Problem Solving through Programming In C

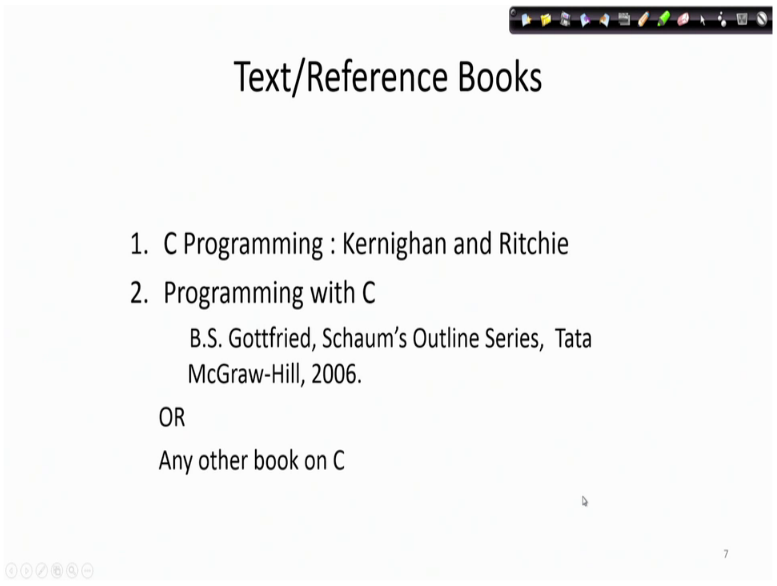
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 04
Introduction to Programming Language Concepts

So, in the last lecture we had seen how we can represent algorithms, and we had mentioned particularly two different ways. One is flowchart which is a diagrammatic way of representing the different steps, and also we saw English like form, which is known as a pseudocode. It is not exactly why is it called a pseudocode, it is called a pseudocode because it is not exactly the code that can run on the computer, but it is a way close to that and by which you can express ourselves, and from which we can also convert to the computer language.

Now, we had given a few examples of such flowcharts for a few problems. And I strongly encourage you to take up more problems from text books or some mathematical problems that you may encounter, and try to solve them; try to draw the flowchart of those.

(Refer Slide Time: 01:27)



Text/Reference Books

1. C Programming : Kernighan and Ritchie
2. Programming with C

   B.S. Gottfried, Schaum's Outline Series, Tata McGraw-Hill, 2006.

   OR

   Any other book on C

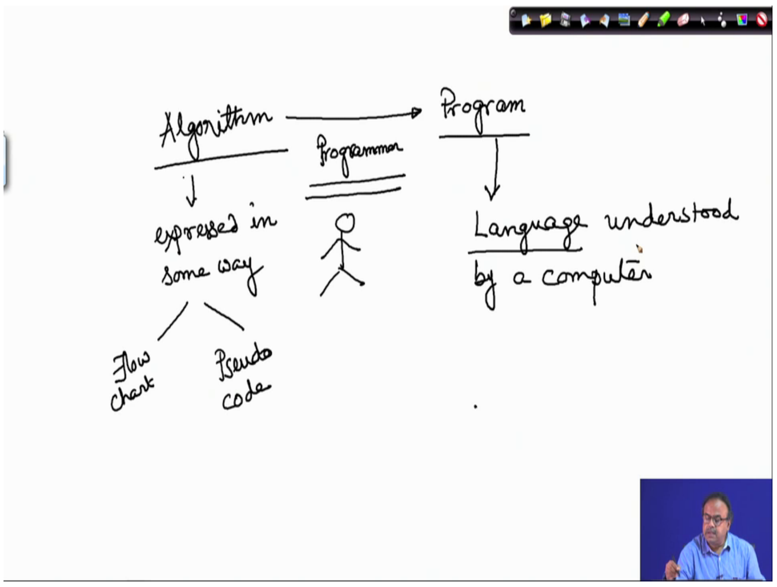
7

Now coming to the context of the books here are some of the suggested books, but I will start from the end that is any book on C will serve your purpose. You can take any good book on C language and try on that. And the number 1 that is C programming by

Kernighan and Ritchie is the most authentic book for C programming C language for that matter and, but I suspect that for some of you may find it to be a little difficult therefore, a middle path be a this second book programming with C by B.S Gottfried which is Schaum's outline series. So, and this second book has got a number of examples solved examples and examples given as exercises. So, I think that would a good very good starting point.

Now, having said that, what we have right now is we have some algorithm.

(Refer Slide Time: 02:37)

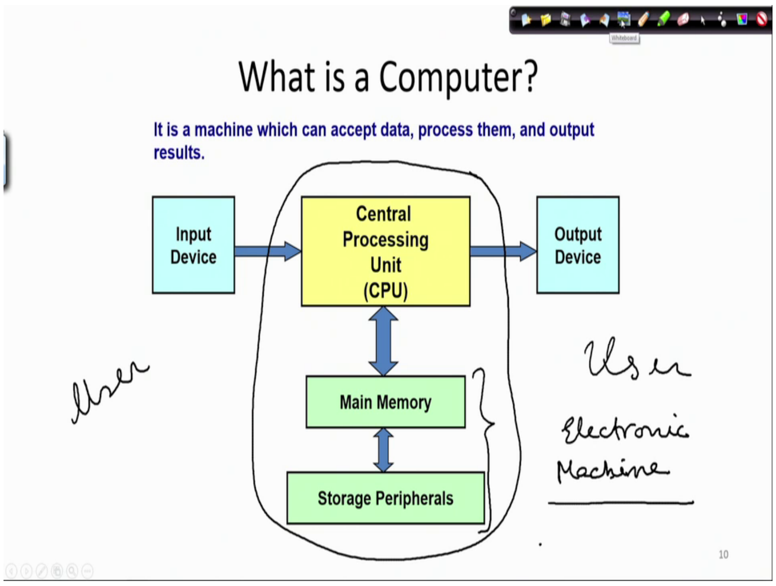


And from that algorithm, we want to come to a program. Now algorithm is a sequence of steps that might be in our mind also. So, I know how exactly I want to do that, I and I express it in some way and that is understandable. Now you know about 2 ways one is the flowchart another is a pseudo code which expresses how what are the steps that a to be taken. Now from this algorithm to a programmer program is a task of a programmer. Programmer is a person who knows how to write programs.

So, from the algorithm to the program whatever language the program might be that has to be translated. Now the key point to understand is that this algorithm I can express in any particular way; however, that program must be written in a language that is understood by a computer; alright language understood by a computer. So, a programmer is essentially a person sitting here who is translating the algorithm or the steps as expressed in some informal way in to a form, that is understood by the computer and this language just like any other language we will have it is own vocabulary, we will have it is own grammar.

Now, here comes the question what is the language that is understood by a computer.

(Refer Slide Time: 05:13)



What is a Computer?

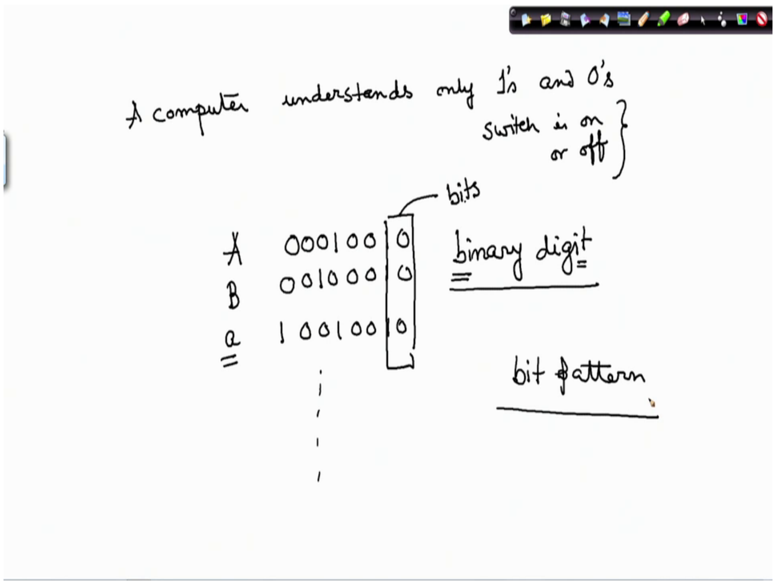
It is a machine which can accept data, process them, and output results.

Now in order to understand that, we have to look at what is there inside a computer; now all of you many of you may be knowing it from your school days, but just it is a quick revision of it is machine computer is nothing but a machine which can accept data process them and output the results. Now there is an input device which can be a keyboard. Nowadays it can be speech, microphone, it can be a joystick it can be many things right mouse is also another one. And ultimately we have got the computer here which consists of the CPU and the main memory and storage peripherals.

So, this is the actual part of the computer. Now the data comes over here, it is processed here and the output is sent to the user. So, the user is on this side the user is also on this side. So, this side is also the user and this side is also the user. Now there are couple of important things to understand here, typically we tell the computer how a problem is to be solved and that is the sequence of steps that we do, and whatever in sequence of steps that is specify. Now whatever sequence we specify that is remembered by the computer in it is memory. Now the memory can be actually it is stored in the secondary memory I will come to that later. But the key point to think about is that this entire box is nothing but an electronic machine and consequently it only understands 1’s and 0’s.

So, whatever we have to express; we have to express that using 1's and 0's.

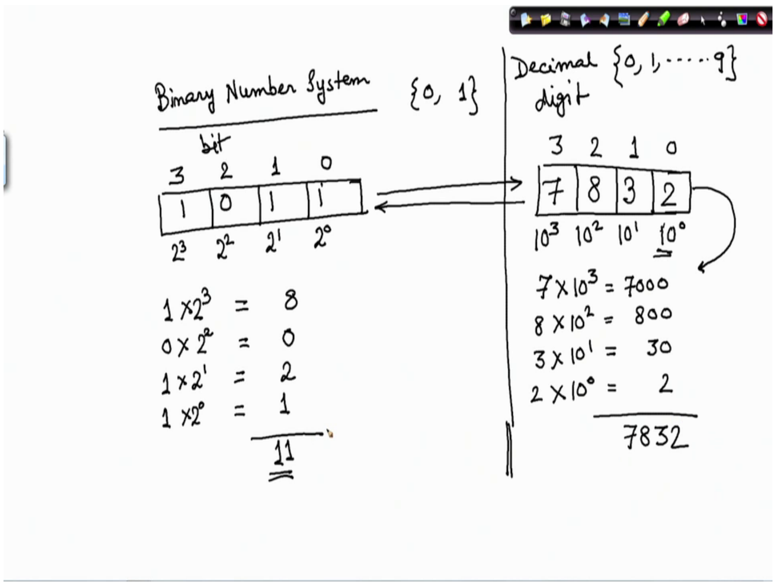
(Refer Slide Time: 07:41)



So, this is the first thing a computer being an electrical machine understands only 1's and 0's, because it is an electric machine, it only understands whether a particular switch is on or off right that is what it can at this understand. Therefore, whatever we have to express we have to express it in the form of 1's and 0's. So, consequently say A B all these alphabets small a everything will have to be represented in the form of 1's and 0's. So, maybe A just hypothetically it is not to be taken to be accurate, I am just saying that suppose 0 0 0 1, 0 0 1 0 is a pattern that represents A. So, similarly maybe 0 0 1 0, 0 0 1 0 this pattern represents b and may be 1 0 0 1 0 0 1 0 represents small a. In that way everything will be represented into a pattern of 1's and 0's and might be knowing some of you that each of these one and 0's these are known as bits, bit stands for binary digit binary digit.

So, bi for binary and bit that last t for this, binary digit from spades. So, it is a sequence of bits. So, anything, anything is represented in a computer as a sequence of bits or we often say bit pattern. Now just 1's and 0's gives rise to a new number system that is known as the binary number system.

(Refer Slide Time: 10:13)



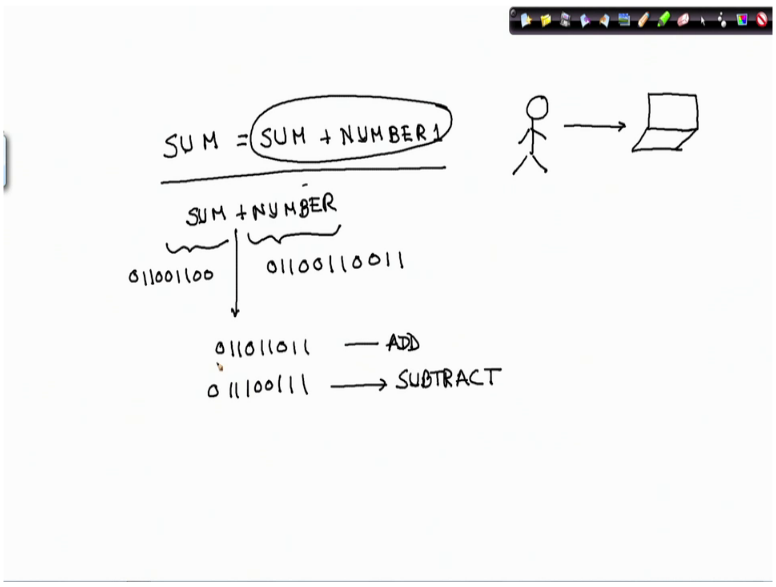
So, that leads to the binary number system might be some of you might be aware of this. Binary number system where I have got only 2 elements, 0 and 1 I have to do. Now in decimal visa we if I just consider that regularly we use decimal number system where we have got from 0 to like that up to 9, and I whatever I express as a combination of this. Now in binary number system everything is represented just by 0 and 1.

So, quickly let us look at; what are binary number system is, that suppose I have got a 4 bit number. So, there are four positions 1, 2, 3, 4 and suppose. So, con parallely let us think of a four digit number. Remember that in decimal we call it digit and in binary we call it bit right. So, if I have 7, 8, 3, 2; now all of you know what is the value of this number each of this place have got a weight, which are known as play I mean weights of the different places. So, this is 10 to the power 0, this is 10 to the power 1, this is 10 to the power 2, this is 10 to the power 3. 0 weight, 1 weight, 2 weights, 3 weight. So, this number is actually meaning 7 times 10 to the power 3 that is 7000, 8 times 10 to the power 2 that is 800. 3 times 10 to the power 1 that means, 30, and 2 times 10 to the power 0 that is 2 so that is giving us 7 8, 3, 2. It will be easier to understand this with this analogy for example, here I have got something like 1 0 1 1.

Now unlike; so here also the weights are 0, 1, 2, 3 but the base here was 10. So, here it is 2 to the power 0 is binary here. So, is 2 to the power one 2 to the power 2, 2 to the power 3, here the base is 2 here the base was 10, that is why this is a binary system this is a decimal system. So, if I have this pattern 1 0 1 1 then I can straightway convert it to an equivalent decimal like what does this mean. So, I can take it there is a 1 here, 1 times 2 to the power 3 that is what 8, then 0 times 2 to the power 2 that is 0, 1 times 2 to the power 1 that means, 2, 1 times 2 to the power 0 that is 1. So, the value of this will be 11. So, this is the binary number system.

So, there are ways and means by which I can convert from a binary number system to a decimal system, from a decimal system to a binary system that is possible right. So, if you want to see that I do not want to go into that because many of you have done it in school. So, let us leave it for the time being. But what is more important is to know that a computer in a computer everything is represented in the binary system as a bit pattern of 1's and 0's. Since everything is expressed as binary number system. So, if I remember the say we had something like this.

(Refer Slide Time: 15:06)



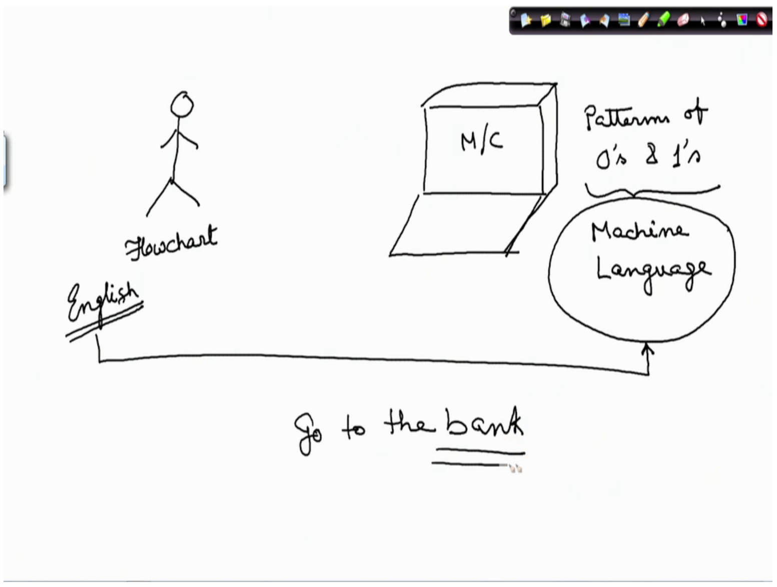
Let us not take max, let us take a sum equals sum number one.

Student: (Refer Time: 15:23).

This is something we saw in our earlier lecture something like that. Now this is a statement how do I represent it to a computer. In order to represent this to a computer say I want to say this is and you have to do some addition and some assign; some you have to copy it somewhere and write it somewhere. Since or I just take this simple part add sum with number, sum plus number how do I express that? Everything has to be expressed say this number have will be some patterns say 0's 1 0 1, 1 0 0 something of that sort sum will be something 0 1 1 0 0 1 1 0 0 whatever and class will also have some code 0 1 1 0 1 1 0 1 1 might be. Suppose this is add and suppose 0 1 1 1 0 0 1 1 1 is subtract similarly there will be a similarly there will be some code of for multiplication etcetera.

Now, since I am here, I am the user and I am also the programmer and here is my computer and I have to translate my thoughts my algorithm in a way so that a computer understands. I have to translate that algorithm or the flowchart into this pattern of 0's and 1's, because that is the only thing that it understands. The language that the computer understands is known as the machine language. So, let me once again draw another picture here.

(Refer Slide Time: 17:30)



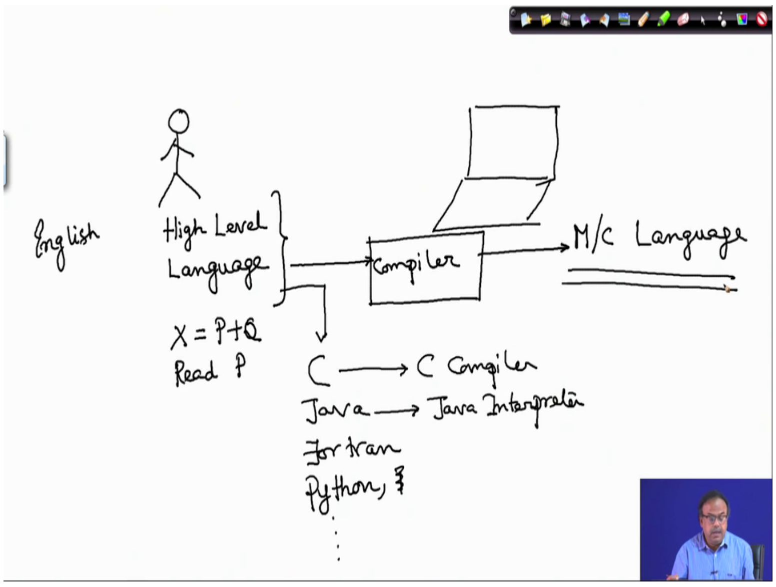
So, here I have got my machine I have got a machine here nice looking machine and here is the programmer. The programmer understands flowchart or the programmer understands English like pseudocode, but this one only understands patterns of 0's and 1's. A pattern of 0's and 1's, and this is a language it understands and this programmer understands the language of flowchart for example. Now, this patterns of 0's and 1's that the machine understands, this is my machine M C is the machine. Now this machine understands that language is known as the machine language, and that machine language is not understood.

If some genius can understand remember all those patterns of 0’s and 1’s for every possible combination, that is a different thing, but for normal people it is not it is very tedious, it is not possible it is not advisable also. Therefore, earlier people had to for small computers it was possible at the very beginning, people used to program the computer by setting up different switches. And thereby, it was being programmed. But at that time the operations and the capability of the computers say much more limited, but once it became larger we are doing much more flexible programming and everything, it is not no longer possible to remember the machine language.

So, what is the other alternative? The other alternative is the best alternative would be that on this side I have got my normal language English and if I could directly translate into machine language, but there are some problems with that because in English or any natural language that we use, many things are often ambiguous. It is not very clear it can have multiple meanings just I am giving an example go to the bank it is in English statement.

Now, what does this word bank mean? It can be many things right it can be in the bank where you deposit the money, it can mean the river bank etcetera. Therefore, English language or any natural language is not the yet suitable to be translated to the machine language. So, instead people thought if we could have some intermediate language. So, once again I come here, I again draw the machine here, the machine is here and the person is here.

(Refer Slide Time: 20:37)



So, on this side is my natural language like English, but which I am not being able to use. So, people developed some intermediate language which is close to English, but the grammar is much more stricter and we cannot allow any ambiguity no ambiguity. But it is high level language because it is easier for us to remember, it is easier for us to explain. For example, if I write this say X equal to P plus Q or I write read P now this is meaningful, this is looks like English and I can understand this.

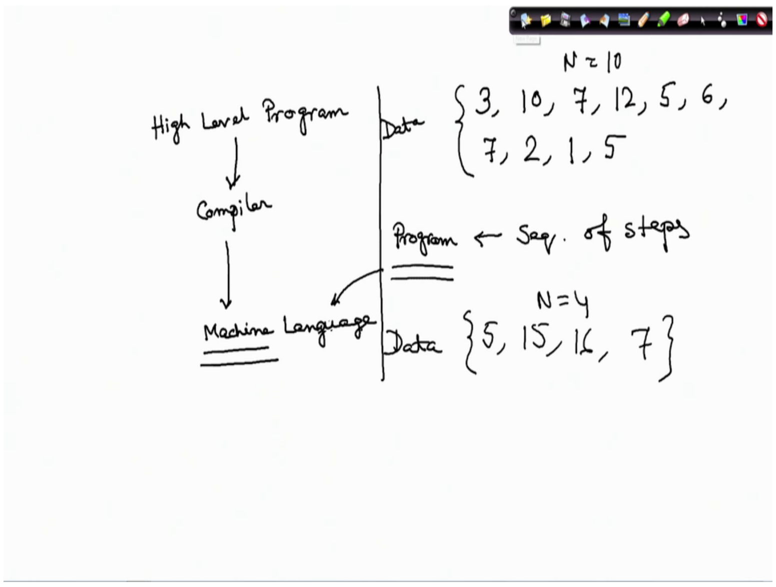
Now, given a high level language and on this side this machine can only understands machine language. So, we need to have some translator which will convert this high level language to the machine language, and this translator is known as the compiler. Now think of the situation that here is a person say just forget about the computer high level language, think of a person who knows French, and here is a person who knows maybe English.

So, a compiler or an interpreter is something which is somebody, who will translate from French to English right. So, here also depending on what language I am using as the high level language, there will be different compilers. Examples of high level languages are C which we will be discussing or taking as an example in our course; it can be Java, it can be FORTRAN, it can be Python, alright. We can have many other languages that are coming up nowadays other languages also coming up.

So, depending on the language high level language, in which we want to express our algorithm we will have to select a suitable compiler. So, for C we will need a C compiler, for Java will have to need a Java interpreter. The same thing the more or less same thing that it will translate the basic function of this is translation. So, it will translate from a high level language to the machine language. And as you had seen that here all those things I will be binary, I mean in binary system in the pattern of 0's and 1's, I have to convert if this machine has to operate with my program this have to process this my program I have to convert it into 0's and 1's.

So, now here is something more that we have to say. Now whenever we write a program what we do here is we write some high level program.

(Refer Slide Time: 24:52)



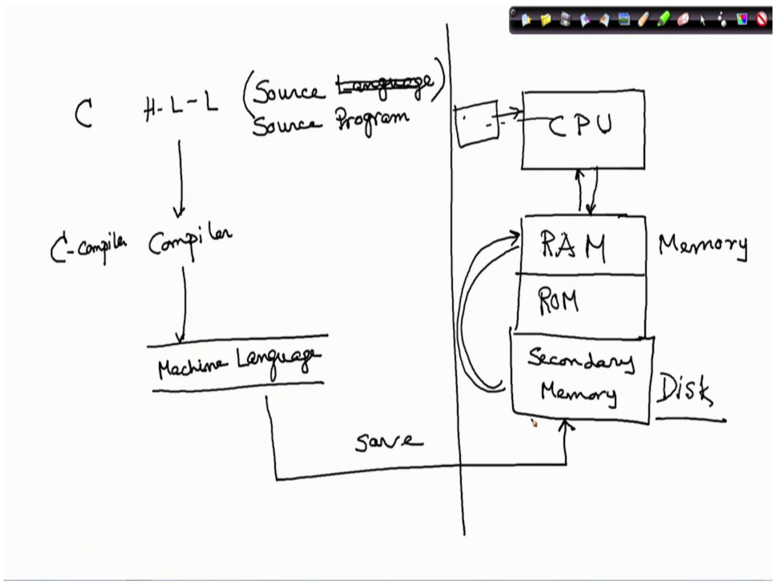
And that high level program is taken up by a compiler, and is converted into a machine language.

Now, here you have to be a little careful that depending on the different machines, you can have pentium machine, you can have apple machine and other things. So, depending on what machine you are using, you will you may need different compilers for that particular machine. So, for the time being let us ignored that thing, let us assume that one high level program can be translated by a compiler to a machine language. So, there is a program right. So, remember what we did in our earlier cases, we had we said find the maximum of 10 numbers, and the numbers were 3 10 7 12 5 6 so and so forth right 1 2 3 47 2 1 5. So, out of these I have to find out the maximum.

Now, through the flowchart or through the pseudocode what did we specify? We specified the sequence of steps. So, those sequence of steps that have to be adopted in order to find out the maximum of this is our program. And this program will be converted to the machine language. Now this program will run on some data this program is a program that can run on this set of data or another set of data say 5, 15, 16, 7 say for 4 data if I just run it for n number of things and here the N was 10. Remember what we did earlier and here N is 4. The same programme should work on both of these therefore, the program remains the same, but here is some data set and here is another data set. So, we have got 2 different data sets.

So, in order to execute we need both the program and the data. Now when we start say our program and the program when we start with the high level program.

(Refer Slide Time: 28:03)



I am writing H L L is the high level language, now this by the in the context of passing I am saying this is also known as the source language. So, in my lecture often I will be using this term source language. Source language means or I will say the I will not I will not say source language, I will say source program. By source program I mean the program that is written in the high level language that will be translated by a compiler to a machine level language. Now when we now this is one side. So, the machine language has been generated by the compiler. If my source language is C then I need a C compiler here, and I will get the machine language. And this machine language is independent of what the source language is, it is only dependent on the particular machine on which it is running. So, the compiler we translate into that machines machine language.

Now, in my computer diagram we had the CPU here and we had memory. Now the memory can be divided into 2 parts, one is the RAM and ROM some let me put it 3 parts RAM and ROM are 2 types of memories, which all of relatively smaller capacity I think this is not very clearly coming. So, let me write it clearly, RAM stands for random access memory and ROM stands for read only memory.

Besides that we have got the secondary memory, which is the disc. Now the machine language after compilation, when it is saved I save it and it is saved and stored in the secondary memory in the disc. Now when that program will be RAM, I have done the compilation and after compilation my program is ready to be executed is ready to run. Now when I run it, can I select some plain color; sorry. So, when I run it from the secondary memory it will go to the RAM and the CPU the program will move to the RAM only when I execute it, and the CPU you will read that program from the RAM and will execute it. So, this is a sequence you should remember I repeat once again. I write the program and type in the program in the machine, after that I do the compilation after compilation it is converted to the machine language and I save it. When I save it, it is saved in the secondary memory.

Secondary memory is typically the disc; I save it in the secondary memory. But on execution when I execute it, when I run the program then only it goes from the secondary memory to the RAM and the CPU reads from the RAM and executes it. So, we will be bothering about mostly this RAM and secondary memory, and how that is stored many variables and all those things are stored well in the next lecture we will see how we can gradually move towards encoding our pseudocode to the machine language to the high level language. So, summarizing what we discussed in this lecture, it is that we have got the pseudocode, but on the other side we have got a machine which does not understand this pseudocode, the machine only understands the patterns of 0's and 1's.

So, we have to have something which will convert the pseudocode cannot be directly converted because of lot of ambiguity. So, will have a high level language designed which can be which is unambiguous, and which can be converted to the machine code by another system that is a compiler. And that compiler is nothing but another more sophisticated program, using which you can convert this and keep it over there take it in the machine language. And the machine can then read that machine language program and can execute it.

Now we have also seen that a program I mean if we have to run a program we need the program as well as the data both this are stored in this memory part and from which the CPU will be taking that. Sometimes the data can be fed directly from the input device at runtime. So, whenever we say read n then from the input device somebody will type in the value and that will come from to the CPU into the RAM. So, this part will see later gradually.

Thank you very much.