


An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 02 Part - 5
Problem Solving Using Computer
Machine Language and How a computer works

(Refer Slide Time: 0:34)

What we discussed

- Memory, ALU, Peripherals communicate with the rest of the world through – "Data port", "Address port", "Control port"
- Control unit places values on control ports of other devices and tells them what to do.
- Control unit arranges movement of data between other parts of the computer.
- Control unit knows what to tell others by reading a "machine language program"




Welcome back. In the last segment we discussed the different parts of a computer, in particular memory ALU peripherals and the Control Unit and we noted what their respective functions were. And we said that the Control unit knows what to tell the other parts of the computer to do by reading a machine language programme. So in this segment we are going to talk about this machine language programme which is made up of machine language instructions.

(Refer Slide Time: 0:58)

Machine language instruction

Machine language instruction = sequence of numbers

- Possible structure of machine language instruction:
 - First number = says what operation to perform (“**operation code**”)
 - Second and third numbers : addresses in memory from where the operands are to be taken
 - Fourth number: address in memory where the result is to be stored.
- Machine language instructions are designed by the computer designer.
 - A machine language instruction will be designed for every operation the computer can perform




Okay, so, what is a Machine language instruction? It is a sequence of numbers and here is a possible structure or possible format for a machine language instruction. The first number might say what operation to perform and in this literature, you may often see the term “operation code”, the phrase “operation code” being used for this first number. So the first number tells what operation to perform or it codes the operation to be performed. The second and third number could be addresses in memory from where the operands are to be taken. And possibly the fourth number might be the address in memory where the result is to be stored, okay? So I just made up this format, but somebody has to make up a format. The machine language format as well as the instructions are designed by the computer designer. So the computer designer says that the structure is going to be this or some other structure, but just for concreteness let us just assume that the structure is exactly this. So the computer designer has to design instructions which will be able to perform every operation that we need or alternately a computer will be able to perform some operation only if the computer designer has designed a machine language capable of performing that operation or maybe a sequence of instructions which are capable together of performing that instruction.

(Refer Slide Time: 3:12)

(Fictitious) Examples of machine language instructions

- Hypothetical instruction: 57, 100, 200, 300
- Operation code 57 might mean "multiply"
- On reading the above instruction control unit does the following:
 - Tells the memory to read the words at the first two addresses and send them to the Arithmetic unit.
 - Tells the arithmetic unit to perform multiplication by sending appropriate number on its control wires.
 - Moves the result from the arithmetic unit to the memory
 - Tells memory to store the received word into the word at the third address
- This instruction causes the product of the numbers stored in addresses 100, 200 to be stored in the address 300.
- 58 might mean the same thing as above, except perhaps the numbers would be added.



So, just to make this, this description more concrete I am going to give an example, but of course this is a fictitious example, there is no computer which has the machine language instruction that I am going to talk about, but it is an indicative, it is an indicative machine language instruction. So my hypothetical instruction consists of these four number, the first number 57 is what we call the operation code and this might mean multiply for example let us say it means multiply. Then, on reading the above instruction and assuming that the format or in the structure that we described on the previous slide, what should the control unit do? Well the control unit will tell the memory to read the words at first two addresses and send them to the arithmetic unit because what we said in the previous slide was that the words at addresses 100 and 200 need to be multiplied so they are being sent to the arithmetic unit. Then, the control unit has to tell the arithmetic unit to perform the multiplication. For this it has to send the appropriate number on the control wire of the arithmetic unit which will cause the arithmetic unit to perform a multiplication. And finally, the control unit has to arrange for the result from the output port of the arithmetic unit to be moved back to the memory and then it has to tell the memory that look whatever you have just got on your data port has to be stored in this third address which in this case is 300 so that address also has to be sent back on the address port of the memory.


So these are roughly the five steps that the control unit has to do on seeing that instructions. Now this seems like a fairly complicated thing, but it will indeed cause the product of the numbers stored in addresses 100 and 200 to be stored in address 300, okay? Now you might have another hypothetical instruction in which 58 might mean the same thing as above, except that maybe the numbers have to be added. And I should point out one more complication, so when I say I want the numbers to be added, I need to really say what format the numbers are in. So maybe the 57 represents, says that look interpret the numbers as unsigned integers and then do the addition and store them back, okay?

So, so the operations actually end up being somewhat complicated because of the different number formats that we have. The machine language instructions have to be sensitive and have to specify whether the operation has to be with respect to one kind of number representation or another kind of number representation. Okay, so I gave you an example of a machine language instruction, now I am going to give an example, again a hypothetical example of machine language programme.

(Refer Slide Time: 6:18)

Machine language program (hypothetical) example

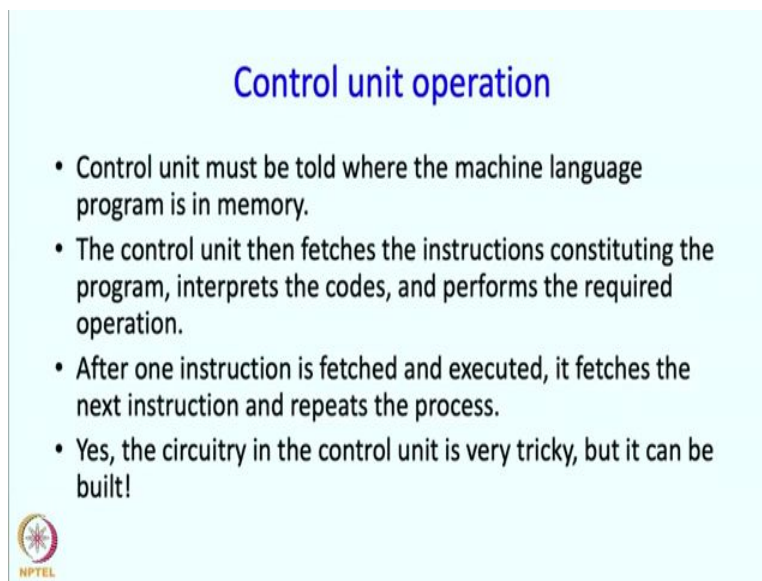
- Example: 57, 100, 100, 100, 57, 100, 100, 100
- This contains two instructions.
- Both instructions cause the word at address 100 to be multiplied by itself and the result stored back in address 100.
- After executing the first instruction, address 100 would contain the square of the number that was present before.
- The second operation would repeat the squaring operation.
- Thus this is a machine language program to compute the fourth power of a number.



So here is a machine language programme, a really simple machine language programme, it only contains two instructions. So the first instruction is 57, 100, 100 and the second instruction is also 57, 100, 100. So if the programme is present in memory, the control unit will first execute


the first instruction and then it will go and execute the second instruction. That is how the control unit will behave. So suppose the control unit does that, what will happen? Well both instructions are really identical so both instructions will cause the word at address 100 to be multiplied by itself because the first word as well as second word, the second address is the same and in fact the destination is also the same. So the words in address 100 will be multiplied with itself and stored at 100. So after one instruction is executed the address 100 would contain the square of the number that was present before. And so what happens after the second instruction is also executed? Well it would repeat the squaring operation so as a result at the end we will get the fourth power. So you can say that this sequence of two instructions is really a machine language program to compute the fourth power of a number.

(Refer Slide Time: 7:59)



Control unit operation

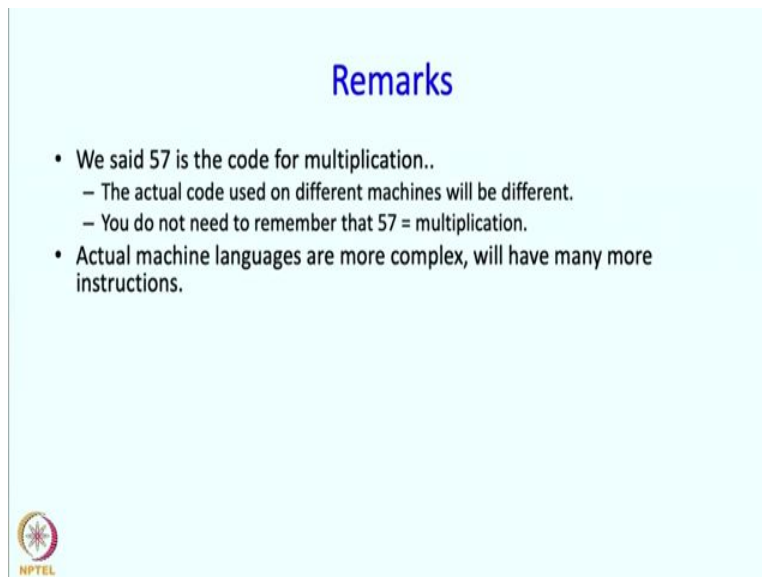
- Control unit must be told where the machine language program is in memory.
- The control unit then fetches the instructions constituting the program, interprets the codes, and performs the required operation.
- After one instruction is fetched and executed, it fetches the next instruction and repeats the process.
- Yes, the circuitry in the control unit is very tricky, but it can be built!



Okay, so here is quick exercise for you just play with this, with this notion. So you are expected to modify the program, which we gave you earlier so that if that instead of computing the forth power it computes the cube of the number stored in address 100 and instead of storing the result in 100 itself, it is going to store it in address 200. Okay so now I have told you what a machine language program is and now I want to complete the description of control unit, how it operates. So the machine language program is going to be somewhere in memory and the control unit must be told that look from this address the machine language program is present. So what does the

control unit do? The control unit fetches the instructions constituting the program, interprets the code and performs the required operations. After one instruction is fetched and executed, it fetches the next instruction and just repeats the process, that is what the control unit does. And, there could be more complicated instructions of course which say to the control unit that look, repeat these instructions several times, so something like a repeat statement could also be there in machine language and something like a conditional statement could also be there in machine language. But we are not going to worry about all those details, this is enough level of detail for our purposes. And yes, the circuitry in the control unit is quite tricky to design, but it can be built and it has been built.

(Refer Slide Time: 9:31)



The slide has a light blue background. At the top center, the word "Remarks" is written in a blue, sans-serif font. Below this, there are three bullet points in black text. The first bullet point is "• We said 57 is the code for multiplication..", followed by two sub-bullets: "– The actual code used on different machines will be different." and "– You do not need to remember that 57 = multiplication.". The second main bullet point is "• Actual machine languages are more complex, will have many more instructions.". In the bottom left corner, there is a small circular logo with a red and white design, and the text "NPTEL" below it.

Remarks

- We said 57 is the code for multiplication..
 - The actual code used on different machines will be different.
 - You do not need to remember that 57 = multiplication.
- Actual machine languages are more complex, will have many more instructions.

NPTEL

Okay so some remarks, we said that 57 is the code for multiplication, of course this is fictitious, again I just want to emphasize that. Different machines will have different codes, and of course you do not have to remember that 57 means multiplication, it does not, we just made up that number, just as an example. And actual machine languages are a lot more complex, will have many more instructions, but the point of this discussion is for you to understand what is machine language. So this is indicative of the types of instructions that you have in a machine language and therefore, this sort of tells you what a machine language program looks like.


Now, machine language programs are actually important in the sense that when you execute a program, what actually executes on the computer is a machine language program. And if you go back to early computers, as a user you would have to write the machine language programs, what does it mean? So then you have to decide what operations you want to perform. Then you go and look up the manual and find the code which will cause that operation to be performed.

So you enter the code into the memory of the computer, then enter the address of the operands and the result - where the operands are going to be present, where the result should be put, you enter those addresses. And then you repeat this for as many operations as you want done. Now this process is really, it looks really painful and it actually is painful, not only it is laborious and painful but it is also very error prone. So thankfully, in this course we are not going to do any of that.

(Refer Slide Time: 12:03)

Machine language programs and C++ programs

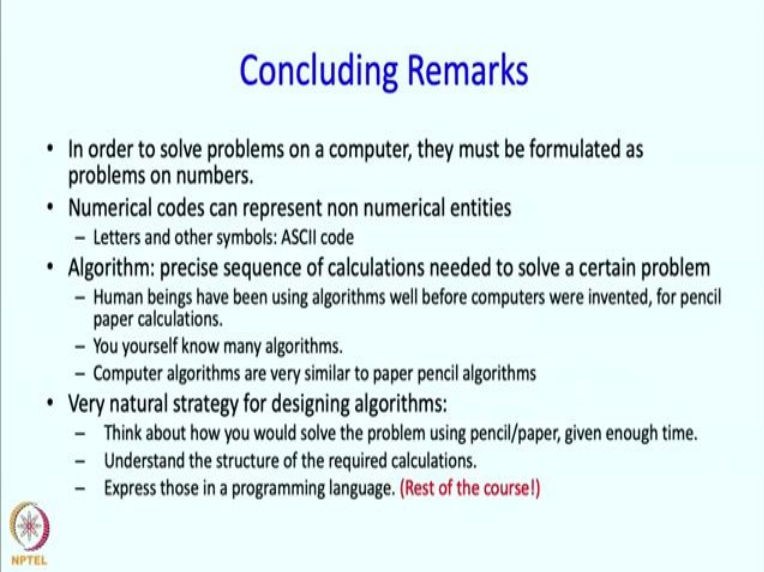
- On a modern computer you write a C++ program.
- A prewritten program, "**compiler**", translates your C++ program to a machine language program.
 - When you type `g++ square.cpp` the compiler is called upon to compile your file `square.cpp`.
 - It creates the "machine language program" which by default is called `a.out` on unix.
- When you type `./a.out` :
 - `a.out` gets loaded into memory by the "**loader**" program
 - Then `a.out` executes.



So now I want to tell you the relationship between machine language programs and C++ programs that we are going to teach you how to write. So on a modern computer you will write a C++ program and there is already a program written called a compiler which will translate your C++ program to a machine language program. So it will generate the equivalent machine language program. When you type 'g++ square dot CPP' or when you type, when you hit the compile button, this is exactly what happens, your C++ program is translated into an equivalent


machine language program. So on Unix the machine language program is called a.out. And when you type './a.out', a.out gets loaded into the memory by a loader program and it executes. And similarly, if you hit the run button on your IDE the created machine language program will be loaded into memory and it will be executed.

(Refer Slide Time: 12:58)



Concluding Remarks

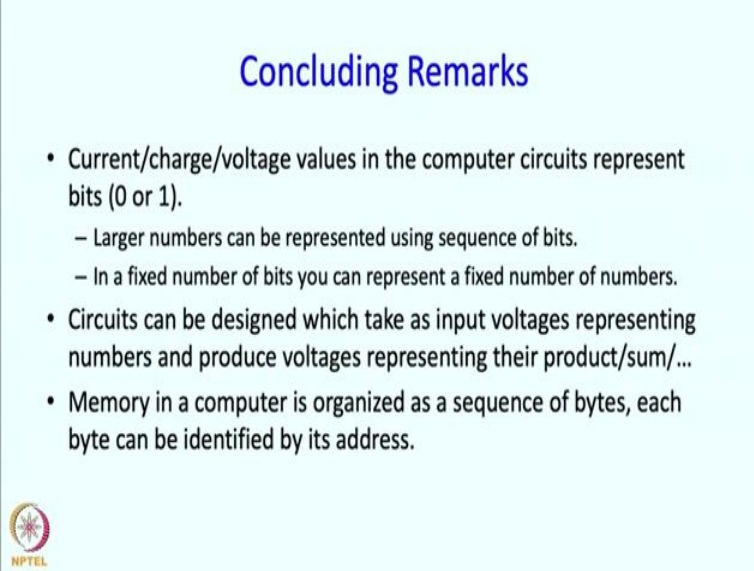
- In order to solve problems on a computer, they must be formulated as problems on numbers.
- Numerical codes can represent non numerical entities
 - Letters and other symbols: ASCII code
- Algorithm: precise sequence of calculations needed to solve a certain problem
 - Human beings have been using algorithms well before computers were invented, for pencil paper calculations.
 - You yourself know many algorithms.
 - Computer algorithms are very similar to paper pencil algorithms
- Very natural strategy for designing algorithms:
 - Think about how you would solve the problem using pencil/paper, given enough time.
 - Understand the structure of the required calculations.
 - Express those in a programming language. (Rest of the course!)



So we really have covered a lot of ground at a very high level, a bird's eye view so to say, so what have we learned? We have said that in order to solve problems on a computer they must be formulated as problems on numbers, then if things are not immediately numerical then we have to use some codes, say for example for language or text we need to use codes, okay. Then we said that algorithms are precise sequence of calculations that are needed to solve a certain problem. And we said that algorithms are not new, human beings have been using algorithms well before computers were invented. And these algorithms were used for pencil and paper calculations but that is fine. Some of these algorithms are terrific algorithms and not only that, you know many of these algorithms already. Computer algorithms are very similar to algorithms which are needed for pencil paper calculations. So a very natural strategy for designing algorithms it is as follows, think about how you would solve the problem using pencil and paper given enough time. Understand the structure of the required calculations and then express those


calculations in the programming language, and this last step is what the rest of the course is all about.

(Refer Slide Time: 14:38)



Concluding Remarks

- Current/charge/voltage values in the computer circuits represent bits (0 or 1).
 - Larger numbers can be represented using sequence of bits.
 - In a fixed number of bits you can represent a fixed number of numbers.
- Circuits can be designed which take as input voltages representing numbers and produce voltages representing their product/sum/...
- Memory in a computer is organized as a sequence of bytes, each byte can be identified by its address.




Some more remarks, we also talked about how numbers are represented and we have said that numbers are represented using currents, charge, voltages. Larger numbers, larger than 0 or 1 can be represented using sequence of bits, or sequence of 0's or 1's. And an important point to remember of course is that if you have a certain fixed number of bits, then you can only represent fixed number of numbers. So if you want to represent a larger range of numbers or if you want to represent numbers more precisely you will need to use more bits.

Then we said that circuits can be designs so that they take as input voltages representing numbers and produce voltages representing their products sum or whatever you want. We also said that memory in a computer is organised as a sequence of bytes, each byte can be identified by its address.

(Refer Slide Time: 15:28)

Concluding Remarks

- Machine language program : sequence of machine language instructions
 - Must be present in the memory
- Control unit reads machine language instructions and interprets them
 - Decides what needs to be done
 - Sends control signals to other units and makes them do the needful
- Users write program written in high level language e.g. C++
 - User program compiled into machine language by **compiler**
 - **Loader** loads compiled program into memory and then it executes.



Then we said that a machine language program is a sequence of machine language instructions and this program must be present in memory. And the control unit reads machine language instructions and interprets them, and decides what is to be done, and sends control signals to other units to make them do the needful.

Then we said that the users write a program written in high level language, say C++ and user program is compiled into machine language by the compiler. Loader loads the compiled program into memory and then executes it, that is the end of this second sequence of lectures for the first week.

Thank you.