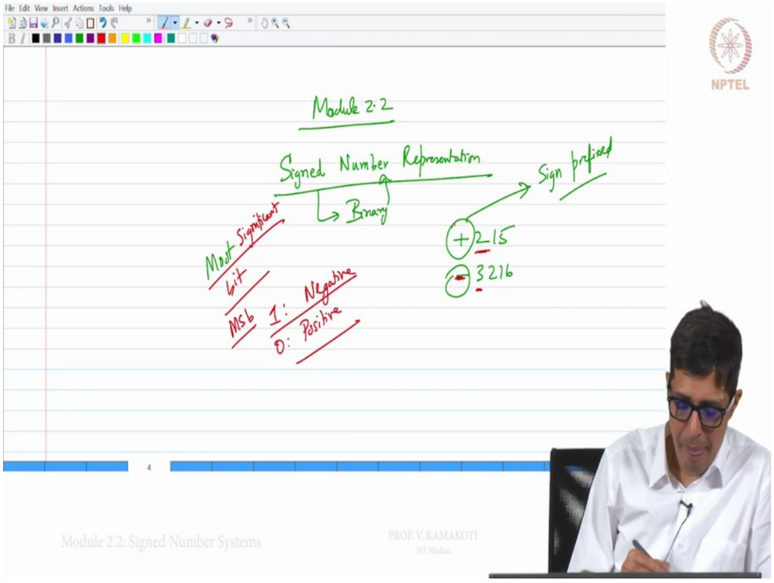
**Foundations to Computer Systems Design.**
**Professor V. Kamakoti.**
**Department of Computer Science and Engineering.**
**Indian Institute of Technology, Madras.**
**Module 2.2.**
**Signed Number Systems.**

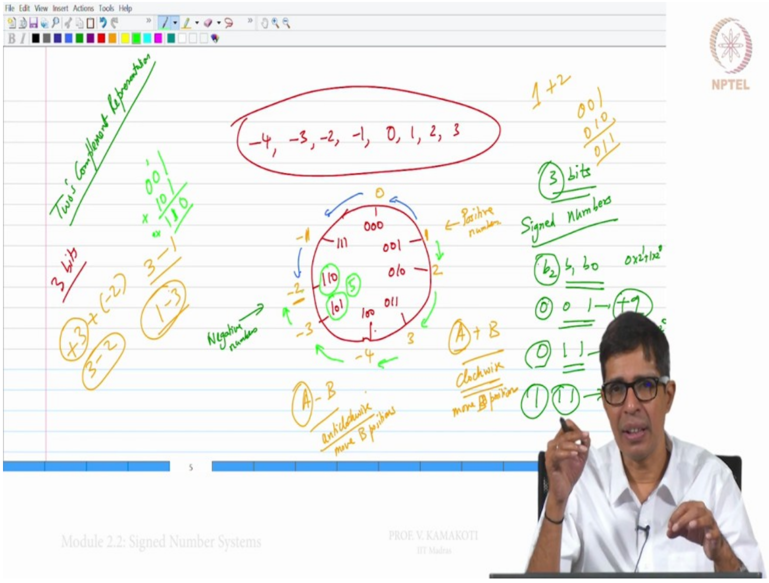
(Refer Slide Time: 0:49)




So, welcome to module 2.2. In this module we will look at signed Number representation, or what we call as binary, signed binary number representation. So, how do you represent a signed integer? Say +2, +215 -3216, so we have something called a sign prefixed. So, the logical way of communicating to your computer the sign of a number is to have the most significant bit. Here also we add the most significant position, right, + 215 which is the most significant digit, 2 is the most significant digit.

And we add the + after the most significant digit. Similarly -3216, 3 is the most significant digit and you add the - just before the most significant digit. So, the best way to tell a computer that an integer is negative or positive is to use the most significant bit for this purpose. If the bit is one, then it is negative, if the most significant bit is 0, I will call it MSB, if the most significant bit is 0, then the thing is positive. If the most significant bit is 1, then it is negative, it is 0 then it is positive.

(Refer Slide Time: 2:42)




So this is how basically we work and this is now let us go and see how we are going to do a signed number representation. Now, let us just see this, let me say that we have 0, 1, 2, 3, -1, -2, -3. Suppose I say I have 3 bits, suppose I say I have 3 bits to represent any number and suppose I want to represent signed numbers. That means as mentioned in the previous slide, previous page, the 1st bit will be used for sign. So remaining there will be 2 bits, so let me call it bit 2, bit 1 and bit 0.

So bit 2 will represent sign and bit 1 and bit 0 will separate the value. So, if I say 001, 0 means +, this 01 is nothing but 0 into 2 power 1 +1 into 2 power 0. So this will represent +1. Suppose I say 011, 0 means +, this 11 will be 1 into 2 power 1 +1 into 2 power 0 which is 2 +1, 3, so this will represent +3, this will represent +1 and so on. Suppose I put 111, then this will represent, this can represent in some sense this can represent -3 also, right.

So, now we will now look at what we call as the, we will call this as 2's complement representation. You would have learned this in your introduction to programming course but I am quickly giving you a recap of this 2's complement representation. Using this 2's complement representation, let us take 3 bits for example, we can represent this range, that is -4 to +3. And let me tell you how this is going to happen.

So, let us see 3 bits, all combinations I am writing 000, 001, 010, 011, 100, 101, 110, 111. And note that by our theory of these 4 fellows, should become, should represent negative numbers because of what, their 1st digit is 1, the most significant bit here is. While these fellows should represent positive numbers, why, because the most significant digit in this case is a 0 which is positive. So, I can represent 4 positive numbers and 4 negative numbers, the total 8 combinations I have.

Okay. So now let us name this. So this is a positive is very easy, so this is 0, this is 1, this is 2, this is 3, right. Now, let us start, so let us, so this is a wheel, you start from 000 to 111 and again back. So, let us please, so let us model addition and. If I have signed numbers, then addition can essentially become subtraction. For example suppose I say +3+ -2 is nothing but 3-2 which is subtraction, right. So the moment I have signed numbers, adding 2 signed numbers of opposite signs essentially implies subtraction.

Here I am adding +3 to -2, so essentially adding 2 numbers of opposite signs, it essentially boils down to subtraction, same thing can happen here. So the way we are doing this is, this is 0, 1, 2, 3, but then we will go and make this as -1, -2, -3 and -4, right. Now the good thing here is that if I want to add some number A with B, I go to A and move B positions clockwise. Okay. So when I want to subtract say B from A, I go to the position of A and move B positions anticlockwise, move B positions anticlockwise from A, move B positions clockwise from A, clockwise from A.

So this is how we basically look at this. So for example let me say I want to add 1+2. So I go to 1 and add 2 positions, so I get 3, so 001+010 if I add I get 0 +1 is 1, 1+, sorry, 001+0, 1+2, with 001+010, 1+0 is 1, 1+0 is 1, 101, I get this, right. Look at the other way, suppose I want to subtract 3-1, I go to 3 and move anticlockwise one step, right and I get 2. So your subtraction becomes anticlockwise, your addition becomes clockwise.

So let us say I want to subtract 1-3, I go to 1 and move 3 steps anticlockwise, 1, 2, 3, I get -2. Right, so 1 -3 is -2 and so that is what you get here. Okay, so this circular representation of the number line, right is basically helps you visualize addition and subtraction as clockwise in anticlockwise movements respectively. Now what is interesting here is, let us see this. Suppose I have 1-3, right, so I am at 1 and I am moving 3 steps backward, right. So I am moving these steps backward 1, 2, 3.

So on a circle of size 8, moving 3 steps anticlockwise is equal to moving 5 steps clockwise, right. Moving 3 steps anticlockwise is equal to moving 5 steps clockwise. Let us see what happens if I move 5 steps clockwise, I get back to the same position. Right. Now you understand, so instead of subtract, instead of opening moving anticlockwise, I can move 5 steps clockwise and I get the same. Now, what is, what is the binary representation of -3, 101 and what is 101 in decimal, it is 5, right.

So if one is represented as 001 and -3 is represented as 101, right, now adding these 3, adding these two, just add these two because from 1 subtracting 3 is moving 3 steps anticlockwise, that is equivalent to moving 5 steps clockwise. And what is clockwise here, clockwise means addition. So instead of subtracting 3 like this, I add 5 here, and so that is what I am trying to do. So 001+101 will give me 1+1 is 0 with carry 1, so I get 110. And what is 110, 110 in our representation is -2, right.

So what has happened here is the subtraction has actually become addition. So if the numbers are represented in 2's complement notation as we have seen here, then what will essentially happen is your, there is a thing called subtraction, everything is addition and you will get an answer accordingly. So, the demonstration we have done here, if I want to subtract 1 from 3, I add 5 to 1, to basically get 110. And that 110, the representation for that is -2. So, if I store the integers in this format, what we call is the 2's complement format, then basically you did not have a separate subtract, everything can basically land up as addition. Right.

So this is a very quick introduction to 2's complement arithmetic, we will see how we can generate this 2's complement arithmetic in the next module. Nevertheless, I suggest that if you are if you need to, if you have not studied 2's complement or you forgot what you are studied, what you have learnt in your introduction to programming course, First-year course this should be taught in any curriculum, then I strongly suggest that you go to any Wikipedia page, there are many many introductions to 2's complement, just have a look, a five-minute read will basically brushed you up in addition to what I have covered here.

We will now look, go to the next module where we will understand, given a number, how to generate its 2's complement, we will see from an hardware perspective, thank you.