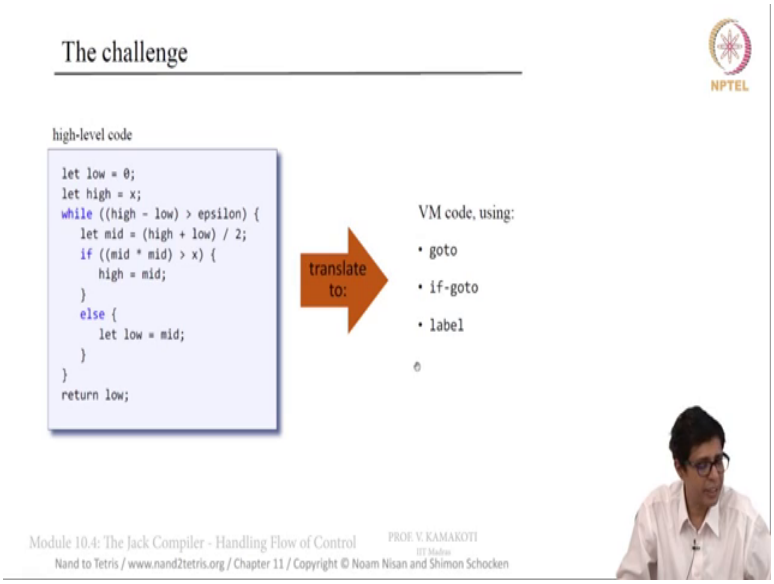
Foundations To Computer Systems Design
Professor V Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Module 10.4
The Jack Compiler-Handling Flow of Control

Welcome to module 10.4, in this particular module we will be talking about Handling Flow of Control. So what we have done so far we have looked at handling expressions we have looked at handling variables now we will now look at handling Flow of Control. Flow of Control essentially says how we go from one statement to another statement, one subroutine to another subroutine so when we are compiling the entire jack one class how do we go from different aspects of that class to the next part.

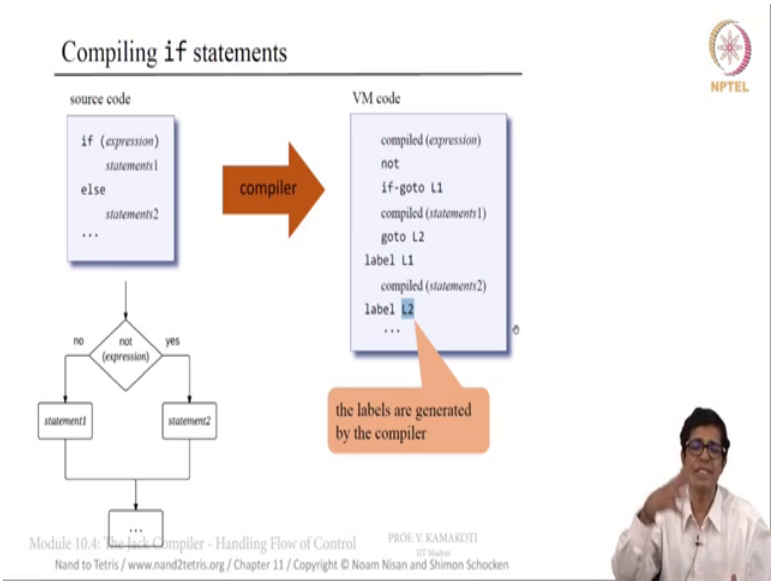
(Refer Slide Time: 00:51)



So that is what we are going to do and as I told you I am using the same slides that are part of the chapter 11 in the website. Now what is Flow of Control? So there are some statements like let statement which are straight line statements does not alter the flow of control right so after let the next statement let low equal to 0 the next statement let high equal to x is going to work but the moment you look at while when this condition is satisfied then it goes to next let then if etc but when the condition is not satisfied it skips and it goes to the this return low right so while is a statement which will alter the flow of control as of the program.

Similarly if this condition is satisfied then high equal to mid will be executed if this condition is not satisfied it will skip this and go to this else low mid so this is how the while statement while and if statements basically work. So they are ones which will basically alter the flow of control and that is what we are basically going to address and so we will be using in the VM to realize the same syntax in the VM we have two statements namely go to and if go to and then there is a label also, label statement which will tell you where to go to right where to if go to, so this two syntax you would be very clear, go to will just go to some label if go to will depending upon what is there on the top of the stack it will take a decision of whether to go to or not to go to right. So let us see how we translate these statements into the corresponding things.

(Refer Slide Time: 02:46)



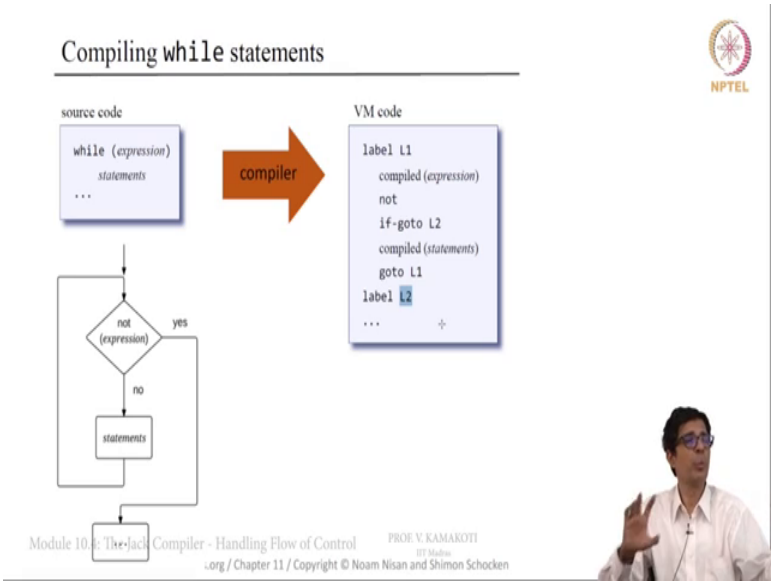
So how will an if look like? So if you go into the grammar if expression there will be set of statements else there will be set of another statements so this is statements 1 let us call it statement 1and statement 2. So what is the equivalent VM code here? Will just compile this expression once this expression is done what will happen on the top of the stack the answer for this expression will be there and that will be either true or false. Now we compile this expression now we will do not right and not not means that not of this expression.

So if the expression is true not of this expression will make it false right, if go to L1 right so if the expression is false then it will become true so if this expression is false not will become true so if go to L1 means what will if go to 2 if the top of the stack is true then it will go to L1 that

means if the top of the stack is true this will got to L1, when will the top of the stack is true? When the expression is false right so when this expression is false the top of the stack will be true and it will go to L1 that means it is going to skip statements1 and that is exactly in this case also if the expression is true expression is false then I am going to skip statements 1.

So if the expression is false not of that will be true and so if go to L1 I will actually go to L1 if the expression is false yes when and by going to L1 I am skipping statements 1 I jumped to this part here as you see here from here ok right so I am skipping statements 1 and similarly when a expression is false I am skipping statements 1. If expression is true then not of this will become false so if go to will not take place so statements 1 will execute and then without checking any condition I got to L2 so that means I will skip statements 2, so if the expression is true not if that expression is false if go to will not take place so statements 1 will be executed I see go to L2 so will jump to label L2 here ok. So this is how the syntax of if statement gets translated into a VM code.

(Refer Slide Time: 05:26)



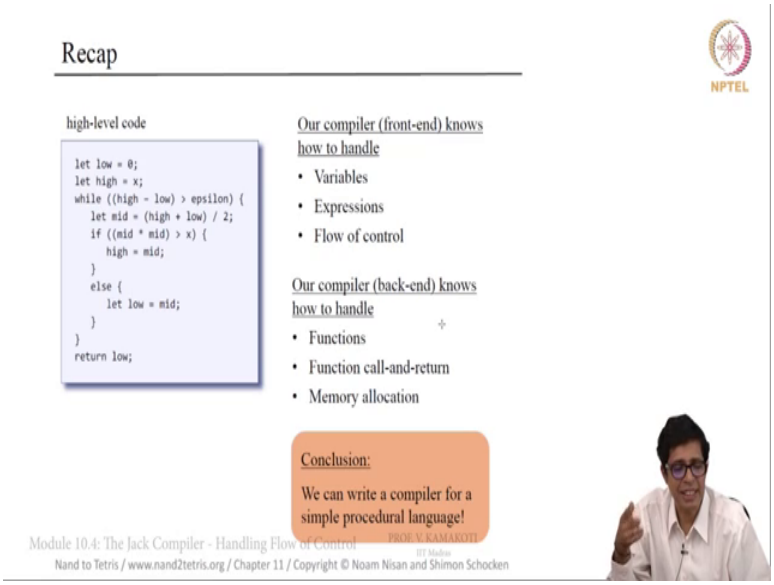
Similarly what is this while? While again when the expression is true it execute the statement otherwise skip it so I will just put label L1 we will compile this expression not so when I download what do you mean by compile the expression? The code for that expression will be here and what will that code do? The answer of that expression will be on the top of the stack. So

when I compute an expression already we see at the end of executing an code for expression the answer for that evaluation of the expression will always be on the top of the stack.

Now I go and say not of that so this is true not of that will become false and if go to L2 so it will not go to L2 so I will be executing statements and again coming back to L1 so if this expression is true not of that will be false if go to will not work so this compile the statement means the statements will get executed again I will go back to L1 and that is essentially happening here. When the expression is true I will execute statements and come back but if the expression is false then what not of that expression becomes true then I go if go to will happen it will take and I will go to L2 skipping this statement in L1 so I will come to here.

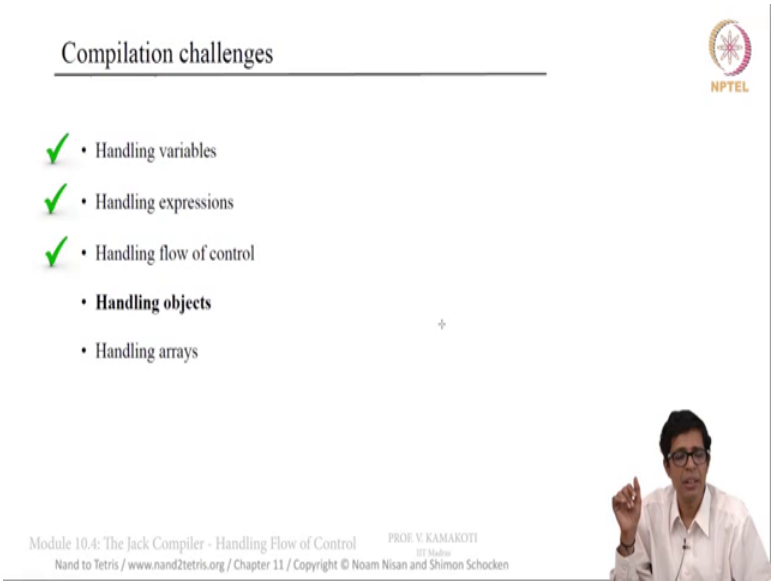
So if the expression is false I will skip statements and go after that so this is how a while statement essentially gets compiled ok.

(Refer Slide Time: 07:04)



So our compiler actually knows how to handle variables, expressions and flow of control now our compiler back end also knows how to handle functions, function call and return and memory allocation so now with this we can write a compiler for a simple procedure or language so this is the whole thing right so variables, expressions and flow of control are captured in your front end syntax and that thus those past rules are basically now translated into VM code and that is how we write this content ok.

(Refer Slide Time: 07:48)



So this is about flow of control and when we actually start constructing the compiler we will see more and more of how intricate details of how this are going to work right so we will basically go at that point of time. Now what we will do next in the next module is we are going to talk about how we are going to handle objects right.