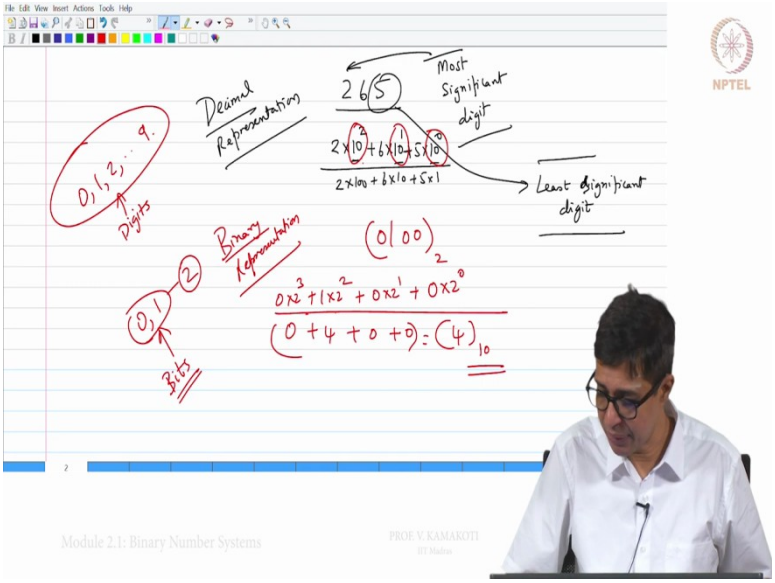
Foundations to Computer Systems Design.
Professor V. Kamakoti.
Department of Computer Science and Engineering.
Indian Institute of Technology, Madras.
Module 2.1.
Binary Number Systems.

So, welcome to module 2.1. Before proceeding to this module I hope that you have done the project1 which was suggested at the end of module1. So, this module will become very meaningful if you have completed that project because it assumes a lot of things that you have, you have done in that project, knowledge of that will make this module very easy. So, if you have not done this project, before you this video, please go and finish that project, hardly takes 1 to 2 hours max.

(Refer Slide Time: 1:00)



Okay. So what we will do in this module 2.1 is first we will talk about in this particular thing we will talk about binary numbers and will also talk about the finiteness of computing. So, whatever we try to do on a computer, there is a limit on what we can do and we will understand that limit in its proper perspective as we proceed in this module. So what is a binary number? So we have actually seen some decimal numbers in, so if I say 265, what does it mean, this is what we call is a positional representation.

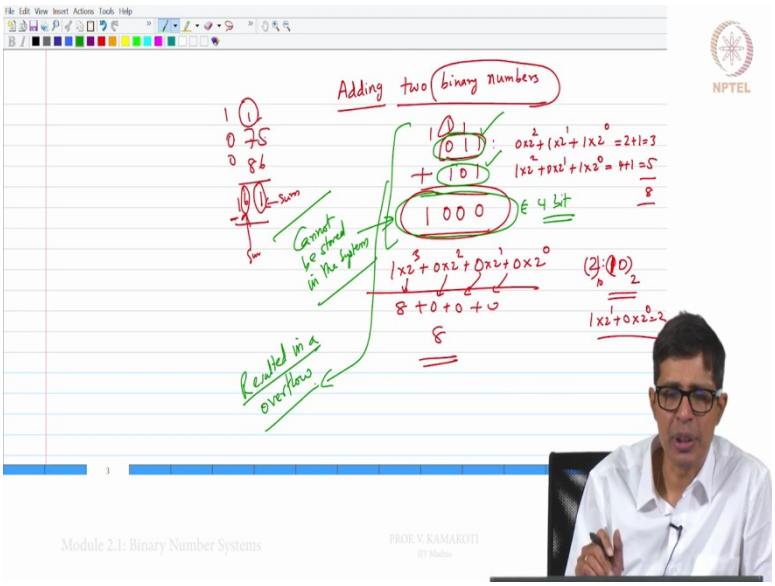
So every digit has a value depending on on its position, right. So, there are 3 digits here, there is 2 which are called the most significant digit. It is called most significant because the contribution of 2 to the entire value 265 is the largest. So 2 contributes 200 to the 265, so that

what is called the most significant digit. So this is 2 into 10 Power 2+6 into 10 Power 1+5 into 10 Power 0. And so that is how this is 2 into 100 + 6 into 10+5 into 1 it gives you 265.

So this 5 is called the least significant digit, because it cut puts the least, for the contribution of 5 to 255 is just 5, while the contribution of 2 to 265 is 200. So this is the decimal representation or base 10 representation, I am sure you would have read this earlier many times. This is called decimal because every digit is multiplied by a power of 10 as we see here. Now, what is binary representation, binary representation is that these digits will be multiplied by a power of 2.

If I multiply, a decimal representation will have 10 digits, namely 0, 1, 2 to 9. Now a binary representation will have only 2 digits, namely 0 and 0 because everything else is going to be represented as a power of 2. This 0 and 1 are called bits, like 0 to 9 are called digits in decimal representation, 0 and 1 are called bits in binary representation. So if I say 0100, what is its decimal representation, what is the value this will give in decimal arithmetic is nothing but 0 into 2 power 0+0 into 2 power 1 +1 into 2 power 2+0 into 2 power 3. Why are we multiplying with power of 2, because it is the binary representation.

(Refer Slide Time: 5:04)



This will give me 0+4+0+0, which is nothing but four to the base 10. So this is the decimal to binary, okay, binary to decimal conversion. Now let us go and do the next step. We will not talk about adding two binary numbers. So, how do we add two decimal numbers? Suppose 75+86, I add the first 2 digits, 5+6 is 11, then I take a carry here, add this here, so this I get

161. So I had the first 2 digits, get a sum digit, this is called the sum, the sum is 1 and the carry is 1.

5+6 is 11, 1 is the sum and 1 is a carry. Now that carry I had with 7 and 8 to get 16. So this 6 is again the sum, another 1 is carry and since these are all zeros, this will become 1 and you end with 161. Similarly in binary also we will add. Suppose, so I will be adding bits, like how I added two digits here, I will be adding bits. And at most how many digits did I add here? I added 3 digits here. Right, 1+7+8. Similarly I will also be adding at most 3 bits in binary when I do that.

So let me say 011 + 101. So what is 011? Let us do this, one into 2 power 0 + 1 into 2 power 1+0 into 2 power 2, which is nothing but 2 + 1 equal to 3. This is 1 into 2 power 2 +0 into 2 power 1+1 into 2 power 0. This is nothing but 4 +1 equal to 5. So totally I have to get it. Now let us do this. 1 +1, it is going to give me 2, which is 01, note that 2 is 10 in binary. 2 to the base 10 is 10 to the base 2, right. This is 1 into 2 power 1+0 into 2 power 0 is 2.

So when I add 1 +1, I get 10, which is equivalent to 2. Now again I add 1+1+0 which is again 2 which is again 0 and 1. I add 1 +1 with this, again 2, so I get 0, the carry, extra carry is 1 here, which comes here and so the answer is 1000. What is 1000? So 1 into 2 power 3+0 into 2 power 2 +0 into 2 power 1+ 0 into 2 power 0, which is nothing but 8+0+0+0, so which gives you 8. So, this is how we add 2 binary numbers. Now, we need to have one understanding here.

Suppose my computer, the computer stores, the computer as we see in the gates, the computer cannot recognise anything other than binary, because the entire computer is working using transistors, had we had seen in the module 1. And what will the transistor understand? And everything we did, the gates we did, right, we are basically switching on a transistor or switching off your transistor. So basically the transition can understand only 0 or 1s. So all the operations that are going to be executed by the hardware has to be using 0 or 1s.

That is why we are not using a decimal system, all the things that we are trying to do, we are doing it using a binary system. Why are we using the binary system which and all because the computer can understand only zeros and ones as we had explained in the module 1. So, all our arithmetic will be done using zeros and ones. Now, how many bits, so these zeros and ones are called bits as I had mentioned. How many bits a computer can store?

For example if the computer can store only 3 bit numbers, the computer obviously cannot store an infinite bit number. For example the 011 that you are seeing here, which I am marking in green here, the 011 is a 3 bit number, 101 is a 3 bit number, 1000 is a four bit number. The computer cannot store infinite bits, so there is a limit on the number of bits a computer can store or handle. If a computer can handle only 3 bits at a time, then what happens, this 011 is a 3 bit number, 101 is a 3 bit number, 1000 is a 4 bit number.

If the computer can only handle 3 bits at a time for a number, then 1000 cannot be stored in the computer. 1000 cannot be stored in the system because the system can store only up to 3 bits. So this particular addition has resulted in what we call as an overflow. This system, this particular addition has caused an overflow because it has created an integer, it has resulted in an teacher which cannot be representable by 3 bits. And the computer today, in this context is, can represent only numbers with 3 bits.

Now the answer of this addition has resulted in a 4 bit number and obviously this has resulted in an overflow. Right, so what we have learnt in this particular module is that we have basically taken the binary representation and we have looked that the addition of 2 binary numbers and importantly we have understood what you mean by the finiteness in the representation of a system. In this case the finiteness has ended resulted in an overflow for this particular addition. Now, in the next module, that is 2.2, we will start looking at the signed binary number representation. Thank you.