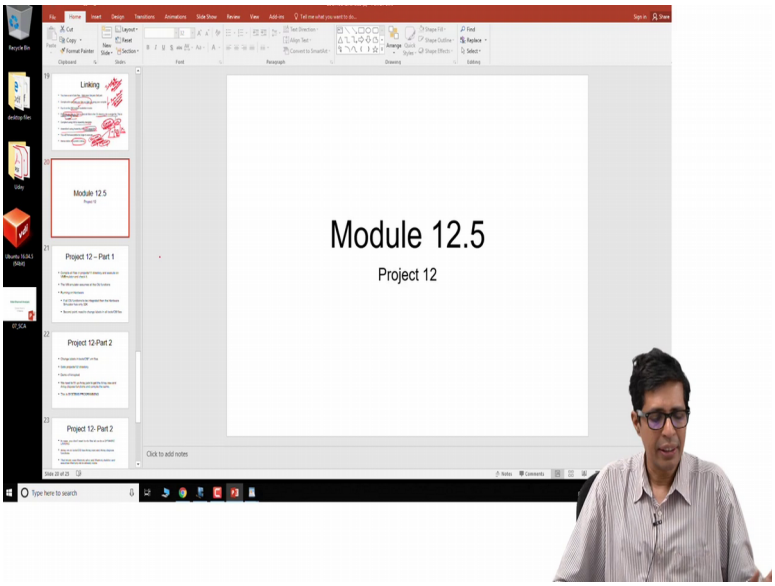
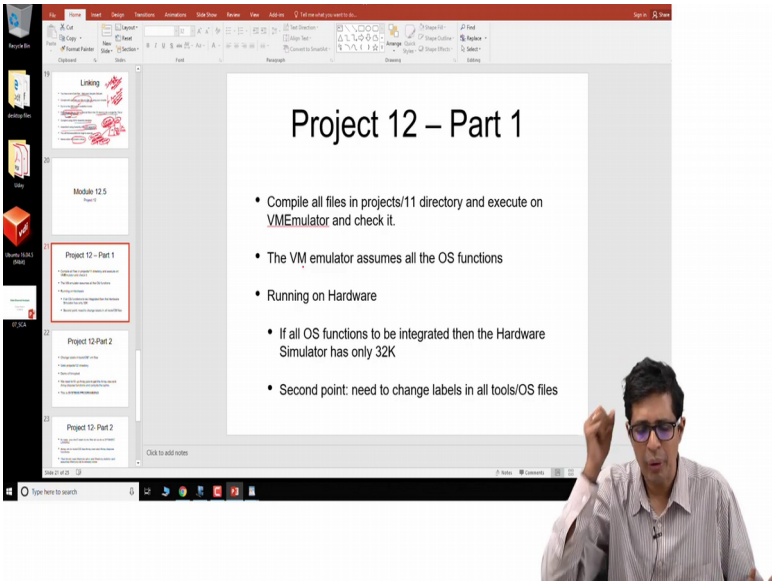
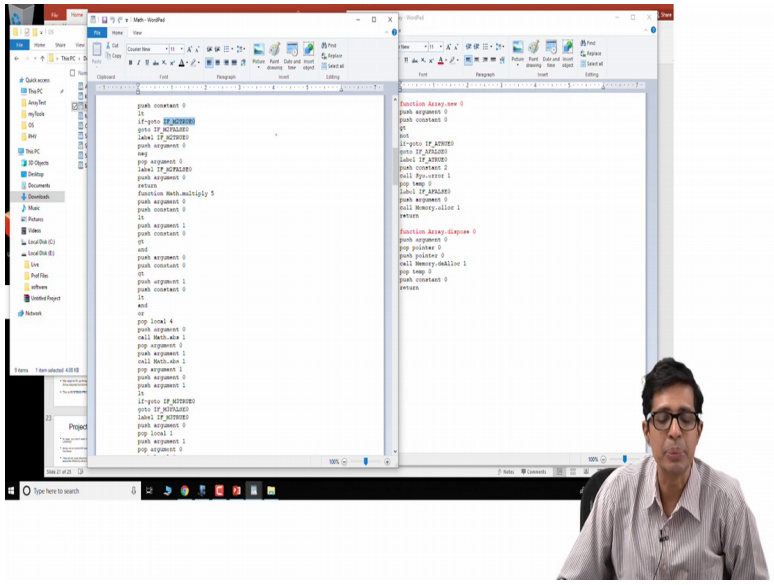
Foundations to Computer System Design
Professor V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Module 12.5
Project – 12: One sample journey from Jack to Hack

(Refer Slide Time: 0:18)







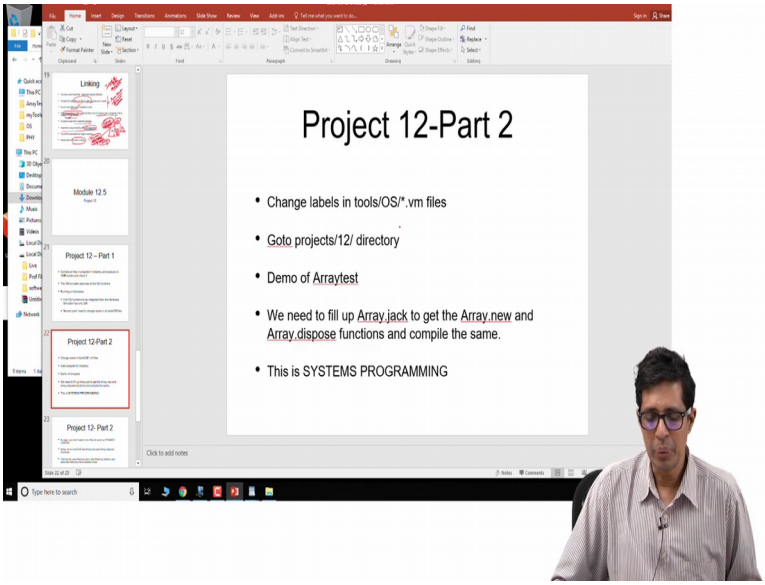
Welcome to module 12.5 and this is where we will be doing… So as explained earlier you would have compiled all the files in Project 11 directory and you will have executed on VMEmulator and you check it and similarly the VM emulator also assume all the OS functions which are running on the hardware. Now if all OS functions need to be integrated then the hardware sim R has only 32 kilobytes right and the most important point that we need to also understand is we need to change all the labels in all tools/OS files because when we link them together one of the things that we will see in the whole thing is that if we go to this OS if we see array you actually see that in the original file you will not see this a if true 0, if false 0, if true 0, if false 0.

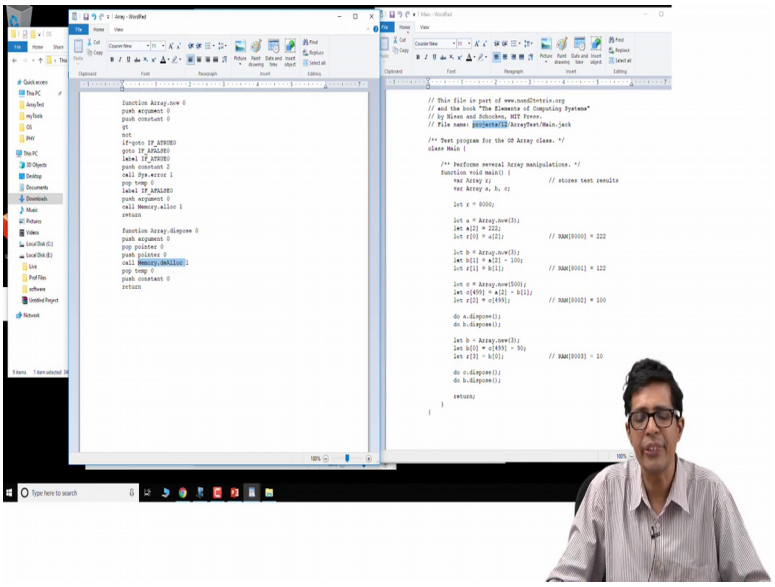
Now if you open some other map file also again you will see some where if you go here you will see if true 0, if false 0. When we put all these files together the labels will clash, so essentially one of the things we need to do is that we take every file and for every label you put in this case as only one function that is using the labels, so we just say if a true 0, if a false 0, et cetera but if you take this file where there is a map function, now here we say, so we just say the map.init is the 1st function, so whatever label I see in the 1st function I put M1 exp 0, M1 n0, I put an M1 here in the second.

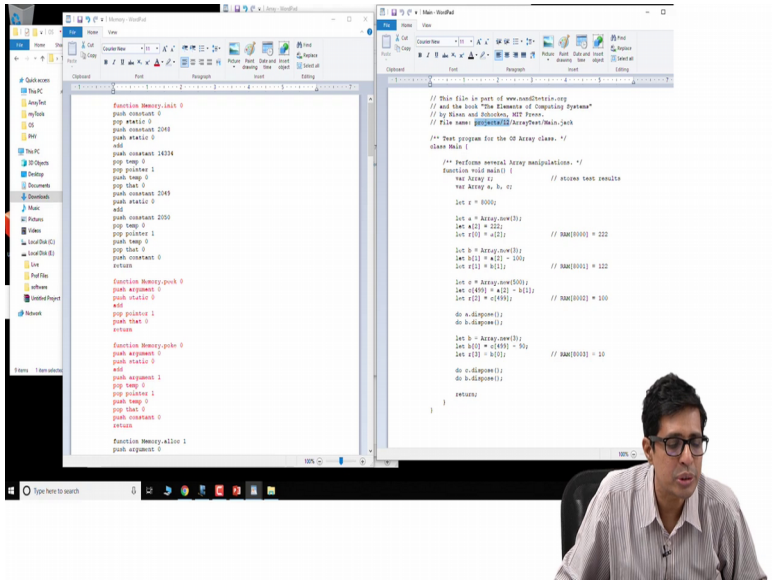
So this is the 2nd function whatever I see here I put M2 false 0 et cetera like that M3, so every label in every function of every file we would like to append a new value prefix a new value, new string there, so that the labels become distinct so that is one very important activity I spend around half an hour doing this as a part of this exercise, so then only when you put all these files together and do a linking then it can basically… The labels will not conflict, so

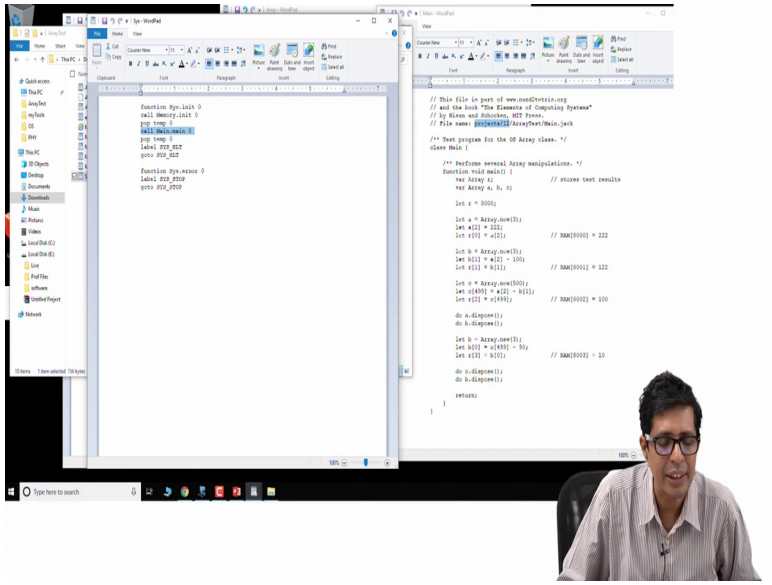
that is one another important thing that you need to keep in mind when you are doing this project.

(Refer Slide Time: 3:02)









Now what we need to do is that we go and change the labels the tools/OS/start.vm files stop now we go to Projects/12 directory, now there is something called Arraytest. In Project/12 directory you have an array.jack, now let us see what array.test actually does when you go and look at array.test here this is a jack file you will see a jack files there, the jack file actually uses 2 operating system functions namely array.new and since a, b, c or array.despose, so it is using only 2 operating system functions.

So it is only using array is why you need all the opening system function here as you see here are so many operating system functions like array, keyboard, map, et cetera why do I need all these things when just there is need for only array, so that is the first question so when we are doing the static linking, so we do not have the Hack compiler that we have does not have enough thing for hack machine cannot support a OS per se which can do dynamic linking,

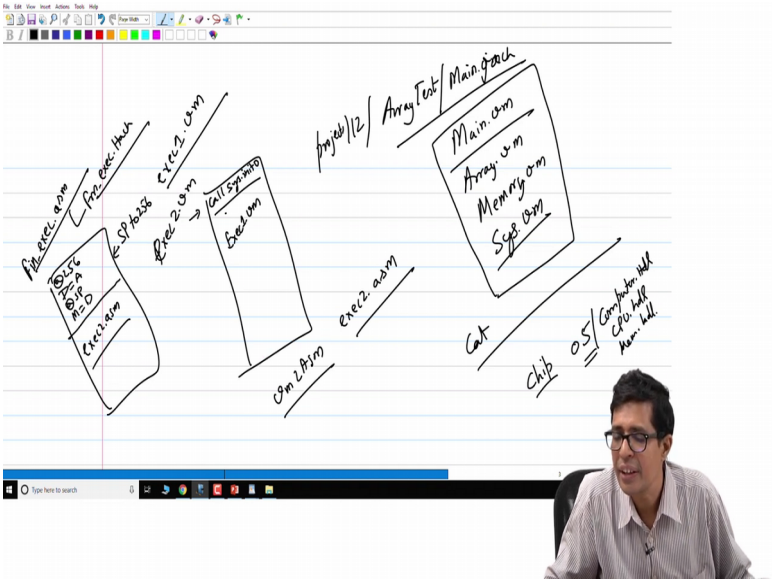
when you are doing static linking why do we need to put all these, so let us go and see what array has, so when you look at array, array.new and array.

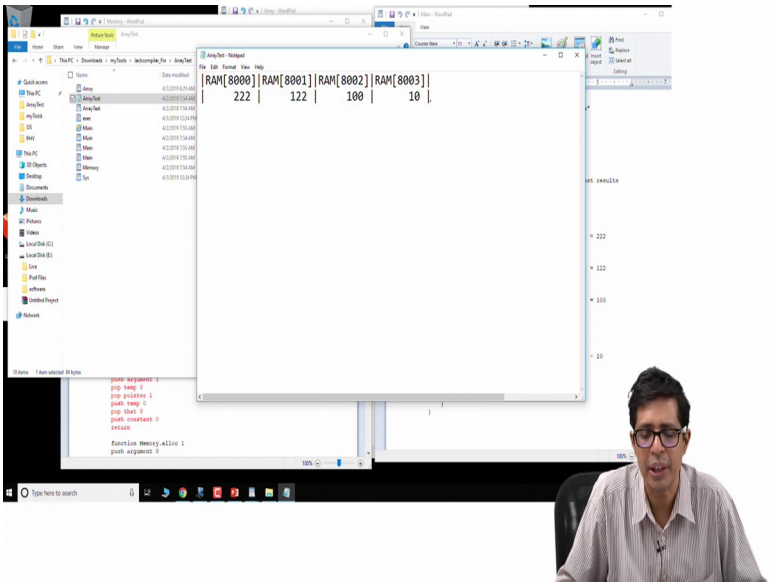
So array.new essentially uses to more functions like sys.error and memory.alloc and array.dispose basically uses memory.de-alloc, so these are the 2 things which it uses, so array basically now will have… Array actually uses 2 more functions from memory and sys, so let us now take memory so the memory now uses… first and foremost I need memory.init why I need memory.init I will explain in a moment but then the memory if you look at there are memory.peak and memory.poke which we are not using, so we can throw that of, so we need memory.alloc and memory.de-alloc which are there, so from the imaginary.vm I do not need memory.peak I do not need memory.poke I just need memory.init and memory.alloc but all these things are already used sys.error function, so what we have landed up is suppose I want to compile array and executed, so first I can just say array.

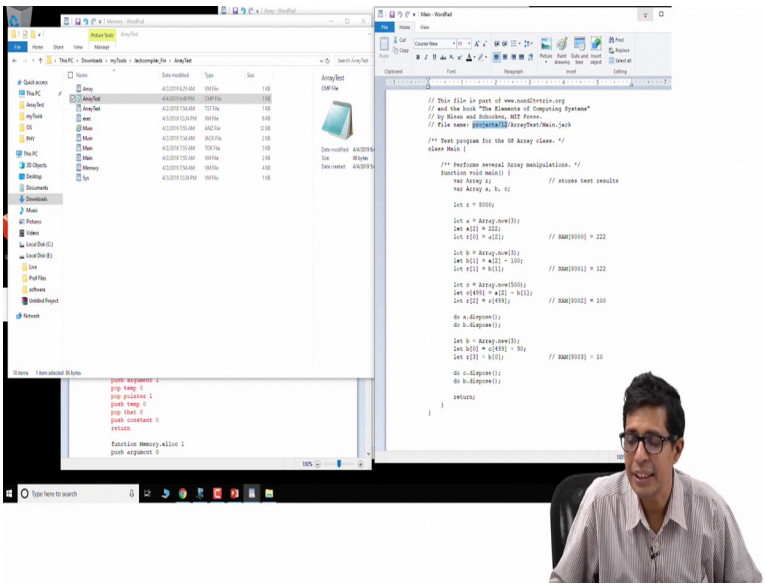
I can just compile this file it will give me main.jack it will give me main.vm but it inherently uses some operating system functionalities, so when I start compiling the array.new and array.dispose from array.vm I need to have memory.alloc and memory.de-alloc and sys.error and when I go to memory.vm, memory.alloc and memory.de-alloc assumes that memory.init has already been executed, so memory.init is necessary but I can throw away 2 functions memory.peak and memory.poke that I do not require here right and of course now we have this all of them use sys.error.

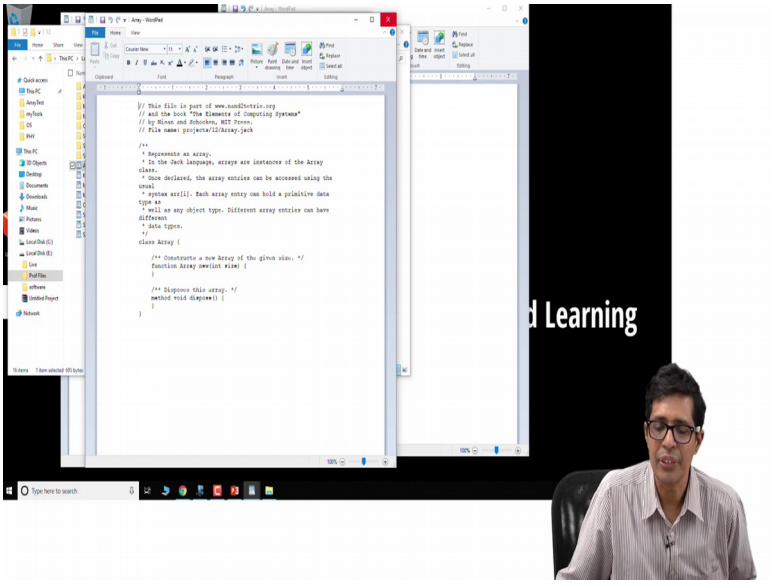
So I have to now start looking at sys.vm, when I go and look at sys.vm this is a huge file which uses all but what we want is when there is an error let us assume that I just want to go and halt at one place when there is an error, so I do not want to do this entire sys.init, so I write a short sys.init this will be our (())(6:44) sys.vm so what we have various there is a sys.init and sys.error. In the sys.init first I call memory.init, I will tell you why we need memory.init and then of course when you call a function by vm it will return something on the stack that (())(7:04) so I just pop tamp 0 then I call main.main, so main.main will execute when it returns I go to labels sis.halt and keep rotating here. Whenever I get sys.error I will go to sys.stop and this is an infinite loop that I will give, so this is how I get my sys, so now let us come to what we are supposed to do as a part of our entire staff right.

(Refer Slide Time: 7:37)









So I have array test this is 12 directory project/12 we have array test and we have main.jack. If I compile this using my compiler to get main.vm. Now I have my array.vm I just include as it is I have my abridged memory.vm where I have removed 2 memory.peak then I have my new sys.vm which has sys.init (())(8:09), so all these things I put together okay right. In addition to this I will just cat these files together, so this is called static linking, so all these things let me say this whole thing I am going to get it as an exec1.vm, so this exec1.vm the first thing I need to execute is call sys.init 0.

So that is the first thing that will call sys.init, sys.init will do memory.init and then it will call main.main so that is of this…so then I put all these exec1.vm here and let me call this as exec2.vm, so exec2.vm is exec1.vm preceded by this simple statement call sys.init 0 okay that should be the first statement. Now I do a vm to ASM assembler virtual machine 2 which will give me exec2.asm right so this will be the entire exec2.asm okay and before I start executing exec2.asm please remember I have to set a stack point of 256, so I do this at 256 then D equal to A at SP M equal to D.

This is a simple code I need to write to set the stack pointer to 256 then to that I append this exec 2.asm, so this will be final exec.asm. Now I use to assemble this to get fin exec.hack. Now go to the tools directory in the tool directory you will have the hardware simulator, open the hardware simulator, load the chip so you would have already created in project 5 you should have this computer.hdl and CPU.hdl and memory.hdl you will have these things, so immediately you can load the chip from project 5 and then on the right-hand side bottom you can load this asm file and you start executing it right.

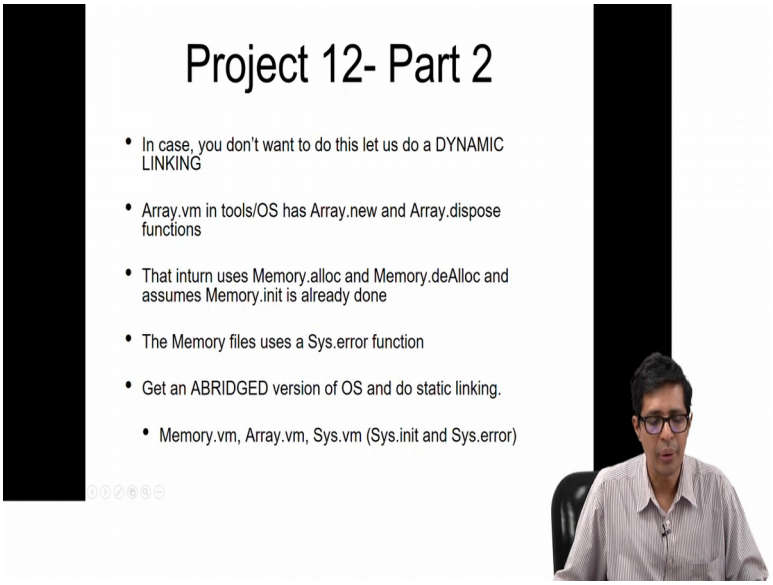
You start executing this array.test and it will take around 7000 - 8000 cycles so it will take around 20 minutes and at the end of this what you will see is that it RAM 8000, 8001 and 8002, 8003in the RAM you will see these value 222, 122, 100 and 10 right so that is what you will see here right, so this is what you will see here. In 8000 you should see 222, 8001 122, 100 and 10, so this is how you can validate this so now this basically tells you how to take a jack program and execute it on the machine.

The same thing you can do for other projects that are part of this entire story. In project 12 you do have after array test you have any other things you can do this same exercise and execute one by one and that will be very interesting exercise it will teach you how to program for IOT devices right you will not have large memory in the current Internet of things, you will not have large memories and so we need very small memory within that small memory, within that scope you need to basically start writing programs.

So that memory footprint that is what we call, the footprint is that size amount of memory that is available for execution and you need to start abridging cutting parts of the operating system to throw them and put it get a smaller executable that is a very virtual exercise and you should do it as a part of this, so there are multiple programs that are part of your project 12 which you can go and execute in this so you have array test, keyboard test, math test, memory test (())(12:57) test so you have 8 programs and all these 8 programs you can basically… and one of the interesting thing is that you also have certain jack files here.

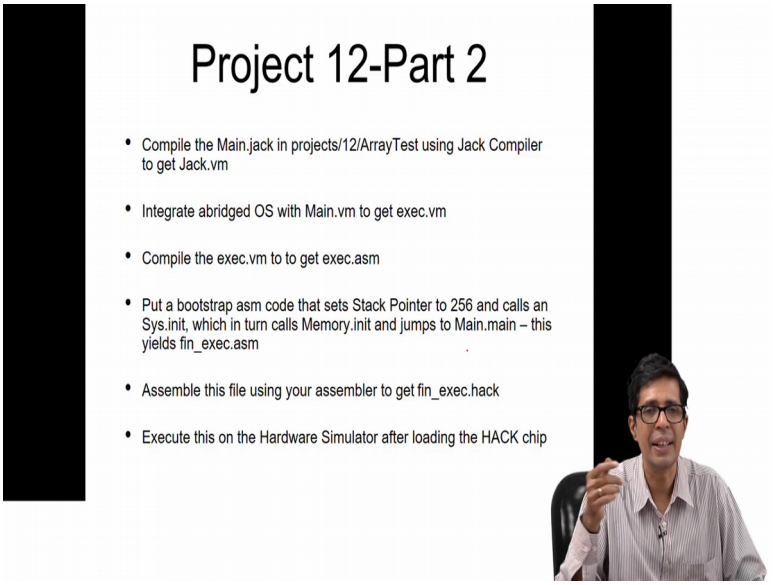
So there is something called array.jack and you have to…see if you look at array test the main function here basically uses array.new and array.dispose. Now what this project also wants you to do if you have time you can do that it will be very interesting is that we are giving you a sample template of array.jack where you need to fill up this array.new and array.dispose right so that is very important, so if you can start filling up this that means you have started writing code for the opening system right and this is basically called as systems programming, so this particular project allows you to write some system program and execute it also right then instead of relying upon the operating systems array.vm you can write your own array.vm and compile it and take it forward, so that is also an interesting exercise.

(Refer Slide Time: 14:17)



So this is something that you need to do as a project 12, so in this case since we cannot do dynamic linking we are doing this static linking and when we do this static linking we have the memory limitation, so we take part of the code of arrays and memory.vm, array.vm and we have our own sys.init and sys.vm and then we got a smaller executable and that will executed on the hardware simulator, the original hack that we started and then we go to program running okay.

(Refer Slide Time: 14:49)



We did had certain boot strap code in between where we just had a call sys.init 0 and then in the asm file we appended a code for setting up the stack pointer to 256, so these are all part of this okay. We come to the end of module 12.5.