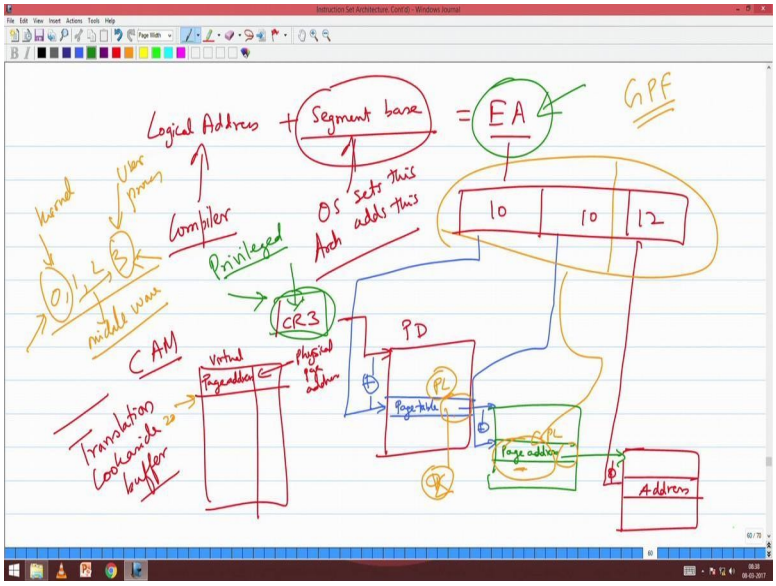
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 29
Part 1
Multilevel Paging

So yesterday we did this multi level paging where in size of 4mb for the page can be reduced to one page directory which is 4 kb, and then as much as page in page tables that are necessary as and when we populate the memory, we can take those page tables that are relevant and the other page tables that which for some addresses which are not used I may not even have the corresponding page tables right and.

So, I could have a dynamically expanding and shrinking you know page table mechanism, which is enabled by this multilevel paging. And now please note that as I told you the page directory the page tables are also put on pages a page directory will fit into one page, each of the page table also fit into one page right and this is perhaps one of the reason why we decided that the page should be 4096 bytes right. So, that I could we could manage all these affairs there and in addition we also saw that the 12 bits that 12 least significant bits need not be populated it is all zeros, because every page starts at a 4096 by boundary. So, these 12 bits can be basically used for storing some information and 2 important information that we put there one was the valid bit and another is the dirty bit we may we may call it clean bit so.

(Refer Slide Time: 01:52)



So, there is a logical address this is actually generated by whom the compiler, this is added with the segment base, this is done the OS sets this architecture adds this we get an effective address. Now in this effective address this comprises 32 bits 10 10 12, now there is CR3 which gives you the physical address of the page directory, this is the page directory. Now I used this these 10 bits I index into this table to get the start address of a page table this will give me the start address of a page table to this I add I use these 10 bits to offset into this, and I get this I get the page address to this I sorry this is 4096 bytes. So, this I add this offset and this is actually my address whatever.

So, this particular location this is the way we go with the page transition mechanism, and to get rid of multiple memory accesses right what we did in this case of segment based we had a shadow resister, here we will have another content addressable memory cam called translation look aside buffer. Translation look aside buffer I look aside the translation I do not want to go through this translation. What this will store? For every page address let me call that that 20 bits right page address it will store. So, I can call it as the virtual page address it will store the physical page address right. So, for every for every input for every 20, 32 bit input for every 20 bit input actually we are interested, this will store this 20 bit address where the page starts.

So, I can just go give this 20 bits as input to this, I will get these 20 bits as a along with you know all these extra bits the security bits, and I can basically take that physical page

address right. So, I get this address. So, to this I can add these 12 bits and directly go and access. So, one shot I will get I will move out of this I will get this 20 bits here, but you when you yesterday afternoon you would have seen that in addition to these valid bit and in the dirty bit, we have also store the privilege levels right. So, as a privilege processor as a processor of privilege level 0, I may have access to some pages if my privilege level is 3 for example, and this page is 0 then I will not be allowed to access this. So, I am enforcing lot of security through the way I can access these pages.

So, in addition to these 12 bits we will have something like in addition to these 2 bits we will have something like the privilege levels stored here. So, I could have privilege levels and these privilege level can be set to 0 1 2 and 3 and as we have seen earlier the 0 will be the kernel process, that 3 will be the user process, this 1 and 2 can be some middle middleware. So, a process can be a user process it can be a kernel process. So, I can set some privilege levels and these pages so, that I can say that if I said this privilege value level as k, any process that k or less than that numerically less than that. Please note that 0 is the most powerful 3 is the least powerful. So, any process with privilege level k or less than k can basically enter can basically access that hierarchy.

If I set this as 2 and a 3 level process wants to access any address right which passes through these 2 then it will be stopped. So, there will be a general protection fault GPF you know you have got about GPF right it is fault number 13 12 or 13. So, there will be a general protection fault that will be you know given basically because you are trying to violate certain privilege level properties. So, essentially what you are saying is that in segmentation we had privilege levels and we are trying to protect segments, similarly the paging mechanism can also be used to ensure protection and security by setting up these privilege levels right. So, we will go do an elaborate discussion on how that 32 bits are organized or how the 12 bits are basically used and then we will see something more in the next Mondays class. So, I will go through that in full detail.

Now,. So, typically what the operating system will do is that there is a, we now we have to start address in paging from a security perspective. So, that we get a better deal of this better overview of this. The paging mechanism the page table and the page addresses all will be set by the operating system, and first before enabling paging the operating system has to set up these page tables correct right it has to create this. So, the first 2 assignments you have done so far, you have not enabled paging there, you have just you

know it was just in the protected mode and you have just used segmentation. Now you have been used for in the absence of the paging you are virtual address space and your physical address space are same there is no translation you are you generate an effective address you generate an effective address has we see here, and the same effective address is basically used as a physical address there is no translation that happens in between. But the moment paging is enabled then this effective address goes through that translation procedure right.

So, there is a very simple way of enabling paging it is to go and set one particular bit in your CR 0 register. So, I think that would that would have explain yesterday. So, you go and set one particular bit in the CR 0 register, and you are paging automatically gets enabled.

So, before enabling that paging we need to go and do lot of things, one of the thing is we need to the operating system has to set up this page tables setup set up this page directory and the necessary page tables, it has to load it into some part of the memory and when while it is doing it is basically paging is not enabled that point. So, it is doing with the segmentation there. So, when you look at in your virtual address space where will this page directory page table etcetera will be stored it will be stored in a segment which is of privilege level 0. Basically your kernel will be responsible for setting up these page directly and page tables right. In other way when a user process in the fourth assignment you will be doing what we call as task switching sorry in your fifth fourth assignment correct. So, no fifth assignment you will be doing the last assignment you will be doing task switching, where you will be moving from a privilege level 0 code to a privilege level 3 code.

So, when you start executing a program what essentially happens in your architecture. So, a shell can be a privilege level 2 or 1 or 0 you have a this shell right. So, now, when you start executing a program on that sell basically then it can it switches to a privilege level 3 right. So, how do you switch from say privilege level 0 to a privilege level 3 so, these are some very interesting things, that we will study in your fifth assignment before that. So, let us assume that there is a privilege level 3 task right. Now the privilege level 3 task cannot go and change these page table in sense what we are assuring here is by having this segmentation, we assure that the page table and the page directories are completely maintained by the kernel, and the user process cannot go and temper with it.

If the user process can tamper with it then all that we are talking about inter processor or intra process protection everything will go for it loss correct right.

So, to start with the first part of your assignment would be that, the first part of enabling paging will be something like you go to the you have a segment which is privilege level 0 in that privilege level 0 segment, you go and fill up all your page directory entry and your page address entry page tables, whatever is necessary for you the operating system has to fit. Then it will go and fill up this CR 3 with the starting address with the starting address of the page directory it will fill up this CR 3 the instruction that can go and fill up CR 3 that is also a privileged. Instruction we all know what a privileged instruction is execute it sets of the page directory setup a stable that is necessary for the program then yes right what is the privileged instruction it can be executed only when the privilege level is 0. So, the kernel is now executing it is sets up the page directory, it sets up the page table that is necessary for that program or that process then what it does it goes and fields of CR 3 it fills up CR3.

Now then it goes and enables paging, what do you mean by enabling paging? You go and make that bit one. So, once that is one immediately what will happen the next address that you are going to next address you are going to put for fetching and instruction or the next address you are going to load for fetching data everything will start moving through right moving through this page translation mechanism, till then you are effective address was equal to your physical address the moment I enable this bit you are effective address now will become translated to this physical address. So, it is starts moving through this translation mechanism. Are you able to follow the procedure how paging is going to be enabled so, that clear.

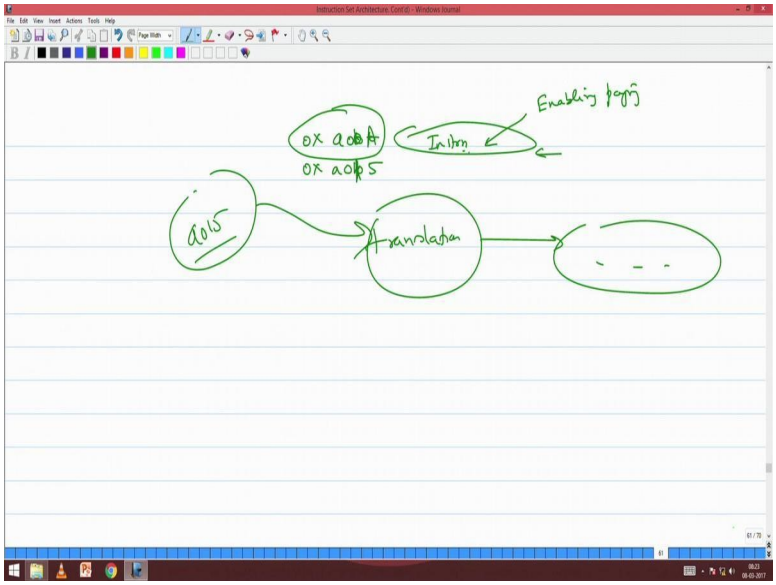
Now, let us so, but one thing even after I enable paging nobody can come and tamper with CR 3, if I want to go and play with that paging first thing is I need to have access to the page directory page table etcetera, but since these are all fixed into a privilege level 0 segment, even at this level when you even generate an effective address and try to access at that point itself architectural will stop you are going outside into the segment. Even before I go through the translation process, your effective address you are generating an effective address and that will be in same segment which is not defined for you. So, you cannot even go and access that you will not even generate because your segment when I am a user level process my data segment and stack segment code segment etcetera, will

be pointing to whatever is allocated to me right. I cannot go and access anything outside the data segment or core segment. So, that is on first level protection. So, I cannot go and touch anything outside of what is allocated to me.

The next thing is can I go and change the CR 3 and make a page table internally within the segment that is allocated to me, that also I cannot do because CR 3 is a privileged instruction. So, I cannot go and change the value of CR 3 nor could I go and change the page table or page directory because all are in segments we basically utilize segmentation here right all are in segment switch I cannot touch right. So, this is how I can ensure that I use a 3 process, it is going to use the page table, but it cannot go and middle with the entries in the page table is it. Are you able to follow right.

Now, let us go next step. So, I am. So, let us say I am. So, there will be one code. So, this is very very important right. So, let me say that there are some instructions which are stored at.

(Refer Slide Time: 16:00)



Let us go with your 0 x a 1000 a 1005 like that right. Now let us say that this is the instruction that is enabling paging so; that means, a 1000 a or a 1000 something this is the instruction that is enabling paging though the moment this

instruction enable paging what will be the pc what will be the program counter after this instruction gets executed the programmer counter. Will be a 1015, what will be the next instruction that is going to be executed will it be a 1015.

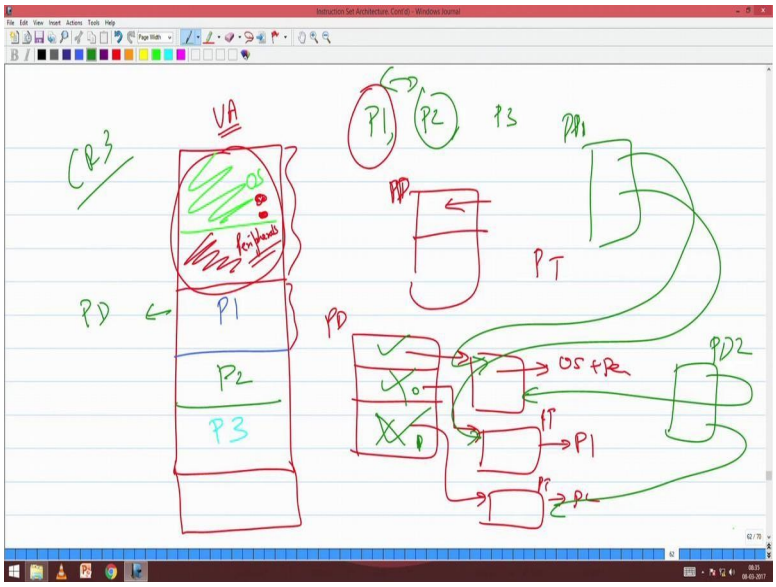
Student: (Refer Time: 16:51).

It will be some a 1015 till now a 1000 a was a 1000 a, now let us assume the next instruction is a 1015, now a 1015 will be pointing to what it will go through this entire translation mechanism, and this may give me some right it may give you something else. So, what should what care should be take. We cannot write programs like that right after this the next fellow will be somewhere I will be in Chennai next fellow will be in san Francisco I can write programs like that. So, what care should we take here?

Student: (Refer Time: 17:39).

Yeah. So, the page in which the instruction that is enabling paging is going to be there that page should be identically mapped right. This is one very very important thing that this is this is a very practical you know point right the page which carries the instruction that is going to enable paging should be identically mapped.

(Refer Slide Time: 18:07)



Now, let me say that a virtual address space will be filled this is filled with OS there will be something for the OS, all the system calls all these things will be there for that then there will be some set of locations for the peripherals, you have already seen this in your third semester right we have some memory allocated for the peripherals, what is that called.

Student: Memory mapped.

Memory mapped.

Student: Io

Io right there is. So, we handle the peripheral as if I am handling memory. So, some addresses will be registered for this. So, those things should be there. So, this peripherals are going to be there. Then let me say I have some space allocated for P 1 then some space allocated for P 2, then some space allocated for P 3, and then there are some more empty space. Now note that when P 1 is executing I could have a page paging mechanism right I could have a paging mechanism where it will see only this and this right, why should P 1 see this P 2 and P 3. So, I could have in my. So, I have a page table right page directory and page table. So, in the page directory I will put pages corresponding to this I will have valid entries here. So, if the if at all my wide why do I need the operating system when I am executing a program? Because I may want to do print f I want to do some system call I want to do so many things, why do I need a peripheral? Because I want to do print f on the screen I want to read and write from a disc I have do an a scan f for a print f. So, all these things I need. So, as P 1 please note this is the basis of how Linux is going to how Linux has developed.

So, as a process P 1, I need to have access only to this OS plus peripheral and this P 1 I do not need access to P 2 and P 3. So, when P 1 starts executing I could have a page directory and a page table setup I could have an exclusive page directory and page table setup within this memory within this OS part. So, the page table page directory will be somewhere here. In which only OS and peripheral section and P 1 section are absorbed or exposed when is switch from P 1 to P 2, I could have another page directory something here different some other spot there which will expose me only which will expose me only this OS and peripheral, but P 2. So, P 1 will not be even exposed. So, as a P 2, I will not be in a position to access that data and thing of P 1; because in you are page directory itself these should be invalidated those pages. So, they will be. So, in the page directory let me say that. So, if I take full fetched page directory in which I have entries, let me say this entry will give to some page table which will point to OS plus peripherals.

Now, this entry will give to another page table which will point to P 1, this will give me another page table which points to P 2 right. When I am creating something for P 1 I will

only make this and this available and I will go and make this valid bit as 0, I just make this valid bit as 0. So, if P 1 tries to access this entry if it is going to generate address which is going to touch anything here, immediately it will give you a general protection format because it is a invalid it will give me a page fault right because it is a invalid page it will be a page fault. Now a page fault handler will go and find why are you going there right it can go and carry are you are you are you able to appreciate this.

Now, when I go from P 1 to P 2 I can do a context switch from P 1 to P 2, I can make this one. So, I will enable this I will make this 0. So, P 2 cannot go and look at this right you are able to understand this. So, I can use one single page directory where I will start as an operating system, I will start playing with the zeros and ones there right. I will just start playing with zeros and ones there and I can see that P 1 will not touch P 2, P 2 will not touch P 1 right are you able to get what I am trying to say yes or no.

Student: Yes sir (Refer Time: 23:27).

It cannot be.

Student: Ok.

So, I do not have allocate the same fault.

Student: (Refer Time: 23:49) in our case like in the lab case memory segment can be.

No no, but that is also not for the single processor you have only one program.

Student: Page length is 4 0 9 6 bytes.

No you are data sorry your data and stack and code are all different segment they are.

Student: But.

But the code of you and the code of him cannot share the same page we do not allocate like that the operating system will not operating system will responsible for it. So, you are writing a code each writing another process his code and your code will not share that 4096 bits.

Student: So, some like in one page it is some parts get allocated to one process, but whole page gets (Refer Time: 24:58).

That is call it page fragmentation we cannot we do not do that, so that minimum 4 1096. So, we will discuss all these things right that is going to be a very say interesting question and I am try to answer right.

Why 4096 right I will given you one answer there are 3 more answers now you understand. So, have to just basically play the bits valid bit and invalid bit to see that P 1 is protected some P 2 2 is protected, now that is also not enough because what will happen is that when P 1 is executing if I just play with this valid bit then P 1 can actually go and find out where P 2 is allocated are you able to follow I do not I want to also make that secret. So, what will happen is for when P 1 is executing I will have a see I will have one page allocated for the page directory alone the pages will be there. So, when P 1 is executed there will be one page directory. So, these pages are there will be one page directory one which will point to this and this when P 2 is allocated there will be another page directory 2 I will just spent one page for just this paging page directory 2 which will point to this and this is not point of view.

So, for every pair every process I could have one page directory right which will point to the relevant pages necessary for that process and as I shift from one process to another process I will also change the page directories because my context which has to be fast I move from you to need to an P 1 to P 2 then that context which has to be extremely fast. Already there are a lot of things when we when we do the fifth assignment we will find out that there are a lot of issues in doing the context switching there are lot of things that will trouble you that will make you slow.

Context switching in general is a very slow process. So, I do not want to add to the latency, but what I am trying to do is when I am contest switching from P 1 P 2 I will have 2 different page directories the top one for P 1 and for P 2. So, when P 1 is executing the page directory for P 1 will be loaded, will be will be used and that page directory will have entries pointing to those relevant pages for P 1 when I move to P 2 I will have a page directory is exclusively which will point to only those P 2 right are you able to get this. So, I can have a combination it may not just top it to the page directory page directory plus a set of pages also I could I could manage it like that ok.

So, now that is why when we move from one process to another process right, when you look at the context of a process what is the context of a process? so, what is general what

is the definition of a context of process context of a process is there is the minimum information that I need to store. So, that I restart the process exactly at the point where I left this answer. I want from you not register right context of a process is that information that I need to store so, that could restart the process exactly at the point where I stopped it right.

Now, when we look when we go to the third fifth assignment right, we will now find out that in the context of a task process we will have this CR 3 also right. So, the question now I want you to go if you are interviewed by AMD or Intel, go and ask why CR 3 is there in that context you ask this question. The reason why CR 3 is going to be there is because the CR 3 for process one and the CR 3 for process 2 are different I will get 2 different page directories and reason why I need to have page directory is, I can now are you getting are you getting a feel of what we are trying to do right.

Now, let us go bit historical about why it is why should I so, the question now is should I ensure protection through paging or should I ensure protection through segmentation. You are already ensuring some protection, and segmentation why this 2 party authentication here why do I go on keep on everywhere OS say security, security, security because it drives the matter right. So, why should I have should I.

So, Linux if you if you are actually studied I do not know whether we going through the x v 6 next semester, I will talk to your instruction. When you actually look at the basic kernel of Linux, the start it will say it will give you one segment which is 4 GB inside there you can read write, you can dance, you can do everything there. So, it is one segment full segment which is some privilege 3 all the protection it ensures it through only paging right it will give you. So, the moment you switch on the system when Linux starts booting, the basic kernel assumes that you have a very large over 4 GB segment which is read write double. So, it I it puts one code segment it is 4 GB it overlaps the data segment on that which is read and write able.

So, it is execute read write everything on 4 GB segment. It will take all the protection gets through this paging. So, between the point where things comes up and by the time this paging comes up that intermediate duration there I have a security venerability, where I can do put my root gate and all these things you understand where I can in fact, a Linux system this is one part I give a open segment which I can do anything with that.

So, that when I am generating the effective address please note that in the Intel architecture, I any other modern architecture I cannot enable paging without enabling segmentation right segmentation is by default right.

Why I now say that I will have one very large segment of privileged level 3 which can be read writable and executable because I do not want anybody to stop anything at this stage. So, if I want execute if I want to do anything that is completely taken care of. The segmentation will not come in the way segmentation has left of let you of do whatever you want right. So, all the protection mechanism etcetera I am getting it through paging and that is perhaps one of the reason why you know the context switching need to have different page tables for different paging translation mechanism for different processes.