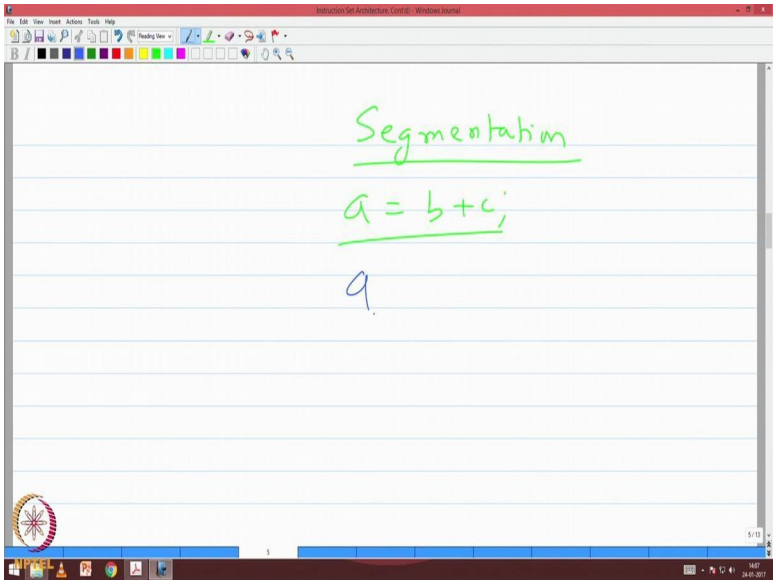
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

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

(Refer Slide Time: 00:18)



(Refer Time: 00:19) which we term it as segmentation. So, when we write an assembly program or suppose I say I want write a equal to b plus c.

Student: Can we use a darker color?

Darker color;

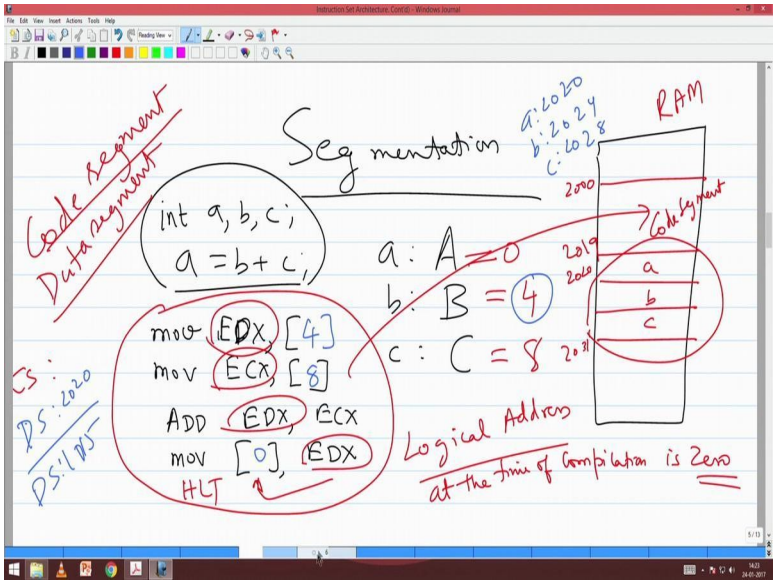
So, that

Student: (Refer Time: 00:43).

Fine I say int a b c a equal to b plus c.

Student: (Refer Time: 00:49).

(Refer Slide Time: 00:48)



Now, please understand that when this program gets compiled, the compiler does not know where a, b and c would be loaded in the memory.

Right, but it is to know because a, b and c are allocated space in the memory and whenever you are trying to access that it should be accessed from the memory wherever it is mapped to and then you read the value of b and c add it and then you write the value into the location A. So, let me say that a is stored at location a, b at say location capital B and c is at location capital C. So, the x axis code for this would be mov say some EDX, comma value of B mov ECX the value of C, add EDX comma ECX mov into location A the value of EDX. So, the value of small b will be loaded in EDX here.

The value of c will be loaded into ECX here, both will be added the answer will be in EDX and that answer will be stored back into location A, now this is what the compiler has to compile, followed, now when the compiler wants to compile will it know where a, b and c are it will not know where a, b and c are correct. So, it needs to know what a, b, c, but it cannot. So, arbitrarily it will say that a is stored at this is equal to 0 what will be this equal to?

Student: 4.

Student: 4.

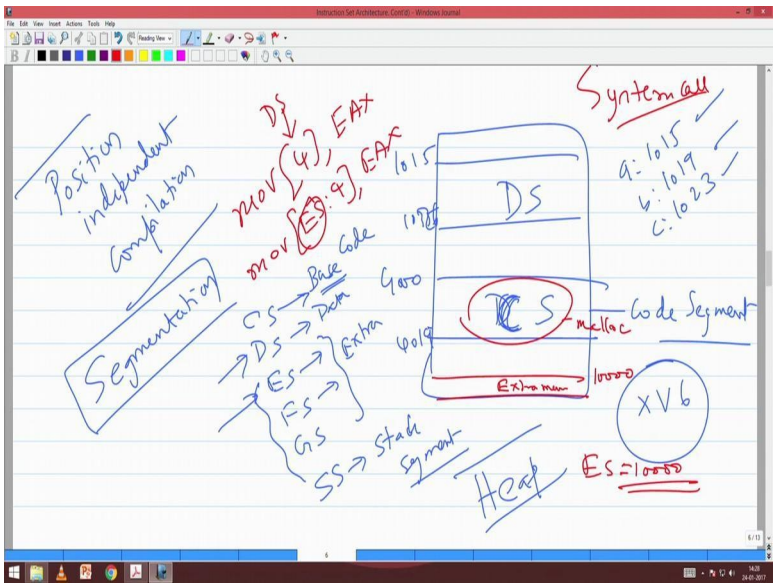
4 because integer can be 4 bites, so 0, 1, 2, 3, so it will make 4 and this will make this as?

Student: 8.

8, so, it assumes that your data segment. So, when I write a program there are 2 segments; one is your code segment which carries which are carries all your instructions there is a data segment which carries all your data now the compiler assumes that your data segment starts at 0 means it starts at logical 0. So, the logical address of your data segment at a time of compilation now this is your ram. So, there will be when this program gets executed there will be a code segment which will contain all these 4 instructions let us assume each instruction as 4 bites. So, let me say there are. So, let me say there is one halt at the end. So, let we say there are 4 into 520 bites. So, it will be loaded from 2000 to 2019, this is where your code will get loaded.

As far as this for first level of lecturing is concerned first level of understanding is now let us assume that data will be loaded between 2020 to 2020, 20, 21, 22, 23 will be a; 24 25, 26, 27 will be b; till 2031, 28, 29, 30, 31 would be c. So, your data is loaded at this. So, your code is loaded at 2000 your data is loaded at 2020.

(Refer Slide Time: 05:29)



Now, there can be another instance where another time when you are executing the program.

Assume we are now executing it on a server let it be some program related to your g mail next time when it is executed. So, somewhere some 4000 to 4019, it will be loaded your code I will call it c s now code segment c s stands for code segment.

And a data will be loaded somewhere between say 1015 and some 1035, well right 1026, this is where your data will (Refer Time: 06:08). So, where is a loaded here a is loaded at 1015, 1016, 17, 1019, 1023; in the previous instance where a was loaded at 2020.

Student: 2020.

B 2024, c 2028, so, but irrespective of wherever I load the code segment and data segment, this program should work correctly and I should have compelled only once I compile only once the same (Refer Time: 06:49) I am executing it once at that point it can loaded into 2000 to this at another point it can loaded into whenever I am executing next time it can load into some other memory you can execute the same executable on different machine like for example, one of your word processors compiled only once by makeover soft right now it is going to get executed and different system at different points of time. So, it can be loaded anywhere. So, this type of a compilation one of the things that compiler wants is what we call as position independent compilation.

Student: (Refer Time: 07:26)

I need to compile in such a way that I am not bothered about I should not care about where the program will be loaded where your data will be loaded that is very very important able to follow. So, this comes this basically now comes as manifest in the form of what we call as segmentation.

Student: (Refer Time: 07:57)

Many of the operating systems even the modern operating systems do not exploit this segmentation that is present in the Intel architection, I for sure believe that today we are talking about security one point of time if at all does could be a secure system then it will be Intel based because they have the foundations for building in security that is why I am teaching you well in advance someday when you want to talk about security certainly Intel will the amount of hardware thought processes that has gone into building both inter and intra process security as I mentioned in the first lab class is phenomenal in

Intel. So, that is why I am taking that as a case study. So, that you could understand that in a better understand and appreciate some of these concepts

Now we introduce the segmentation unfortunately no none of the operating systems. So, next time when you do the operating system course you will be learning something called x v 6. So, last few years we have moved on to what we call x v 6, x v 6 is a abashed version of Linux; Linux in 8000 lines Linux earns to millions of lines closed to millions lines the even the major part of Linux, but we have MIT has taken lot of effort to bring it to some around 8000 actually literally 8050 lines including commands spaces brackets everything this is c code.

Now, in that frame work, so, we will do a complete deep dive of that frame work. So, that you understand how a operating system like Linux would be built I think that is very important. So, we should go to that level that that is why we are spending lot more time and that x v 6 is bootable on an Intel flat form because of all the architectures today Intel is the most open architecture you see 3 4 thousand pages of manuals right already you will see right say Intel has the most well defined documentation etcetera. So, lot of education materials have gone around Intel and m d processes namely the x v 6 process. So, we teach this because for you to understand a first few classes or first even the first few assignments that you will do in operating system which is extremely crucial this knowledge of x v 6 will take you really fun

So, when you want understand the bootstrap the real booting up of the operating system that x v 6 that we are going to see now. So, what is segmentation? So, if you look at the Intel architecture there are 6 registers CS DS ES FS GS where CS actually stores the base address actually from CS I can get the base address of the code segment from DS I will get the base address of the data segment ES FS GS are extra segments and of course, there is one SS which will give you the base address of the stack segment.

Now what we do is whenever I access memory automatically your d s will the address that is stored in d s will automatically get added to your address for example, now I would have put here 0 I would have put here no, no, I would have put here 4 I would have put here 4 I would have put here what 8 and here 0 while compilation when I load the program before loading the program the operating system will allocate some space for you in the memory correct one of the important thing is the operating system has to

allocate some space I say I need some space for executing code keeping my code and keeping my data the operating system will allocate some space what do you mean by that space allocation essentially it will create some segment for you what it means to create a segment I will talk about it and it will give a base address for the segment that base address it will store it in your corresponding segment register. So, your d s here will store what 2020.

So, when d s stores 2020 then what will happen in our accessing memory automatically d s will get added to it. So, now, where is a stored where is b stored 2024. So, whenever I am executing mov EDX comma 4 the moment it sees the bracket it sees 4 it takes the base address of the data segment adds it. So, automatically this will be routed to 2024 similarly this will routed to 2028, this will be routed to 2020, next time when the program gets loaded here your CS your DS will become 1015 when DS becomes 1015 your this will get routed to 1019 this will get routed to 1023 while this will this will get route to 1015 and that what you see here right.

1015; 1019 segment is larger than the largest contiguous memory we will address that. So, there are lot of solutions for this because that is that is entirely this segmentation we will answer that question that is this answer to this question needs 2 months for me when the when the system basically when the program actually gets loaded the executable file will tell the operating system. So, much bite is needed for code segment. So, much bite is needed for data segment correct. So, this fellow will allocate it.

So, the operating system will know while it is going to load a you press a dot out enter immediately your operating system will go and analyze the a dot out the first thing it will found is out what would be the size of the code segment what should be the size of the data segment and it will allocate only one answer I can give you immediately like. So, I have only one. So, the answer for this I thing may be you have already understood this I have only one data segment, why should I have all this extra segment ES FS GS I have only one data segment is there and I used you say everything will be inside the data segment.

Student: You will have large files.

Large files; no I am executing a program file will be in the disc. So, why as a memory I am not bothered about file size?

Student: (Refer Time: 15:14)

So, when I do a dynamic allocation malloc that I will not know right as a compiler will I know no. So, what will I do when there is a malloc what will I do you just allocate a space for the pointer correct followed that point when we point at to the memory that starts and then remaining entire memory can we taken out by another data section called a heap. So, what happens is in your code when the code is actually getting executed at this point when there is a malloc the malloc is actually a system call is called a system call system call means malloc I am a program executing malloc I cannot go and allocate memory for myself I go to the operating system god and say hey god bless me give me some memory right. So, what is argument to malloc.

Student: Size, size (Refer Time: 16:13).

Size; so, much memory I need, correct, so, the operating system will bless you, [FL] take it from where it bless you somewhere here. So, this is your extra memory correct. So, the; so, when you do the operating system course you will lot more learn about this there is all the free memory will be kept in a heap right in a heap why heap because I want some memory right I need to find out the memory closest to it correct closest in size to it I could have best fit worst fit different algorithms are there every time I ask I want thousand bit of memory right, I will go and find out that closest contiguous memory which is very closest to thousands, just greater than 1000 to do this type of a searching yeah max heap or a min heap will help.

So, the heap will basically store the values of how much free memory is there and where and. So, you can search the heap to find out something which is very closest to you. So, if you start (Refer Time: 17:24) for 1000; the element you will land up will be very close to thousand in the heap. So, that is why heap type of structures maintained basically where free list will be there and you can take there are also algorithms like best fit take the largest chunk of memory and keep filling it there is another algorithm called worst fit take the smallest chunk and fill it if I want implement the best fit and the worst fit then I can use a max heap and a min heap.

So, the moment I do a malloc one minute I will basically get some extra memory from the heap and that memory say it is at ten thousand now I will make ES is equal to ten thousand and every time I will access this data I can say if I just say movs 4 comma EAX

automatically DS will be added, but I can also say mov ES colon 4 comma ES then I will go and access the fourth bit in your segment pointed to by e s correct you are getting this segment pointed to by ES, I can go and add it. So, that is why extra segments are used in case I want more memory then what I see during a compilation if I have a malloc statement inside a compile program, the compiler in no way can find out how much memory in a because it is dynamic memory allocation we need. So, only at the time of execution you will find out how much memory extra I need and then I can use these segments for doing this. So, that is why extra segments are currently being used, fine.

Now, I will come to doubt yeah.

Student: Sir there is a heap affects the performance while using the memory (Refer Time: 19:14).

There is no other heap actually improves the performance somewhere I should go and find out where the memory is heap can do it in logarithmic time the other solution other than heap would be a link list a linear list.

Student: no, I am saying compared to DS storing (Refer Time: 19:33) does it affect the performance.

Why should it? No, I did not get a question, can you repeat it?

Student: (Refer Time: 19:45).

Can you repeat the question?

Student: Suppose I am writing a code;

Student: (Refer Time: 19:51) malloc say 10,000 (Refer Time: 19:54) space.

If I access through ES and I access through DS, is there a difference.

Student: Yeah.

No, there will be no difference both are segment registered its in the proximity of there.

So, accessing any memory there will be no difference.

Student: so like we do not have been there might be multiple malloc (Refer Time: 20:10) it is like the number of malloc (Refer Time: 20:13) multiple of a malloc statement. So, there could be multiple data.

Yeah, yeah, yeah, there could be;

Student: How can like why only 3 registers there might be?

Yeah that is a very interesting question. So, why only 3? So, let us go out to I will be teaching in the morning course in the in the theory part there will be. So, many variables right [FL] variables, correct, now I have only 4 general purpose registers I will be very happy if I store the variables in the general purpose register, correct, why I should be happy because I have very fast access.

Student: Like, like.

Wait, wait, why 4 registers.

Student: Like I am the like all the variables are we already know at compile time. So,

(Refer Time: 21:02), but now we do not know how many;

So, your malloc; number of mallocs in any normal program we can kill the program we can kill any architecture by writing a program that is anti architecture for every architecture we can for every policy that I am going to tell here you will always have a program that will kill it that is easy process, but in an average when you write a program for sake of writing a program not for killing the architecture then those program if I say just 3 more 4 registers are sufficient to handle variables cant 3 segments enough for handling mallocs.

Student: If I say; if I have a malloc statement in a wild loop by say I am;

You will not have a malloc statement, you will actually free it.

Student: Like, like, what I am trying to say is there might be more than say hundred non contiguous data segment to the program right. So, how will we how will we know where the data segment.

Yeah. So, the see the first thing is you will not have such things suppose you have then you have to you have you have you have something called a descriptor table we are going to come to those things where you will maintain the information, but you will vary the registers of this if I start answering that question everybody we will go for a toss, but your question essentially will get angle, but the reason for why 3 the answer is 3 empirically analyzing all the workbenches workbench means typical programs that will run on your system 3 is more than enough correct. So, that is sort of at any point of time that is why people fixed 3.

What happens if it would more than 3 right you have 4 general purpose registers the remaining thing you are AESP and all other will be used for stack etcetera EVP for base pointer etcetera that we 4 general purpose is. So, sure available to you ECX will be used for some counter if you have a wild look that is all ECX will go there. So, so there be some 4 ES I ED; ED I and the EAX; EBX and EAX will be used lot of time if you use a due 2 of those 4 will also go off. So, still we are very happy Intel survived for 3 decade using 8 general purpose registers why again that is an empirical analysis we will come to that.

So, all these why are in very important question why 3 why not 4 why not 2 all these are very very important questions for several things in architecture this why has been evaluated on some benchmark and. So, there will be a paper on why 8. So, lot of theorem why is the answer is as why you did not attend morning 8 o clock class, right, 2 days you will say feels I slept sir then the third time you will say you have to find some random reason. So, what are occur to your mind that day you will tell that as a reason I cannot verify that you know not am I interested I want to ask a question just to believe it.

So, similarly here that day when that designer tried to design this hardware he was very happy doing 8 he was very happy doing 16 normally it is power of 2 because that you need to address them then if it is n 2 power k 9 I need k bits if I put some prime number then lot of bits get wasted there. So, normally it will be power of 2 that is one decision many times it is the best I could do at that of time or my mood I had that mood that I had decided 8 you as random as this. So, lot of answer in organization or like that, but we will see many things right, because when you look at later things like there is there is a very nice (Refer Time: 24:58) some decisions some statements made say 2 centuries before sorry 2 decades before how are they relevant today.

Like there is a very nice article in spectrum I will try and download it probably the last class of computer organization I will give you that [FL] and go. So, that will be very interesting. So, fair enough. So, I have answered as I elaborate answer for your question so.

Student: Sir;

Yes?

Student: Can we (Refer Time: 25:29) segment (Refer Time: 25:30) manipulated (Refer Time: 25:30)?

Sorry, yeah now comes other question segment register who will manipulate how it will be getting manipulated we will come all those question I am going to answer [FL] you understood what is segmentation?

So, segmentation see segmentation the first interest of segmentation was given by the compiler which said that at the time of compiling I do not know where the data has going to be loaded I want to assume it is 0 and the hardware please support me. So, that operating system and hardware please support me. So, that I can do, so, what is the role of operating system operating system will allocate a segment and it will initialize the segment register what is the role of hardware whenever there is a memory access the hardware will take that value from the address like 4 here value from value of the address from the instruction add it to the segment base and fix that. So, the please understand the problem of the compiler is being solved collectively by the operating system and the hardware.

All getting this, so, this is one very interesting thing about segmentation.