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

Lecture – 01

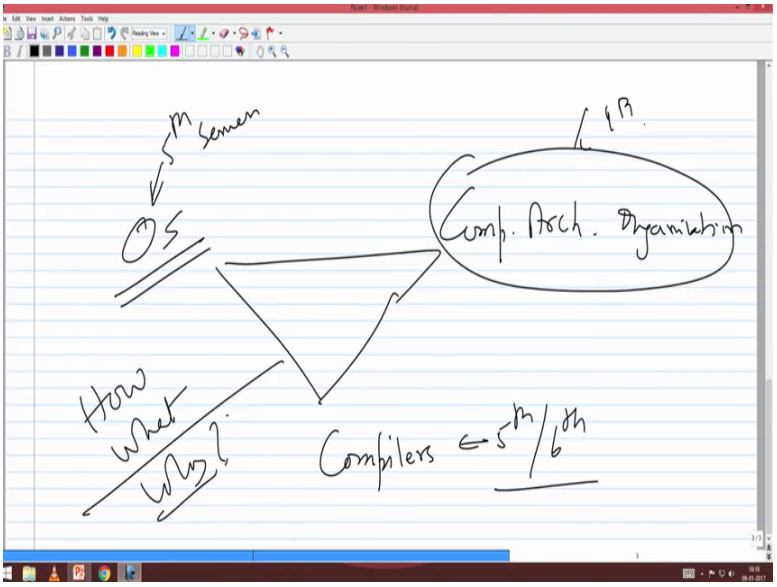
Introduction

High-Speed Circuit Design Recursive Doubling

I am trying to make this course as independent as possible to that course, but there will be some points where you may have to do a little more work to get in phase with what we are trying to teach here. Because your digital course essentially dealt with only logic minimization, but logic minimization and some counter design etcetera hardware design, but that was actually one-fifth of the course of our course, so they have learnt four-fifth more than what you have learnt in that course.

So, we will try and bridge the gap, but I should tell you that you have to do a little more homework to see that these gaps are bridged. So, that is one thing. So, with this as the background, let me start this course again this course there are 3 pillars to computer science.

(Refer Slide Time: 01:10)



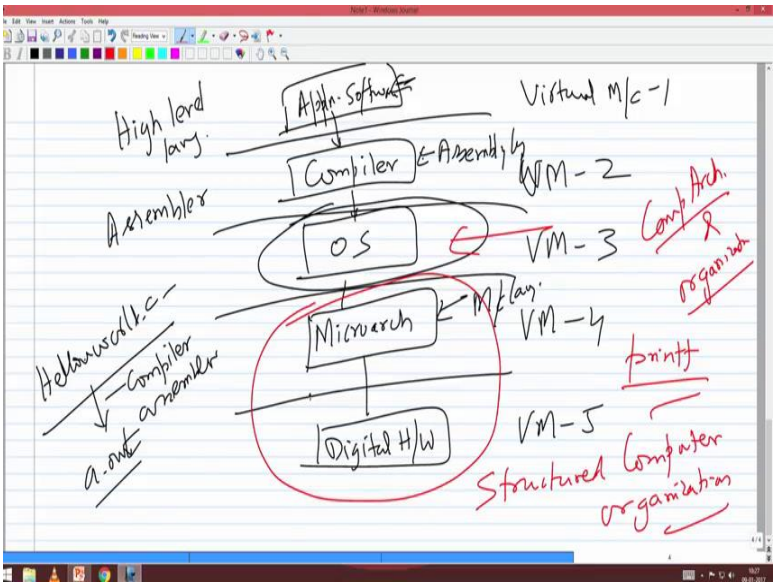
The one is the computer architecture and organisation. Another course is the operating systems, which we will be doing in the 5th semester, and then compilers, again you will

be doing in the 5th or 6th semester depending upon the curriculum. And these 3 pillars actually hold key for you to be a good computer science and engineering student.

Computer science is all machine learning all this stuff. Computer engineering is another you know suffix added to your degree called engineering. And to be a good computer engineer you need to understand these 3 subjects in great detail. So, this is the 4th semester course, we will be doing this now and then of course, the other course. And please note that all these 3 are highly interrelated. So, lot of concepts in organisation will be used in OS and also in compilers. So be very attentive in the class and try to understand. And the way I have structured the course is that every time I talk about some concept I will also talk about the operating system issue of that and also the compiler issue of that. And there are lot of books available in the market. So, lot of books will teach you how and what how something is done and what is done, but they will fail to teach you why it is done.

There are several reasons for why the books are missing this why part, but we will not go into that, but this course I will be basically teaching you why certain things are done, what is the rationale behind those things, and you have to understand that, so that you know we make best out of it and understanding that why is very crucial fine. So, when we look at you know the organisation of a system, there are 5 levels in the organisation of the system.

(Refer Slide Time: 03:08)



One is the basic digital hardware, on top of it is the what we call as a micro architecture, and top of it is the operating system, then the compilers then your application software.

So, there are at least 5 levels that you need to study to understand this system correct. Now please note that each level is isolator from the remaining path. So, each level I could call it as a virtual machine. So, I can say virtual machine 1, virtual machine 2 virtual machine 3, virtual machine 4, virtual machine and each of this has a language for example, the application software is basically written in a high level language like C, C++, java. Then there is an assembler which actually converts this high level language into a language that is understandable by the hardware right. So when I look at say hello world dot c, this is a high level language program, this gets converted to some a dot out. The software involved in this conversion are the compiler and the assembler.

So, from the application software here, there is a compiler which actually converts it into an assembly language and there is an assembler which will convert it into 0s and 1s. So, this is the machine language right. So, there are 2 translations that happen. The first translation is the application software getting compiled into an assembly language program and then the assembly language program getting converted into a machine language. So, this virtual machine that you see on the top most layer can understand one language which is the high level language the compiler basically takes the high level language and converts it into an assembly language.

The micro architecture can understand only the assembly language or it can understand only the machine language. So, there is an interpreter or an assembler which will convert that compiled assembly language program into a 0s and one, which is the machine language. So, every machine here the virtual machine here essentially has an input language and an output language or what you call as an input source language and a target language right. And so there is some conversion process. So, when we go from top to bottom the top most layer is the one that is understandable to the human being much understand easily understandable. And as we go down the stack it becomes more understandable to the hardware or the machine right. So, there is a gradual decrement in the obstruction at the highest level I have a very high level of obstruction, and as I go down the level of obstruction comes down, and finally, you land so land up with ones and 0s is your micro architecture (Refer Time: 07:11).

Intermediately there is one layer called operating systems, one layer called operating system. So, there are certain functionalities which a program cannot directly execute it needs a help of an operating system. For example, in the hello world you have printf you never write the code for printf. So, you ask you ask the help of someone and that someone is the operating system. So, the operating system basically tries to satisfy some of the needs of your application software right. For example, in the assignments that you did you use lot of classes and libraries correct. Your math library, your memory allocation libraries, so many things that you use in your assignment here, correct. Those we claimed as those are some of the facilities prevailed at you by the operating system right.

So, the functionalities that are basically done by the operating system and those are the things which are required by the application software right. So, in this entire organisation we call this as a structured computer organisation. So, in this structured computer organisation every machine I call it as a virtual machine. Virtual machine in the sense that the person working at that level for example, if I am writing an application software I need not really bother what my who is going to compile my quote what is the underlining architecture. So, the entire thing below the application software the 4 layers below the application software is completely hidden to me, for me I see a machine which will take a c program and execute how it executes is not my problem right. So, that is why we have this adjective virtual there right. So, virtual machine, because we are not bothered about the layers below that and so if we take any layer any say for example, operating systems. So, it has an isolation from the other layers. So, every layer is completely isolated from the other layer and so and that is the reason why we call each of these layers as a virtual machine.

So, now this course essentially talks we can call it as computer architecture and organisation. Some years before these 2 were taught as separate courses, but there is no point in teaching them as separate courses because if you do not understand the underlining architecture then you cannot understand the organisation and vice versa. So, we and unfortunately our B.Tech curriculum is so packed that I cannot have 2 code courses one for organisation and another for architecture correct. So, what we are trying to do is that we will try and try to teach you both in this course right. And the way we

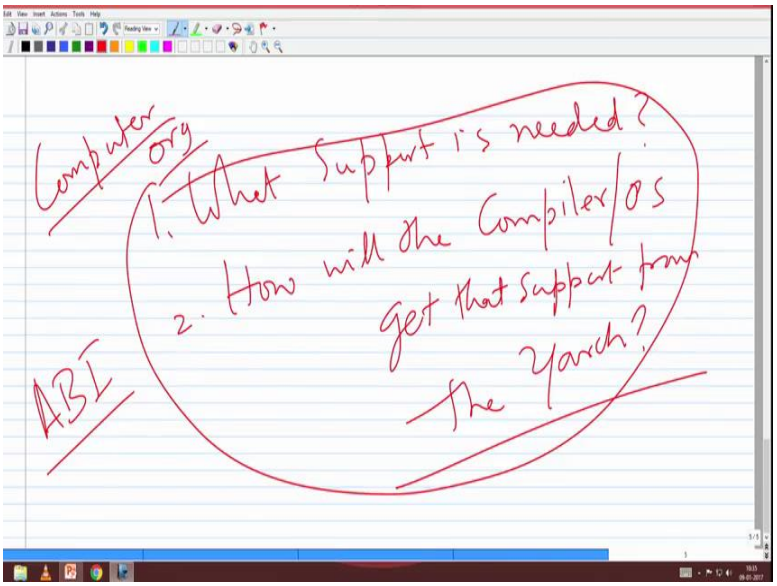
have structured. So, first and foremost one fundamental question is what is difference between computer architecture as a subject and computer organisation as a subject right.

Now, why did I create this, why did I create micro architecture, why did I create a digital hardware because somebody wanted it, correct. If nobody needs it then there is no point in creating it correct. And so who is needing this who is driving this, from say a calculator today we are having a mobile phone with 8 processes who drove it right. The customers now how did the customers. So, the customers need is reflected to the hardware through what through the software, through the compiler, through the operating system right.

The compilers grew in strength the compilers became more complex they started supporting variety of features right. They started supporting variety of computational processes just because the user wanted it correct. The operating systems started supporting say they now supporting large amount of memory, it is supporting large different functionalities different types of libraries, why that what is the need for the operating system to do it because the user wanted it. There is some programming environment there is a software requirement, there is an information technology requirement which is forcing the user to basically to ask certain functionalities from the underlining programming language and the operating system. And if we start asking those things then ultimately your compiler grows and your operating, so operating system also grows in terms of the required functionality. And those functionalities it has to realise it they are all software right. Operating system is a software computer compiler is a software and how will they realise that functionality by executing on the hardware. So; that means, they will require some support from the hardware right. You understand this right.

So, what are the supports that are needed and how will the operating system and compiler you know communicate with the hardware to get that support. So, 2 questions we need to answer what are the supports that are needed.

(Refer Slide Time: 12:44)



What support is needed how will the compiler slash OS get that support from the micro architecture. These 2 questions need to be answered. Answer to these 2 questions form the subject matter of what we call as computer organisation right.

We want answer specifically what sort of support the modern operating system is asking the hardware right. I can give you lot of examples here right. For example, if I am looking at a real time system I did define what a real time system is there in the previous right. Do you remember, forgot. So, what is a real time system? Can you give an example of a real time system versus a normal system? When you drive a car there is some electronics that is supporting or breaking system is it a real time system, yes right. A normal system is one where the correctness of execution is determined by if you get a correct result

A real time system is one where the correctness of your execution is determined by not only the correct result, but also the time at which that result comes right. If you break if you apply break the car should stop that is a correct result, but then it should stop within the next few seconds, then only it is correct. Now you apply the break and that it does not stop within the next few seconds dashes then it is not a correct result right. You understand so a real time system is one in which the correctness of execution is not just the correctness of the result alone, but the time at which that result is manifested right.

So, when I start look. So, suppose I want to design a real time system there is an operating system running there right. The operating system responsibility of that software is to see that I give you a service the moment you ask very quickly within the deadline you understand this. Now when I want to design a real time system how will, So, suppose I am putting a computer in a car to basically look at breaking right. How will the processor in that computer understand that there is a break that is necessary there is a that it has to act on a breaking routine, how will it realise that there is a break that is applied can you give me a mechanism by which it can.

Student: Sir, electric.

No that is how the break comes up then how will the processor realise it.

Student: Electric signal electric signal.

Electric signal will come and then it will come in the form of. So, you type you just press the keyboard how will the system realise that the there is an input coming from the keyboard.

Student: Interrupt.

Then there is some basically there is an interrupt right. There is an interrupt that is coming up now in the case of the real time system that interrupt should reach from the time the interrupt comes to the processor, the time that the processor realises that the interrupt has come right. That time what we call as the first response time should be very fast. So, when I am designing an architecture an hardware for implementing real time system one of the thing that I need to ensure is there should be a fast interrupt path. You are understanding this, so because once the interrupt comes I cannot take long time to realise that there is an interrupt. So, when the interrupt is generated using your piezoelectric or whatever when the break is applied the interrupt is generated from the time it is generated till the time the processes starts responding to that right. It should be very fast right.

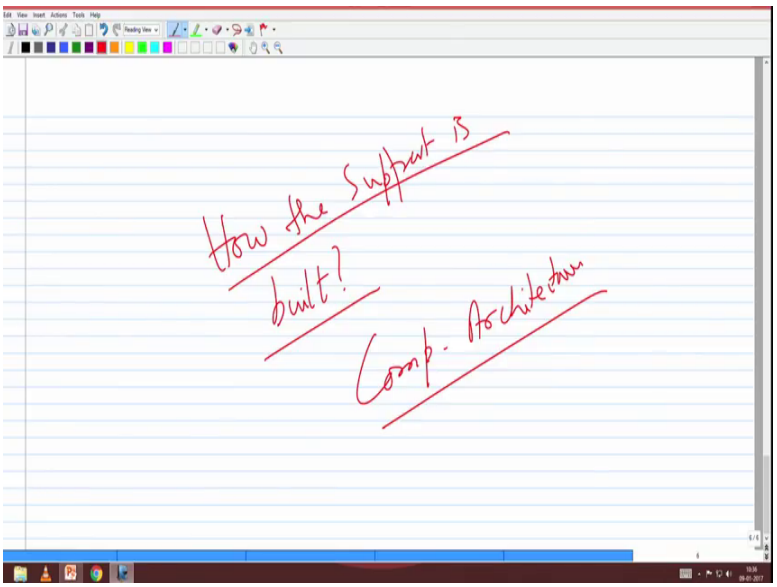
So, that is a requirement from the user or the operating system if the operating system says I am responsible for the safe running of this car then it says the hardware re please give me a very fast interrupt response then I should design my architecture in such a way

that the interrupt response is fast correct. Are you able to follow what I am trying to say right? So what sort of support is needed and how will the compiler OS get that support from the micro architecture these are subject matters of computer organisation. And I have just told you one such need or support that is needed by the operating system in one context of an automotive example. Now this is also called like you have API right. What is the expansion of API?

Student: Application program.

Application program interface this interface is called a b I application binary interface right. Now understanding this API is the subject matter of computer organisation.

(Refer Slide Time: 18:15)



Now, that I have understood what are all the support needed right. The type of support needed. How will I build that support in the underlining hardware? How the support is builded how the support is built? So, this answer to this question is a subject matter of computer architecture.

In my opinion computer organisation cannot be taught as a theory subject. So, computer organisation if you say I am going to teach it is a stupid thing in my opinion it is my strictly my opinion there are n universities and n profs teaching this. So I am going to align computer organisation with the lab. So, the computer organisation will be taught during the lab hours along with the experiments. So, I will teach something immediately

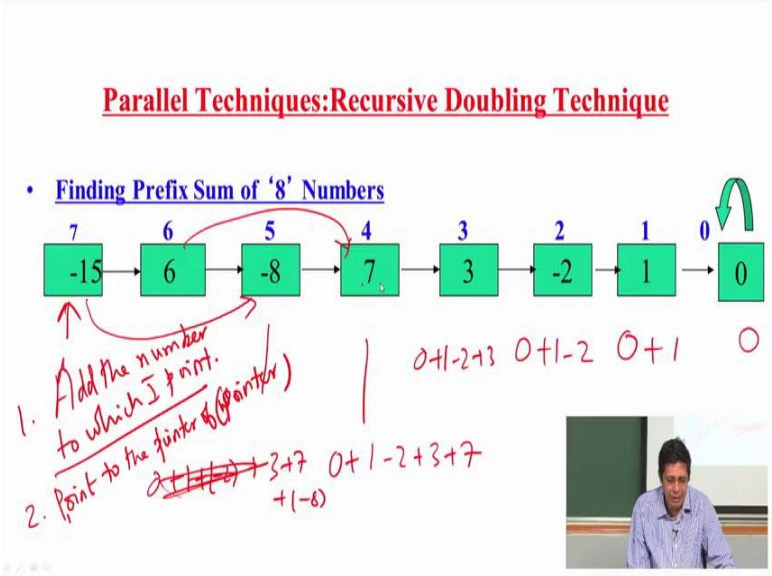
that that evening if you spend that 2 hours you will finish off then do not bother till next Tuesday.

So, every day we will give you some exercises and ask you to immediately go and implement it and see it for yourself. And we are giving you a excellent environment using the most complicated processor every done in the globe call the AMD or the Intel platform we will teach you lot of things there of what could be a possible good application binary interface and we will make you work on that we have we have spent lot of time in creating that environment will now available I hope all of you bring your laptop and I actually shared the links to all of you I hope at least finish the installation and talk about it tomorrow also. In the theory class I will cover architecture and then relate it to what is happening in the organisation. So, we have to answer both these questions in this course. So, that we get a good understanding of what is computer architecture and organisation fair enough.

Now we will now start with hardware basic digital hardware right. Now we will start with the basic unit of operation called addition. Addition and multiplication I will teach you one very good adder and a very good multiplier once I understand addition automatically you would have understood subtraction, but anyway I will cover some basics tomorrow then we will look at some division algorithms and then we will also look at the IEEE 7 54 floating point representation is also extremely important. So, we will look at these floating point operation, how floating point division is basically carried out or floating point multiplication is carried out right.

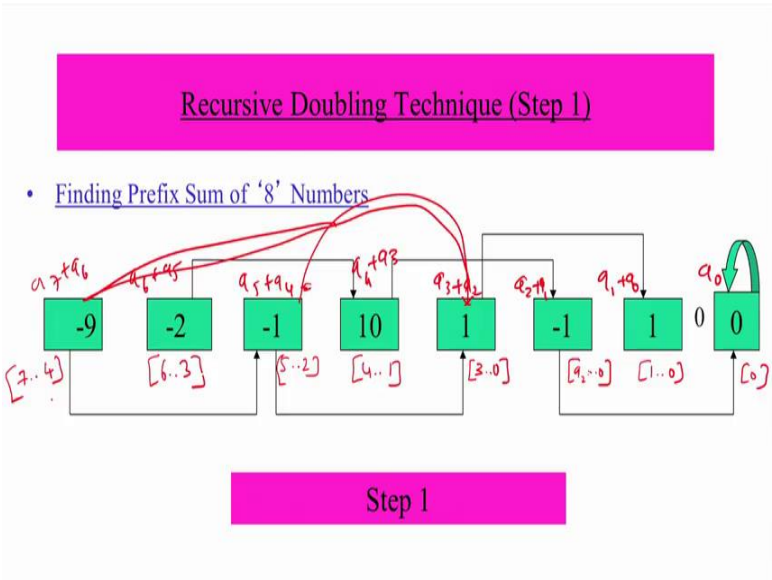
So, we will look at this thing for 2 weeks we will concentrate on high performance circuit design right. So these are some things which are not covered even it was not covered for the palghats students also. So, I have not covered in the foundation system because it is supposed to be covered now. So, we will do this in the next 2 weeks and I will try to make it as interesting as possible, but it will be interesting. So, we have some very interesting ways of implementation.

(Refer Slide Time: 21:55)



Now, let us start with this very interesting techniques called recursive doubling technique. I want to find the prefix sum of 8 numbers. What I mean by prefix sum is so I want 0, 0 plus 1 0 plus 1 minus 2 0 plus 1 minus 2 plus 3. So, this is 0 plus 1 minus 2 plus 3 plus 7, 0 plus 1 plus minus 2 plus 3 plus 7 plus minus 8 and so on. This is called prefix sum. So, at the end this will have 0 this will have one this will have minus 1 this will have 2 this will have 9 this will have one this will have 7 and this will have whatever 7 minus 15 minus 8 or vice versa.

(Refer Slide Time: 23:01)



I think we could also yeah vice versa see; I want minus 15 I want minus 9 I want no this way. So if you do this so if I keep summing this I have a 0 a 0 plus a 1 a 0 plus a 1 plus a 2 a 0 plus a 1 plus a 2 plus a 3 a 0 plus a 1 plus a 2 and so on.

Suppose I want to do this. And I do it one a normal system right. How much time will it take assuming addition takes one unit of time, how much time will this take to add these to get these results. Assume this is an array how much time will it take.

Student: 7.

7 units is of time right. Now can I do it faster let us assume that I have a processor for each one. So, I have 7 processors available to me. Can I do it much faster this is a question you are understanding. So, if I have one processor it is going to take 7 units of time. I have 7 processors can I do it faster you. So, what I do is this is the algorithm very simple algorithm what I do is at every step. So, at every step I add the number to which I point and make this is the first step, and second I will point to the pointer of pointer. For example, I will add this element will add 6 to minus 15 so; that means, you get minus 9 and it will now, point to 5 pointer of pointer. So, 15 minus 7 is pointing to 6 6 is pointing to 5 at the end of the first step it will now point to what.

Student: 5.

It will point to 5 and similarly what will 6 do 6 will add to minus 8. So, you will minus 2. And it will now point to 4 you are understanding this. So I will add to whomever I am pointing the content of whomever I am pointing and then I will point to that fellows pointer. So, are you able to follow this previous step? So, on the first step I will add minus 15, 7 points to 6. So, the content of 7 will be added to content of 6 and stored at 7 and now 7 s pointer will point now to 6 pointers.

Similarly, 6 is pointing to 5. So, the content of 6 will be added to content of 5 in parallel because I have one processor for each right. So, in parallel I will add 6 to minus 8. And now 6th pointer will be pointing to the fifth pointer. So, 6 to 4 and so on you are able to follow yes or no now you see. So, this is the step one after the step one please understand that 7 has minus 9, 6 has minus 2 this has minus 1, 4 has 10, 3 has 1 and you know 2 has minus 1 and all and they are pointing.

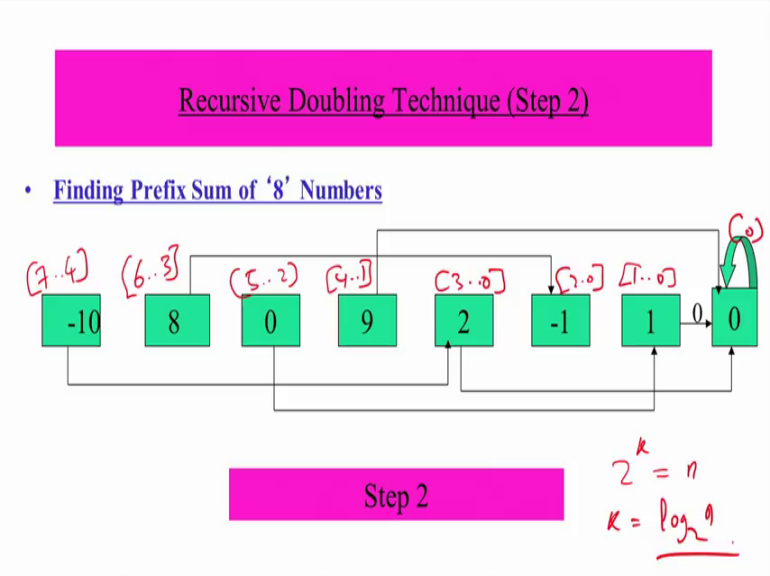
So, what will be here this will be a 7 plus a 6 a 6 plus a 5 a 5 plus a 4 a 4 plus a 3 a 3 plus a 2 a 2 plus a 1 a 1 plus a 0. So, at the end of my first step I have added a number which is one away from me it every element has added a number one away from it, and it is no pointing to a distance of one away from it right. So, now, please note that whatever is the content of this location is a 7 plus a 6 and it is currently pointing to a location which is storing a 5 plus a 4.

Similarly, whatever is stored here is a 6 plus a 5, and it is currently pointing to a location which is storing a 4 plus a 8. So, in the next step what will happen minus 9 will get added to minus 1 right. That means, what will be stored here it will be 7 to 4. Because here it currently it has a 7 plus a 6, and I am pointing to a 5 plus a 4. So, I will have 7 to 4 what this is having 6 to 5 and this is 4 to 3. So, this will have 6 to 3. And this has 5 to 4 and 2 to 3. So, this will have 5 to 2 sorry 3 to 2. And this will have 4 to 1 a 4 plus a 3 this stores a 4 plus a 3 and this is pointing to a 2 and this will have 3 to 0 and this will have 2 to 0 and this will have always one to 0 and this will have 0 able to follow right.

At the end of the next step when I click this that is step 2 these elements will now store from step because currently it is storing 7 to 6 and it is pointing to 5 to 4, now this will become 7 to 4 correct. Now where will this point to this will point to pointer of this so this will now point to this. At the end of when I go to at the end of step 2 what will what will happen this now will store the values from say a 4 to a 7 sum of a 4 to a 7 and it will now point to what sorry, this will now point to thing storing 3 to 0 correct. You are able to follow this.

So, let us see what is happening at this end.

(Refer Slide Time: 29:53)



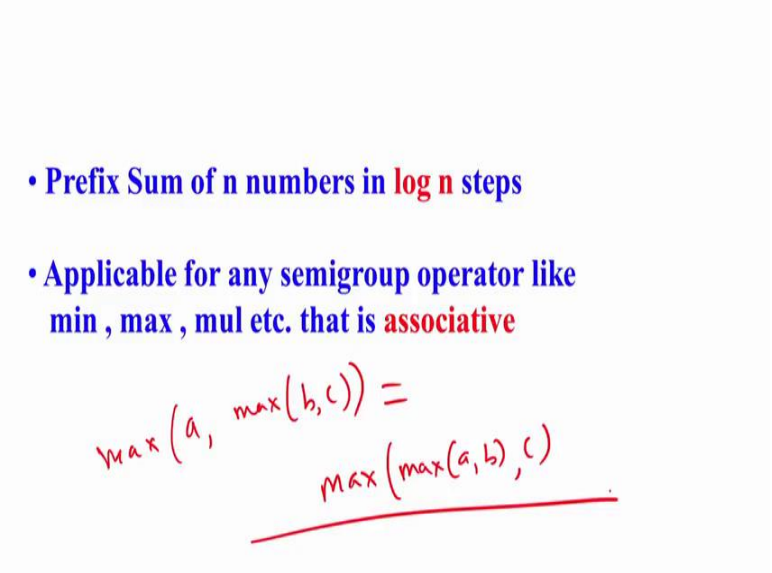
This will store 7 to 4, 6 to 3, 5 to 2, 4 to 1, 3 to 0, 2 to 0, 1 to 0 and 0 this is at the end of step and the end of step 3, what will happen this 7 to 4 plus 3 to 0 will become 7 to 0 6 to 3 plus 2 to 0 will become 6 to 0, 5 to 2 plus 1 to 0 plus 5 to 2 is pointing to one to 0 it does not become 5 to 0 4 to 1 plus 0 will become 4 to 0 3 to 0 2 to 0 1 to 0.

So, next step what happens. So, you will have the answer from all and this is the answer this is the step able to follow this so if I have n processes. So, at every step I add the number to which I point I add the content of the location to which I point and I make my pointer as pointer of pointer. If I keep doing it, please note that at the end of a step first step I am pointing to an element which is 2 power 0 away from me and I have added 2 elements. In the end of next step, I am pointing to an. So at the first step I am pointing 2 power 0 at the second step at the second step I am pointing sorry. So, this is 2 power 0 at the end of first step I am pointing 2 elements away from me. So, this is the 2 elements if you subtract the index it is 2 at the end of the second step I am pointing to 4 elements away from me right. So, at the end of k steps I will be pointing 2 power k. So, I stop when 2 power k is equal to n so; that means, k is log n.

So, if I have n and note that all the operations I do in parallel correct. So, every step is done in one unit of time. So, there are 7 processors and each processor in concurrently in parallel, just goes and adds to what I am pointing to and then makes my pointer to pointer of pointer. Are you able to follow? So, in log n steps I will be in a position to

finish off this prefix sum. So, if I have one process I am going to take n steps, but if I have n processors I can do it in log n steps.

(Refer Slide Time: 33:01)



Why is this possible because we are doing addition what is addition is a associative operation since I am doing addition I can do it because I assume that the operation that I am doing is an addition is an associative operation if it is not associative then a plus b plus c is not equal to a plus b plus c meaning if plus is a non-associative operator if plus is addition then. So, I can do this for any associative operator. Please note that this algorithm will not work if I do not have an if the operation that I am defining is not associative correct. You got this?

So, I can do this operation I can do prefix min I can do prefix maximum because maximum of a comma b, max of max of a comma b comma c is max of a comma max of a b comma c right. Max of a comma max of b comma c is equal to max of a comma b comma c correct. So, max is an associative operator similarly min. So, I can do similarly multiply. So, I can do prefix I had addition prefix sum prefix maxima prefix minima prefix multiplication.

So, we will just stop here. So, this particular concept of what we call as recursive doubling will be used to construct an adder and we will see how we are going to do that in tomorrow class.