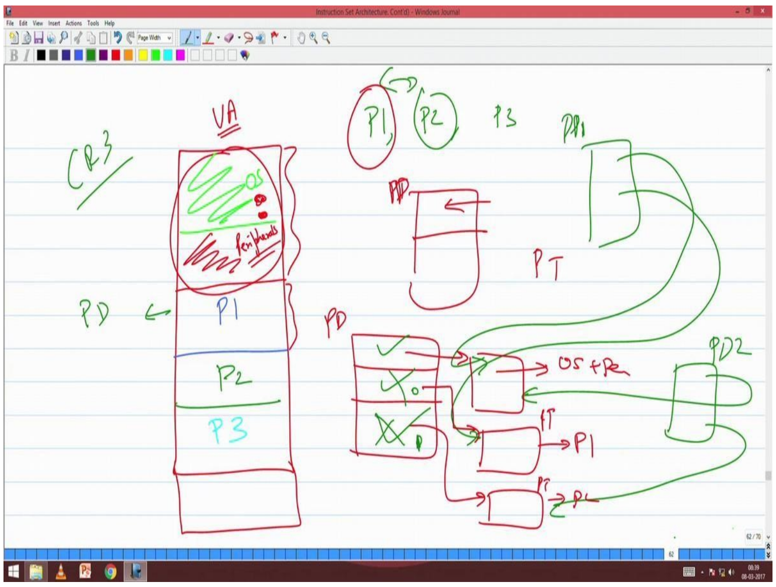
Computer Organization and Architecture

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

Lecture – 29
Part 2
Multilevel Paging

(Refer Slide Time: 00:21)



So, what is what drove somebody to say I prefer paging rather than. Can you guess I will give you two minutes, can you come out with an answer. Why should say for example, we next talk about ensuring you know protection through paging rather than segmentation one simple reason what could be the simple reason. So, most obvious

Student: (Refer Time: 00:46) paging is dynamically

Segmentation also dynamically, it can done. I said OS designer right, look at from your OS designer point of you.

Student: It is more control

Segmentation is more controlled. See there the essential thing. As a os designer what will you do. So, you want to start a OS group. So, you do not want to bind yourself to an architecture correct, correct, yes or no? You do not want really kill yourself with some with respect to an architecture. When I write an operating system you should work on processor a, processor b, processor c, processor d, you cannot write one operating system for each one of these processors correct. Are you able to get what I am trying to say? So that is that is perhaps one of the major reasons why.

If you look at segmentation versus paging and if you finish your fourth assignment after you finish your fourth assignment you will realize that paging is much more generic than the segmentation. See that segmentation here that the 64 bits that you have assigned some 16 bits are here, some 3 bits are there, some 2 bits are here. It is like you know you know like it is if you if you take a cockroach put it in ink, and put it on a paper white paper know, the way it will go and scratch it like that things are discrete that is the easiest analogy I could get correct right that is no rational.

If you ask me why this for a limit bit some 16 is there, some four is somewhere in the middle. So, absolute thing why these things, if you ask this why questions we do not know. We do not have an answer much more than that probably there may be some something there. Now, but when you look at that and this right at least you should do your fourth assignment after that you will appreciate that paging is much more generic than this.

So, if I want to take this operating system and put it on to say spark or into IBM power AX or anything I want to move it onto the next you know architecture power or spark and whatever it is are arm right. If I am going to investment I am on paging then for me to go and do that paging say everything has a multi level page box. So, for me to go and switch these operating system for you know the next architecture, I will be more happy to play with the paging, the effort would be much less it depend upon the paging mechanism rather than the segmentation. Segmentation will go manually, you have to rewrite the entire operating system well.

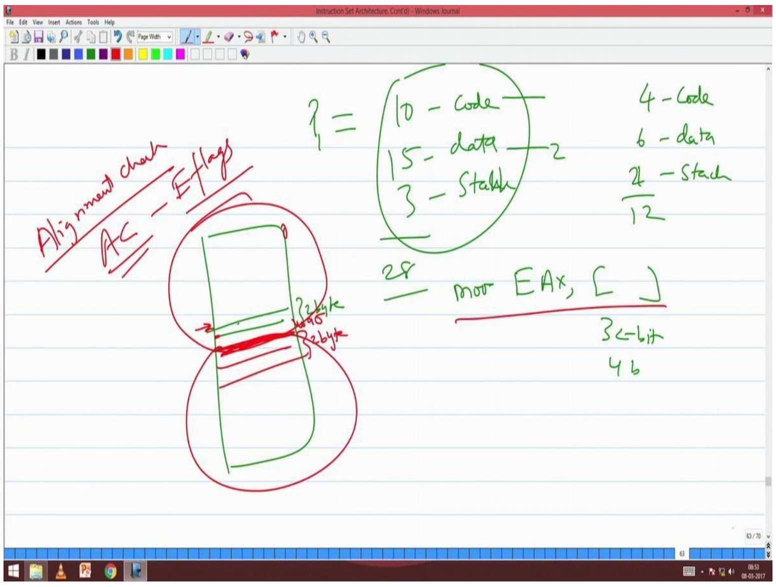
Now, What is after all what is operating system the major thing there is memory management right process management is trash it is not that great. The entire whole thing is memory management because everywhere I see as I told you right from execution of an instruction, there are five stages, three stages that is memory. When I want to start execution of an instruction when I want to start the process, I have to allocate memory. And when I want when I look at protection what is protection, you should not read my memory it is not that you should not use the CPU or anything. You can share everything else in that life there, your general purpose registers can be shared, your CPU can be shared, ALU can be shared, bus can be shared, peripherals can be shared, one thing you cannot share is memory right.

The whole operating system is basically governed by memory. Today, the whole problem of information security comes because we have memories, because there is some loop whole in memory management. Are you able to appreciate these facts? So, that is the so operate the strength of an operating system basically comes from how effectively they could handle memory. So, when I move from when I have invested so much time on developing an operating system for an architecture, and I want to move to next level architecture, the major term in that flush would be memory management and that I want to say that. So, when we look at segmentation as a memory management principle was a paging as a memory management principle in the context of x 86, we find that paging is more generic than segmentation that is why we would like to invest more time on.

So, before we wind up today, let me just cover one more topic and then. When a process comes to execution now I give it say some I am operating system how many pages should I allocate for it for that. When compile I know the size of the code and all I know the size of data it will take. So, I will allocate that many pages as much as that many data use that. So, that that page allocation will happen in the virtual address space.

Now, what will happen is when I want to execute I am doing demand paging. So, I have to move from the virtual, I have to move the pages from the virtual address space to the RAM. In the RAM, suppose I as a program I have say I need say 5 pages of code, and say 6 pages of data, I cannot allocate 5 and 6 there then that means, your virtual address and effective address and physical address becomes same. So, I will allocate some amount of pages. I will say how say k pages for code, k page frames for code, some r page came some data. You need some 10 pages, your codes fonts across 10 pages, I cannot give you 10 pages in the main memory right I will only give you say 2 or 3 pages. So, you use these 2 or 3 pages in the main memory and keep moving your page frames here and there and then try and execute it.

(Refer Slide Time: 07:16)



So, as a process p 1 my code sponsor gross say 10 pages of code, 5 pages of data actually 15 pages of data, 3 pages of stack this is my full requirement. But in the thing I will go and say this is your size, so I will give you 4 pages of code, some 6 pages of data and say 2 pages of stack. You use these twelve pages and execute this 28 page program so; that means, as and when you will keep replacing pages you will take page. So, the number of actual page frames allocated for your program you has to use those page frames and keep shifting your pages in an doubt to see that your program is go to execute this is right.

So, as a process I will need so many pages. As an operating system I will give you. So, many page frames use these page frames to execute a program. The number of page frames that I am allocating may be proportional to the number of pages actually you require sometimes it is so. So, that is again there is again hundreds of papers written you know and fifties of students of got Ph. D and tens of processors will got ten you track by this it with that study this relationship, but that the thing is very, very simple. Try to understand this as a concept it is just I am giving you k frames, this is the k frames, this is you are requirement k pages for your requirement some r frames are given. Use these k r frames to execute that k page program.

Now, I will just ask one question and stop here. Why should I give two pages at least say for data or instruction, why should I give two pages at least, if give one page what will

be the problem? I am giving one age, so use that page for code one page for code one page for data, one page for stack.

Student: Suppose in our (Refer Time: 09:28) one page for a frame, directory.

That is ok, that is, one page directory and one, one, one - four pages I will give you. One page directory, one for code, one for data, one for stack, yes, question.

Student: If you have to do two different operands and you needed then if they were in two different pages then you need certain (Refer Time: 09:55).

Two different operands ok, a very simple thing. Let me say I am accessing a say EAX comma some address right move. So, suppose this 2 bytes of so this is a 32 bit right 4 byte. Let me assume 2 bytes are in this page, and the remaining 2 bytes are in this page, and this is the page boundary. Now, I have allocated only one page for data. So, I will be loading this. And when I start executing this command what will happen it will get the 2 bytes, the remaining two bytes it is a page faults are.

Once I get a page fault, what I do I go to the page fault handler, I want only one page for data. So, I will go and load this and start re executing this instruction then these are page fault again. So, I will go. So, I will be I will be just ping ponging one page will come here, again it will go another page will come this move will never complete this is the very, very simple obvious reason why I have to allocate at least two pages. If I do not allocate two pages, then what will happen I have these 4 bytes, 2 bytes on one page and exactly on the boundary or 3 bytes there, 1 byte here or 1 byte there and 3 byte here just form it across.

So, when I try to access that one data page will be there correct I will get that data page now I will find only one byte is there remaining 3 bytes, there is no provision of keeping 1 byte and saying come next. So, I will just cancel that instruction because that instruction has cost me a the granularities in one instruction right this instruction has cause me a page fault. Suspend that instruction store that PC, now I will go and execute the page fault handler, again come and start that instruction again, then it will find out that again 3 pages are remaining. So, that is perhaps another reason why there is something called alignment check AC flag.

If you go and look at your E flag register there is something called an alignment check flag. Alignment check essentially says that you have to align all your 4 byte access on 4 byte boundaries. So, this basically will not allow you this basically will not allow you to have something in a two. So, this will happen if this particular variable starts at a 2 byte boundary, this is because this is zero to 4095. So, if I have a 4 byte variables starting with 2 bytes here or 3 bytes here, 1 byte here it is not actually starting on a 4 byte boundary.

So, when you are compiling, you do when you are compiling and executing for the first time, you put this alignment check flag to check where and all these type of violations are happening. And you go and take care of that violation essentially you put some dummy bytes inside to see that all these are aligned properly. If they are aligned, then this type of an issue not come. So, alignment check flag I will also talk in the context of memory management little later, but this is very, very important. I think I talked about alignment check in the third semester of course also, but we will we will cover it little more.