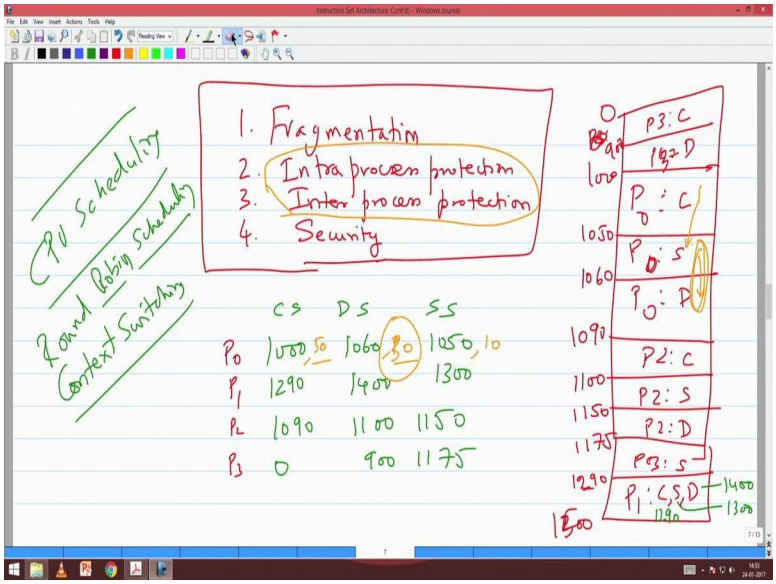
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 11
(Part – II)
Lab 2: Segmentation

(Refer Slide Time: 00:25)



Now what are the other big big benefits? So, we will talk about 2 important things, one thing is lot of thing. So, first thing that I will cover today is fragmentation, number 1 is fragmentation what is fragmentation? This is an operating system concept we will come to that; 2 - Intra process protection; 3 - Inter process protection 4 - Security.

So, four important aspects we need to study as a part of segmentation these are the four things in addition to the position independent compilation. So, compiler had an issue and that is already solved. Now these are all these four are basically operating system issues and how will segmentation address this is what we are going to talk about. Now let us take this, I am now working with a big server now I have process zeros code P 0 1 minute sorry P 0s code is here P 1s stack is here sorry P 0s stack is

here, P 0s data is here let me call this 1000, 1050. So, this starts from 0 1060, 1090.

Right now let me say (Refer Time: 02:22) is a total is 2 1200 1500 total memory is only 1500 let just for a. Now I have P 2 code P 2 stack and P 2 data. So, let me say this is

thousand 1100, 1150, 1175, now this is P 3, P 3 code. So, this is 900, P 3 data and P 3 stack and this say 1290. Now this entire thing is hold at P 1 code stack and data are held in this let us see. So, this is how the memory is there are four processes P 0, P 1, P 2, P 3 and this is how these things are loaded. Please note that when P 0 is executing P 0, P 1, P 2, P 3.

So, let us see let us also have P 1 will be 1 2 code will be 1 2 9 0 stack will be 1300, data will be 1400 something like that. So, when P 0 is executing c s will be 1000, s s will be 1050, d s will be 1060.

Now, when I want P 1 to execute when P 1 is executing, this will be 1290, 1300 and 1400. When P 2 is executing will be 1090, 1100, 1150, when P 3 is executing it will be 0, 900 and 1175 are you able to follow. So, when I move from. So, what will happen is today when you login to your g mail server right suppose your friend also logs in both should be given some fair amount of service if I say I will attend to a friend then to you then what will happen then you will never log out you will never log in. So, is it is basically what it means that you are when you click something happens on your screen your part of the program gets executed. So, when I open there is a session open there is a process created for you in the mail server when he or enters again a process is created for him.

So, both these processes are given some time on your CPU. So, I will give some five units of time whatever that unit, throw you out then give him five units again give you five units five units. So, both of you will have some feel that you are being served. So, this, so I am allocating a process to your CPU. So, this is basically called as CPU scheduling.

And this you will learn extensively in your operating system course right. So that means what? Your processors are also started executing her processor also started executing both of these processes are loaded into memory, then what I need to do is I need to switch from 1 process to another process to another process to another process and again come back and do, this type of a scheduling is called round robin scheduling why robin is that robin hood right. So, who said fellow should not have all the mnemonic money took away the money and then distributed to the poor right.

So, something like you know equal distribution all those things are attributed to Robin Hood right this is also equal distribution that is why this name round robin algorithm right. So, so what is round robin? So, I will execute I will allow you to execute sometime pull you out and allow her program to execute then the next one. So, P 0 will be given some time it will be pulled out next P 1, it will be pulled out next 2 it will be pulled out P 3 again say P 0 P 1 P 2. So, like this say round robin will happen, you will learn lot about scheduling and especially you know the Linux scheduler the fair scheduler of Linux is very very interesting this you will cover very deeply in your operating system course.

But what you should understand is the architecture has to support this. So, when I move from him to him, one process to another process that is actually called as context switching. We will read more about context switching here context switching need to be understood at the computer organization level yes. So, that when you go to a operating system you say I switch from this process to another process you will immediately appreciate what is context switching right. So, we will we have one dedicated assignment in this course which is for context switching which will just deal context switching right what it means to context switching, and there are lot of security implications there we will also cover those things in great detail (Refer Time: 08:29) ok.

Now, in this case of context switching for me it becomes easy with segmentation because when I want P 0 to execute what I need to do I have to load 1000, 1060 and 1050 into the respective result. When I pull out P 1 and I want P 2 to execute what I need to do just go and change the segment registered value I need not go and change anything in the memory I load everything into the memory and keep, all the process that are all the programs that are in execution namely the process is all its data stack and code I can load it in memory and I keep, whenever I switch from 1 context to another context I have to just go and change the value of the segment registers right.

So, context switching context switching means switching from one process to another process, the context switching essentially becomes extremely simple. So, segmentation helps you in that context switching. Now what will happen is the next thing that segmentation will also do is that along. So, what do you want this is just a base address right. So, now, I will go and say I will have 50 bytes here, I will have 10 bytes here sorry 10 bytes here I will have 30 bytes here what is this 50, 30 and 10 are you able to see color all over is good; what is 50, 30 and 10? Limit correct this is a limit. So, along with

the base I will also store the limit. So, that is if this P 0 fellow goes and start executing beyond 50 bytes right if the offset is more than 50; that means, I am going to access something. So, the limit I store the limit. So, that if I go outside that limit then I am doing something wrong correct I should not go outside the limit.

When the program compile it said I need only 40 bytes for all the instructions sorry sorry 50 bytes for all the instructions. So, I will have given that 50 byte, you start more than 50 if you go and say I want more than 50 bytes then what will happen?

Student: (Refer Time: 10:53).

Right then I am wrong, I do not have an instruction beyond 50 bytes from start because my entire code segment was only 50 bytes in size are you able to follow. So, if I try to execute beyond 50 bytes or if I go and access data beyond 30 bytes in this case as you see here or if I am going to access the stack if I am going to go the stack beyond 10 bytes then somebody has to catch you and say hay it is wrong. The operating system needs this help from the architecture right with other there is what will happen is he will come and peek into your mobile into your mail box and he can take your assignment because it say data segment the mail box say data segment. So, friend some friend of your can take it and submit it as his assignment and say you copied from him all these things are possible so.

So, essentially I need to save one fellows data segment from another fellows data segment right. So, if I am going to access anything beyond my capability beyond what I am allocate allocated at sorry what I am allocated, allocate at then I should be caught who will catch me?

Student: Operating.

The architecture should catch me not the operating system how can the operating system catch me? There is only 1 CPU that is executing your program is executing on it, operating system is also another software it is not executing it is dormant correct. So, when you are doing something wrong the operating system is not even existed exist at right it is it is dormant it is not alive it is not in the it is not having control over the CPU there is only one CPU in that CPU your program is running the operating system is resting in the memory. When P 0 is executing P 3 let it be the operating system it will be

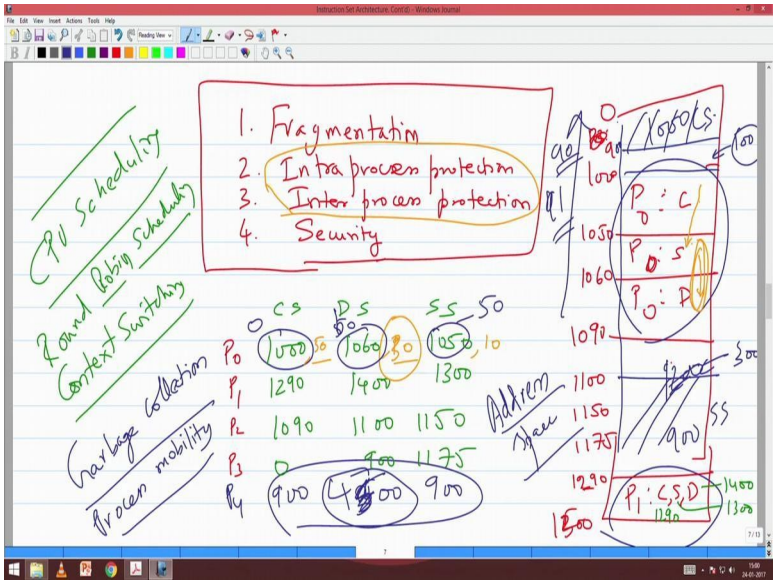
resting in a memory. Are you able to follow, right. So, if P 0 does a mistake the operating system can never catch it on its own. So, it will request the architecture [FL] is this fellow does some mistake please tell me.

Right there is an exception right. So, along with your segment register base I also store the limit and if I if and that is given and when the program is executing it either crosses the limit of either of these 3 segments code data stack, then immediately the architecture will raise an exception and the control will be transferred to the operating system. If you go and do a mistake hey stop it will catch your neck pull you out, put the operating system (Refer Time: 13:43) and say sir caught by miss right. So, this fellow has done something wrong I am just bringing back your LKG memories right this fellow has done something wrong. So, take this him got it right. So, that is what we call as inter intra and inter process protection, I cannot overgrow my stack for example, if this fellow overgrows his stack and start writing into the data please note here I am putting the pen here, if this fellow moves from the stack P 0 over grows his stack and writes into data and that is also corruption.

If P 0 starts executing from the stack it keeps executing instruction of to instruction and this stack executing from the stack that is also corruption right. So, intra process protection also I give you, and P 0 cannot go and access anything beyond what it is assigned 1000, 1050 it cannot execute beyond 1000 2049, it cannot access data beyond 1090, 1089. So, these 2 things are basically ensured these 3 things are basically ensured by segmentation right.

Now, we will talk about security last before that I want to tell you let us say now P 2 finishes. So, P 2 process is over. So, P 2 goes out. So, this memory becomes free and let us say P 1 P 3 also finishes. So, this memory also becomes.

(Refer Slide Time: 15:36).



So, what is the total memory available? Total memory now available will be this is 1000, 1090, 1075 would be 75 plus 10 85 right. So, 95 ya sorry, 1290, 1292 know 1200.

So, totally I have 2200 bytes of memory suppose I am getting now a fact fat process P 4, which is asking for you know 900 c s and 900 c s and. So, 400 d s and 900 stack sorry 500, 500 d s and 9 (Refer Time: 16:45). Now what will happen is I can I have 1000 here. So, I can put 1 c s here, I can put c s here. So, I will have 100 bytes here. Now the remaining I need 1400 I have only 1200, here hmm sorry sorry sorry this can be 400 ya 900 c s, 900 s s and 400 d s. So, I can put 900 there I will have 100 bytes left here, I put 900 here I will have 300 bytes left here. 900 I can full fill it here and I can fill 900 here. So, I will have 100 and 300. So, I have 400 bytes available, but that 400 bytes I cannot allocate to c s d s.

I have allocated c s here I have allocated s s here. So, 900 and 900 bytes now there are four hundred bytes are free in the memory, but they are not in contiguous space getting what I am saying yes or no.

Student: Yes.

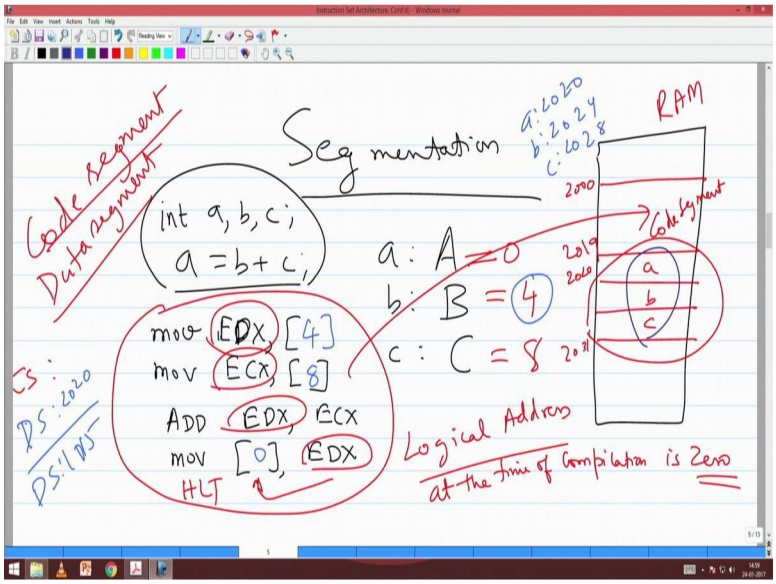
Right. So, I cannot allocate that 400 because I need it to be contiguous. So, I not I need to do now I do something called the operating system suddenly finds, hey there is so much empty space, but I cannot give that contiguous space. So, then it goes and does

something called garbage collection demeaning right. Suppose your program just executed just before it and now the operating system calls it garbage. Now what is garbage collection? Now what I will do is I will put this P 0 to this I will move it by 1000 bytes. So, this will when I move it, it will fix off at it starts from 1000 it fix at 90. So, 90 bytes ok.

Now, I will move this 1100 and sorry this 1290 I will go and compact it with after 90. So, 91, I will move this, what this entire P 1? P 0 and P 1 when I move then immediately I will get lot of contiguous space, I will get that 2200 bytes it is I can use it for P 4. So, what it means to move this P 0 c s d out? Just copy this contents there and then what I will do I will make P 0 c s as 0 and 50 50 remains as same, this fellow as 50 because this moves to 1050 moves to 50, and this data sorry data, data is 60 and stack will be 50. I need to go and change these values first thing is I copy everything from here to there and then what I need to do? I have to go and change the value of the segment registers.

After I change the value of the segment register please note that the program will execute correctly, no I need not change your compiled executables still what will happen sorry your 4 will remain 4.

(Refer Slide Time: 22:04)



In the previous case it was that 4 was 1064, now it will become actually 4, now there is a data segment 60 it will become 64. Initially it was 1064 when your data segment was

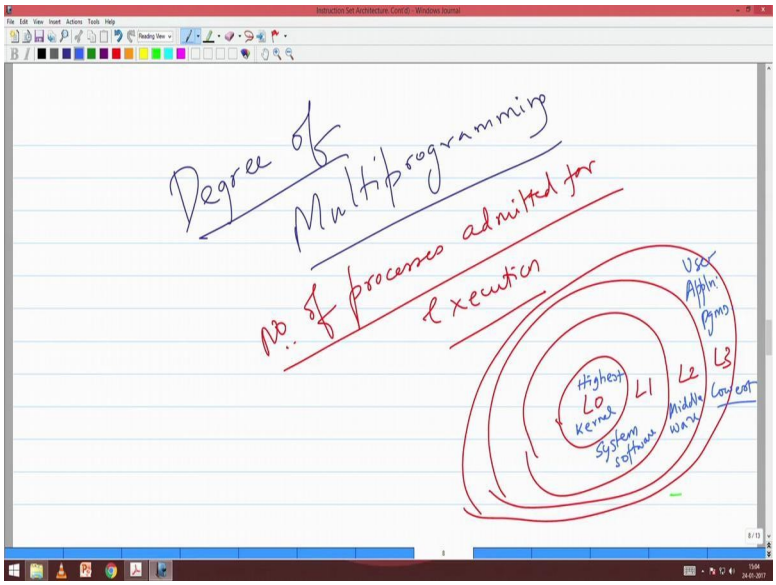
here right I will not go and change the organization of the data, this organization of the

data is not changed not nor this executable is changed. I just copy it and just go and adjust the value of the segment register everything works perfect.

Are you able to follow this? So, this particular concept is actually called sorry process mobility what do you mean by process mobility? I will be in a position to move the process anywhere in the address space; this entire 0 to 1500 is called address space. So, I will be in a position to move the code the data stack anything anywhere, but that entire segment should be in contiguous memory, but segment can be placed anywhere and code need not follow data, data need not follow (Refer Time: 21:32) code can be in [FL] and data can be in [FL] and stack can be in [FL] no problem it can be anywhere in your address space.

You understand I need not even maintain that contiguity between segment, but one segment has to be in some contiguous location, and I can just adjust what I need to do I can put it anywhere and just go and change the value of base of the code data and segment registers that is all nothing more yes 1 minute. So, this is basically process mobility and this is very much important from the operating system point of view; especially this is terminology used in operating systems what we call as degree of multi programming.

(Refer Slide Time: 22:11)



What do you understand by the term degree of multi programming? It is the number of

processes that are simultaneously admitted for execution. Number of processes admitted

for execution. You getting this now there are 4 processes admitted, and as I keep on admitting more and more processes like every every session can be a process when I have millions of processes in a server right.

Then one important thing that operating system needs to do at regular intervals is garbage collection. So, when I want to do this garbage collection then probably I need to do this procedure I need segmentation able to follow right. So, so ya coming to doubts yes.

Student: (Refer Time: 23:27) pointer pointed to some allocation and the data and then we (Refer Time: 23:30) compensate and moved. So, what will happen to the pointer that has to change?

Wait, we will come to that, I will finish this and I will answer this is a programming question I will answer that. So, 4 things fragmentation, intra process protection, inter process protection security all these things are clean, now let us go out now no security is not clear why did you nod your head would not your head right. So, we will talk now more about security hmm, now before going to security I we should understand that operating systems are built in layers right.

So, typical layered operating system looks like this, I thing I have covered this in 1 of the earlier class layer 0, layer 1, layer 2, layer 3, layer 0, basically has the kernel, layer 1 actually has the system software, layer 2 has middle ware like compilers etcetera and layer 3 are the application programs or user programs. Typically operating system can be these layers, the kernel has the highest privilege the application program has the lowest privilege; highest privilege means privilege what privilege means what it can do it cannot go and shoot somebody right privilege means? That it can access what do you mean by privilege what do you mean by privilege.

Student: (Refer Time: 25:42) for execution.

No, no that priority for execution privilege is. So, what do you.

Student: Permissions.

Permissions to what do what.

Student: Access to hardware.

Access to resources right the exact name is access to resources. So, the kernel you just L 0 at level 0 which is the highest privilege, we will have more access to resources and more power in terms of you know I will again say more access to resources, while as I go from L 0 to L 3 the access to resources will start diminish. So, the numerically lowest privilege 0 is the highest privilege or highest power privilege, right. So, Intel based on this (Refer Time: 26:29) x v 6 based on this type if an organization this type of a view, came out with four privilege level. So, you could ask me why 4. So, four privilege levels came because of this type of tyres that we see in the operating system. So, answer is why 4 you will ask this question yes you should ask this question. So, this is a reason.