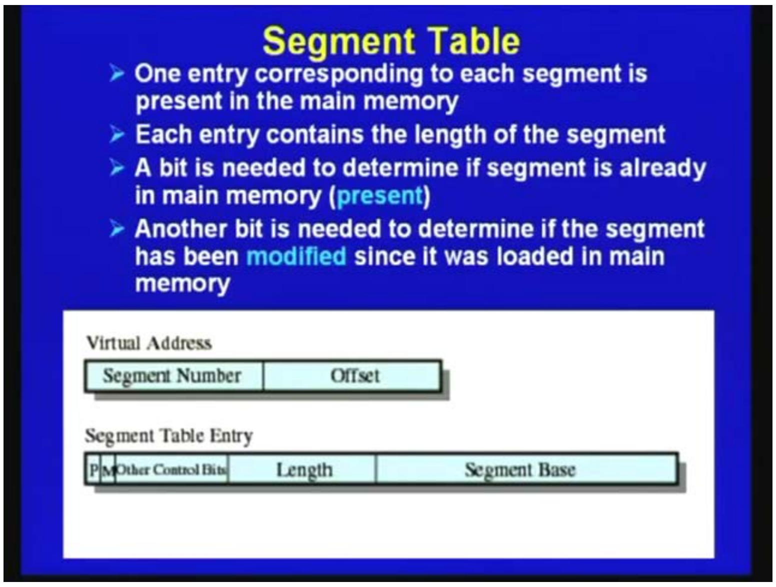


You can have another segment corresponding to another user of variable size. There are in contiguous size for user one this is for user two. So, then this is segment one this is segment two is one segment two. They may be corresponding to different users and as you can see there is no, I mean there are in contiguous memories. So, advantage in this case is simplifies the handling of growing data structure, because we do not have to stored at page table information. Size of the page table is quite large that you do not have to handled. It allows programs to be altered and recompiled independently.

Since, there are stored you know the starting address and end address. That enter program can be recompile, whenever it is necessary can be altered and recompiled, while independently, which you cannot do whenever is paging technique. So, it will lends a itself to sharing among processes, that is also can be done, so as paging given efficient memory management. The other hand segmentation provides you more user convenient. So, it is a paging allows you good memory management.

On the other hand say segmentation allows you to more user convenient. So, how can you combined advantages of both, best both the worlds. That means we are trying to write the advantages of segmentation, that is user convenient. Also, good memory management that is achievable with the help of paging scheme. So, we can combine segmentation with using and how it is done I shall explain briefly. This shows about the segment table one entry corresponding to each segment is present in the main memory.

(Refer Slide Time: 46:02)



Segment Table

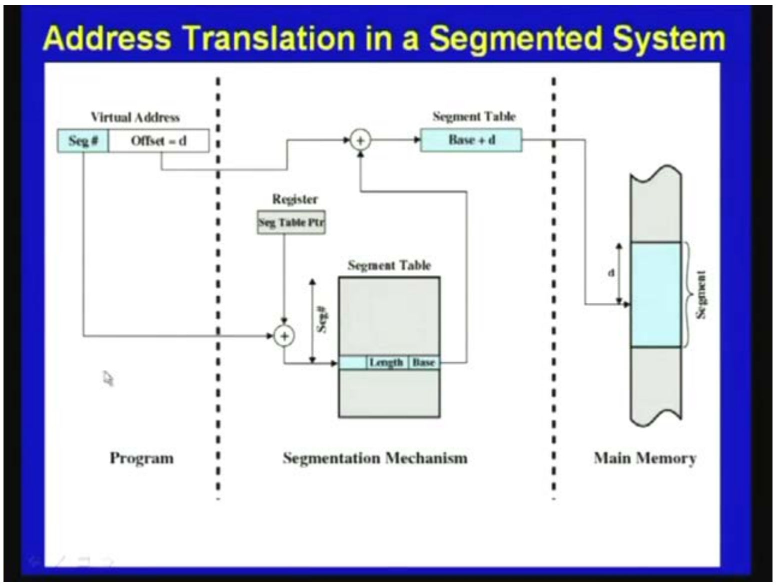
- One entry corresponding to each segment is present in the main memory
- Each entry contains the length of the segment
- A bit is needed to determine if segment is already in main memory (present)
- Another bit is needed to determine if the segment has been modified since it was loaded in main memory

So, just like page table we have also maintain a segment table. However, the size will be quite small here. Because, it will essentially corresponds to number of process that is running in the system. So, each entry contains the length of the segment a bit is needed to determine. If segment is already in main memory and another bit is needed to determine. If the segment has been modified since it was loaded in main memory. Here also those various segments are originally created with respect to the disk memory. They are stored in the disk then from the disk, they will be taken to the main memory.

Whenever, they are being used by the C processor, when it is executed. For that purpose we will required a segment table and the way, it will access the explained here, the processor will generate a virtual address providing segment number and offset. The table will have that segment table will have these entry first up all segment is length of the segment, has to be specified length of the segment.

Then it will have two import two flag bits one is P and another is M, P corresponds to whether a particular segment present in the main memory, whether it is transferred from the hard disk memory to the main memory or not and memory, whenever it is transferred to the main memory. If I tell it has been modified or not it is very similar to that P bit. That is this is used in the paging scheme and the way works is shown in this diagram. So, the processor generates a virtual address that segment number and offset.

(Refer Slide Time: 48:00)

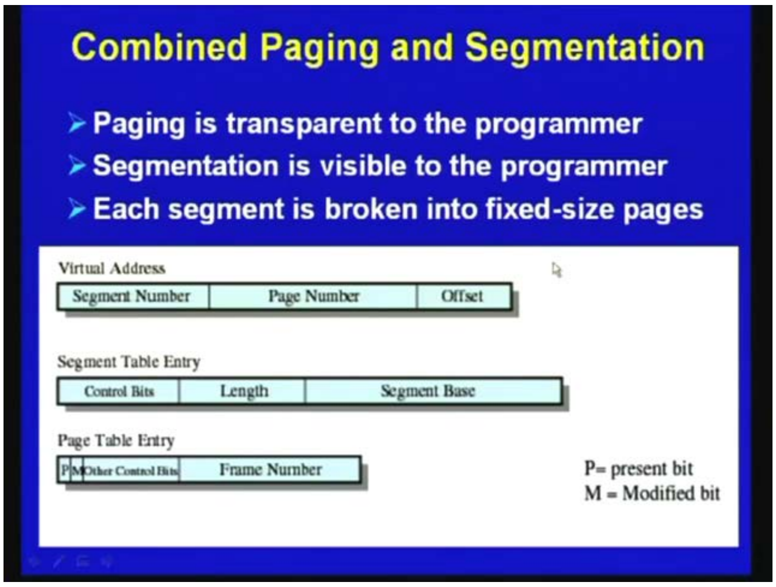


That segment number goes to is applied to the segment table. There is a segment table pointer a hardware register is used to get the base address of the segment table. That is along with a that segment number is segment table pointer is used to generate the segment number, which used for the purpose of the indexing in the segment table. So, segment table as to got I already shown you the different fields, but particularly it will have base and length.

So, base with the help of the base and the offset d, which comes from the virtual address generated by the processor, the segment address, which is generated address corresponds to the main memory address. As you can see this is the segment that is stored in the main memory. As I told the entire segment is stored it as memory and d is the offset with respect to that base address with respect to the base address. This is the size of the segment, this is length of the segment, which is provided here.

So, those violations can be shake F the length is at the address. That this that offset is more than, I mean exceeds the length of segment. Those things can be done and from the main memory you can get the data or instruction in this way. So, this is how the address lengths consist of the equation is done, in the segmented scheme. Now, you can use combine paging and segmentation.

(Refer Slide Time: 49:46)



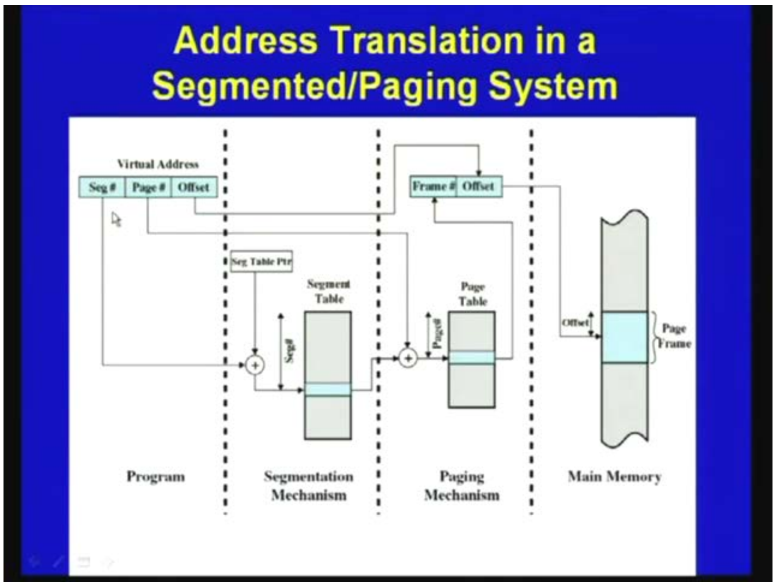
Combined Paging and Segmentation

- Paging is transparent to the programmer
- Segmentation is visible to the programmer
- Each segment is broken into fixed-size pages

So, paging is transparent to the programmer, because it is done by the hardware and the other end segmentation is visible to the programmer, as I already mentioned and each segment can be broken into fixed size pages to get the advantage of the along with segmentation.

So, this is the virtual address here we got the segment number page number and offset. Now, whenever we are using segmentation along with paging we are combing two things segment number and the page number, along with offset and the segment table entry will have the segment page. As usually the length, as usually various control bits that already mentioned and the page table entry will have frame number various control bit. Then the P and the M bits p for trans for present bits and name is found corresponds the number of P bit. How this being done is known in this diagram, stated with the help of the this diagram.

(Refer Slide Time: 50:46)



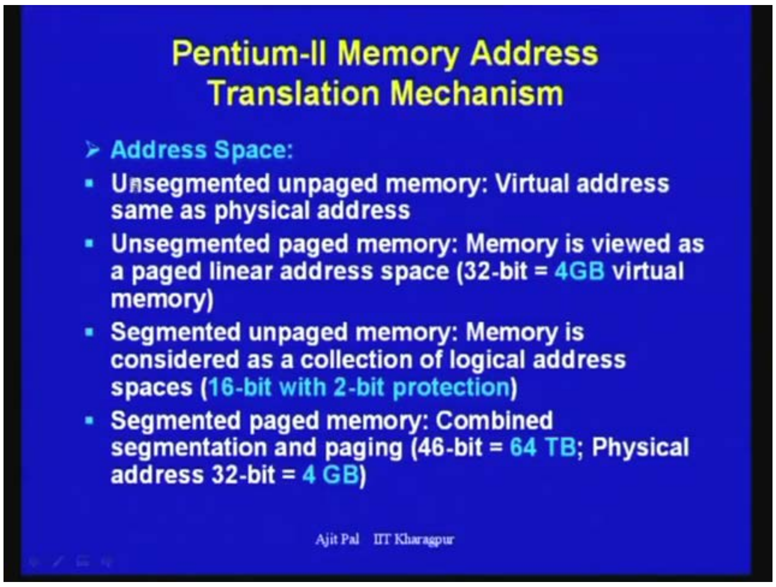
Address Translation in a Segmented/Paging System

So, this is the virtual address generated by the processor combining segment number page number and the offset whose segment number is used for indexing of the that segment table. You can see the segment table pointer and the segment number together is used to get the that. That the segment number and here you get that entry and that is used segment number is used along with the page number to get. I mean purpose of the indexes in the page table.

So, this segmented number segment number and the page number, that is used for the purpose of the indexing from the page table page table, as you know the page number page frame number. So, that page frame number and the offset these two are from used. I mean offset is concatenated with the page frame number to generate the physical address and that physical address.

As you can see this is the page frame. So, here we are seeing a page, now it is segment. So, earlier the entire earlier segment was present here. Now, you have got a page there is a page and within that page the offset is there and you access from the memory. So, you can see we have combined segmentation in paging to be there, in this particular in this way and how the translation occurs. That is shown with the help of the diagram. Particularly, Pentium two memory address translation mechanism is uses the segmentation with phasing.

(Refer Slide Time: 52:27)



Pentium-II Memory Address Translation Mechanism

- Address Space:
- Unsegmented unpaged memory: Virtual address same as physical address
- Unsegmented paged memory: Memory is viewed as a paged linear address space (32-bit = 4GB virtual memory)
- Segmented unpaged memory: Memory is considered as a collection of logical address spaces (16-bit with 2-bit protection)
- Segmented paged memory: Combined segmentation and paging (46-bit = 64 TB; Physical address 32-bit = 4 GB)

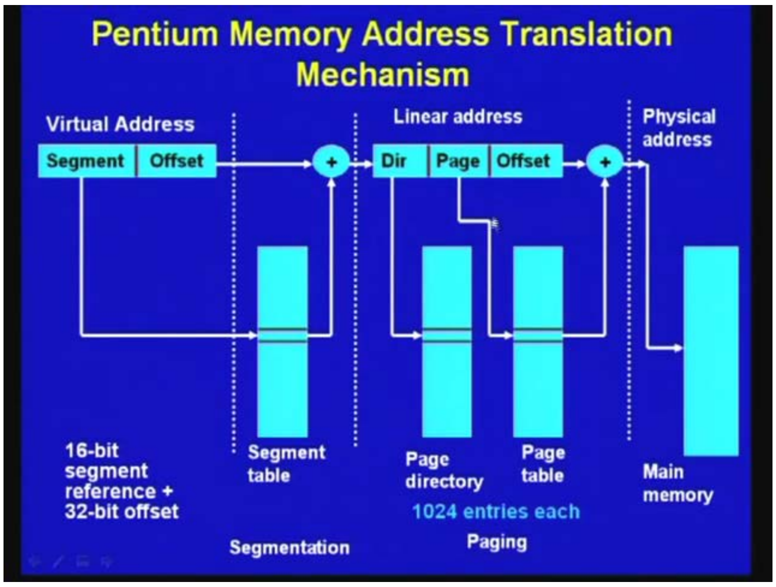
Ajit Pal IIT Kharagpur

So, it gives you several alternatives there you have several alternatives un segmented unpaged memory, where virtual address is same as physical address. So, several alternatives can be used in Pentium two are you can have un segmented paged memory, where memory is used as a page linear page linear at this space. So, you have got 32 bit address, you can have 4 G B virtual memory.

So, this is corresponding to un segmented page memory. Now, you can have segmented unpaged memory memory is considered as a collection of logical address spaces, as it is done whenever are used only segmentation. Finally, you can have segmented page memory where combined segmentation and paging is done of the at the size of this can be 46 bit.

So, 64 4 tera byte physical address is 32 two that is 4 G Bs the virtual address and physical address are sizes are differing, whenever we are using this segmented paging in a combined manner. So, if can have different alternatives is flexibilities is provided in Pentium two memory address translation. How it is done, is shown here, first virtual address generated segment and offset. From the segment table, you get the segmented number that it is concatenated with the offset to get the linear address.

(Refer Slide Time: 53:55)



So, you got the page directory and page table and with the help of the page directory you get the information. Actually, this bit has to be I mean concatenated with page number this particular thing is missing here. So, this as to be concatenated with this page so there is error here. So, this as to be concatenated to get the entry from the page table and this will get this will be concatenated with offset to get the physical address. You can access the memory main memory in this way.

So, you can see this part is corresponds to segmentation at the segment table is used and for the when this part the above for the paging in the paging. You have got the two entry is I already mentioned page directory and the user page table these two are present. Each of them having one K entries that is present. That generates the physical address, where you can access from a memory and this is from the main memory, this is how it done in the Pentium. I think let as stop here today in while. Next, lecture I shall discuss about the various and the fetching policy. Then various other policy that is used for paged replacement. I should discuss in my next lecture along with the way the, this memory organized. So, this memory organization along with those, which shall I discuss in my next lecture.

Thank you.