

**Digital Circuits**  
**Prof. Santanu Chattopadhyay**  
**Department of Electronics and Electrical Communication Engineering**  
**Indian Institute of Technology, Kharagpur**



**Lecture – 06**  
**Number System (Contd.)**

So, in our last class we were looking into negative numbers and we have seen that they can be represented either in sign magnitude format or 1's complement format or 2's complement format. Now, how to get back the number like once it is a negative number and represented in 1's complement, how do we know what is the number.

(Refer Slide Time: 00:38)

**Recovery of the Numbers**

1's Complement	2's Complement
Let $f(x) = 2^n - 1 - x$	Let $g(x) = 2^n - x$
Theorem: $f(f(x)) = x$	Theorem: $g(g(x)) = x$
Proof: $f(f(x))$	Proof: $g(g(x))$
$= f(2^n - 1 - x)$	$= g(2^n - x)$
$= 2^n - 1 - (2^n - 1 - x)$	$= 2^n - (2^n - x)$
$= x$	$= x$

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

So, it is like this for 1's complement system so, you can say it is so, we define the function  $f(x)$  to be  $2^n - 1 - x$ , then  $f(f(x))$  is  $x$ . So, that is if we apply this particular function on itself ok so, you will get back the  $x$ . So, why because if we are trying to find this  $f(f(x))$ , then  $f(x)$  is  $2^n - 1 - x$ . So, we put it here.

Now,  $f(2^n - 1 - x)$  is equal to  $2^n - 1 - (2^n - 1 - x)$ . So, this  $2^n$  and  $1$ , they will cancel out what remains is the  $x$ . So, if you so, this is the 1's complement representation of the number  $-x$ . So, from there if you want to get back what is the original  $x$  so, you can apply the function on it and get back  $x$ . So, if you

apply on this f, if you apply on the a apply the function f on this input. So, you will get back the original number ok.

So, whose negative is actually represented by this representation. Similarly in the 2's complement case also suppose so, this is the formula for getting the 2's complement of a number. So,  $g(x) = 2^n - x$ . So, we have we have a similar theorem, which says the  $g(g(x))$  is equal to  $x$ , that is if we apply the function  $g$  on the result  $g(x)$ . So, you will get back  $x$ .

So, as a proof we can see that  $g(g(x))$  is equal to  $g(2^n - x)$ . So, you put it here. So,  $g(2^n - x)$  and by this logic that  $g(x)$  is  $2^n - x$ . So, we can write the  $g(2^n - x)$  equal to  $2^n - (2^n - x)$ ; again  $2^n - 2^n + x$  because this  $x$  in the at the place of this  $x$ , we have to put  $2^n - x$ . So, we put it here. So, this you this way we get this one so, you get back the  $x$ .

That means both in 1's complement and 2's complement notation, if you want to get back the original number whose negative was represented as  $2^n - 1 - x$  or  $2^n - x$  in 1's complement or 2's complement, to get back  $x$ . So, you take the 1's complement or 2's complement of the number. So, you will get back the original number  $x$ . So, it is helpful for our understanding ok. So, if you are getting a negative number. So, what was the original positive number. So, we can get back by applying the function again.

(Refer Slide Time: 03:17)

Floating point numbers

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

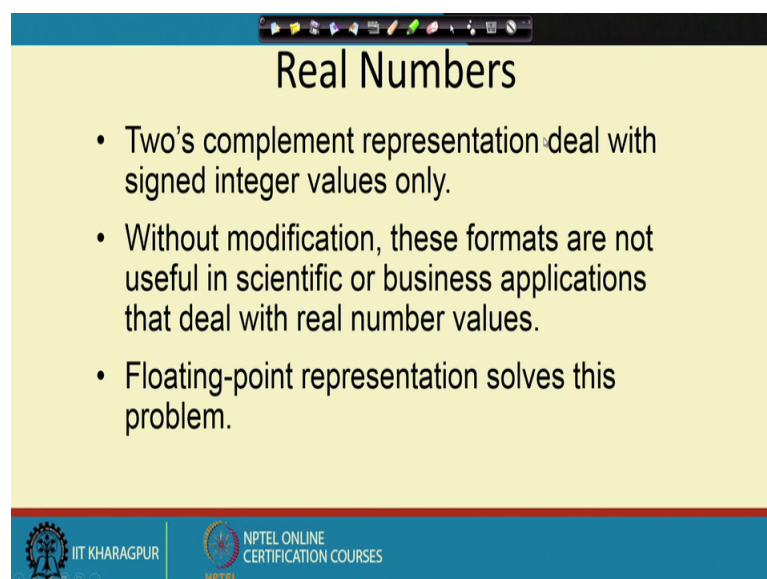
So, next we will see the representation for floating point numbers. So, far we have considered only integers positive and negative, and we have also consider fractions like say we have seen that this numbers where the number of digits that we are using for representing it is fixed. So, like we can say we can we have representation like say 15.53. So, we have seen that this 15 part, we are converting it into some integers some integer part then this point is there and then this 53 also we are convert into binary to represent it.

So, that way this representation of a problem comes when this number of bits that are allocated for this part and number of bits that are allocated for this part they are same they are fixed ok. So, you cannot change them. So, in so, like if I say the so, for example, for representing 15, you will need at least 4 bits and for a representing 5 3 properly. So, you will be again requiring say suppose I give you say 5 bits. So, 5 plus 4 total 9 bits are available; but if 9 bits are needed, but in a reality so, if we give you say only say 5 bits.

Then for the sake of accuracy how do you choose like how many bits we can give for this integer part and how many bits we can give for the fractional part. So, if you are using this real number representation in the form of fixed point notation. So, there are problems in terms of accuracy. So, we will see that we can do better by realizing them in the floating point form.

So, we will be looking into this floating point representation in the in the next few slides.

(Refer Slide Time: 05:17)



**Real Numbers**

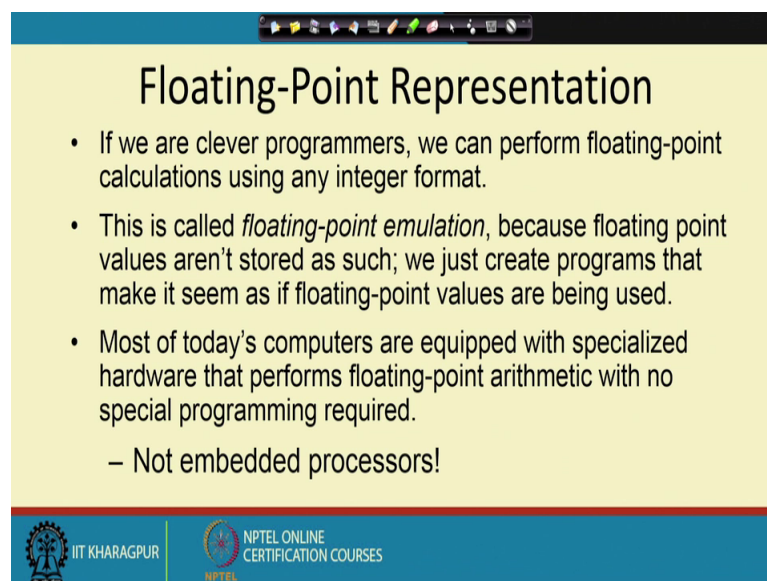
- Two's complement representation deal with signed integer values only.
- Without modification, these formats are not useful in scientific or business applications that deal with real number values.
- Floating-point representation solves this problem.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this real numbers so, two's complement numbers that we have discussed so, far they this they deal only with signed integer values and without modification these formats are not suitable in scientific or business applications that deal with a real number values because of the accuracy and all.

So, floating point representation, they will solve this problem. So, we can we will see that this floating point representation so, they can help us in having a better accuracy and all.

(Refer Slide Time: 05:43)



The slide is titled "Floating-Point Representation" and contains the following text:

- If we are clever programmers, we can perform floating-point calculations using any integer format.
- This is called *floating-point emulation*, because floating point values aren't stored as such; we just create programs that make it seem as if floating-point values are being used.
- Most of today's computers are equipped with specialized hardware that performs floating-point arithmetic with no special programming required.
  - Not embedded processors!

At the bottom of the slide, there are two logos: IIT KHARAGPUR on the left and NPTEL ONLINE CERTIFICATION COURSES on the right.

So, if in fact, before going into this discussion on this floating point representation one thing we should keep in mind that ultimately this floating point is a representation is nothing, but collection of a some integers. It is a every floating point number is represented by means of a collection of integers.

So, if a if in the programming we can use some tricks by which we can do this floating point operations ourselves. So, if underlying a machine or say operating system, it is not supporting that floating point representation also. So, as a programmer so, you can always do that and you can use the integer formats for representing different components of the floating point number and represent it that way.

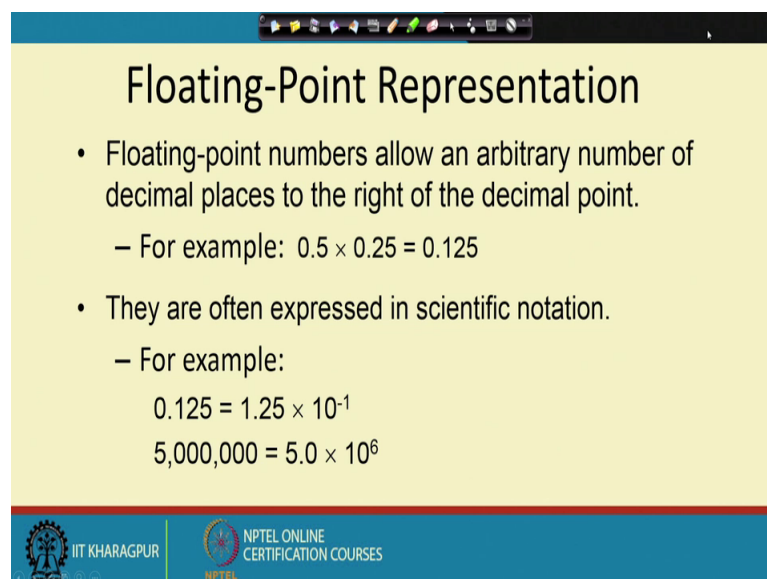
So, this is known as the floating point emulation because the so, because the floating point number is not stored as such. So, we just create programs that so, that it seems like

we as if we are handling floating point numbers. But most of the computers that we have today or the processors the advanced processors, they are equipped with specialized hardware that can do this floating point arithmetic without any special programming requirement.

So, this is very important, because otherwise whenever you are you whenever you represent this floating point operation by means of software, the overall operation will become will become slower, because software is always much slower compared to the hardware.

So, if the hardware directly supports the floating point format. So, it is better. So, most of the processors that we have now the advanced processors so, they support these floating point formats in the hardware itself. However, these are not embedded processors because these are not dedicated for the floating point operation only. So, they do many other things, but they also support the floating point operation.

(Refer Slide Time: 07:37)



**Floating-Point Representation**

- Floating-point numbers allow an arbitrary number of decimal places to the right of the decimal point.
  - For example:  $0.5 \times 0.25 = 0.125$
- They are often expressed in scientific notation.
  - For example:  
 $0.125 = 1.25 \times 10^{-1}$   
 $5,000,000 = 5.0 \times 10^6$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how do we represent a floating point? So, floating point numbers they allow an arbitrary number of decimal places to the right of the decimal points.

For example so, floating point means so, we can have any number of digits after this decimal point. So, that is the floating point; obviously, for actual representation so, it cannot be arbitrary. So, there has to be some higher limit on the number of digits that you

have after the decimal point, because a computer has got any processor or computer it has got a finite word size. So, we cannot go beyond that anyway. So, if you are multiply 0.5 by 0.25 you get 0.125.

So, you see the problem the point that is the this point the that this bullet tells is that these numbers, 0.5 and 0.25, they have 1 digit after decimal here and 2 digits after decimal here and in the result I have got 3 digits after the decimal. So, after decimal how many digits will come. So, that is not fixed. So, that is why it is a floating point. So, in a scientific notation so, we can represent it like this, the say this 0.125 that we have got so, it can be represented as 1.25 into 10 power minus 1, say this 5 they are 6 0's after 5 is a very large number.

So, you can represent it as 5.0 into 10 to the power 6. So, you see that if we are representing it in this format, then the numbers they have got two components. The first component is a number and the second part is a power of 10 ok. So, that way we can have different levels of we can the range of the numbers that we are going to store. So, it becomes larger because we are we are converting it into this format.

(Refer Slide Time: 09:26)

The slide is titled "Floating-Point Representation" and contains the following text:

- Computers use a form of scientific notation for floating-point representation
- Numbers written in scientific notation have three components:

A diagram below the text shows the components of scientific notation: "Sign" (a green circle with a plus sign), "Mantissa" (a pink box containing "1.25"), and "Exponent" (a yellow box containing "10<sup>-1</sup>"). Arrows point from the labels to their respective parts in the expression  $+ 1.25 \times 10^{-1}$ .

At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL ONLINE CERTIFICATION COURSES. A small video inset of a speaker is visible in the bottom right corner.

So, computers they use this, but this type of scientific notation for floating point representation. So, numbers written in scientific notation they have got three components; one is the sign another is called mantissa, another is called exponent. So,

the sign is whether the number is positive or negative that is all right, then the mantissa part.

So, mantissa we will have a particular format like in this case so, we are assuming that we have got 1 digit before the decimal point so, only 1 digit before the decimal point. So, this is a 10 power minus 1 so, that is the exponent part. So, I need to remember this sign bit this 1.25 and this minus 1 to know what is the number that I am going to store. So, they are called sign mantissa exponent representation.

(Refer Slide Time: 10:17)

**Floating-Point Representation**

- Computer representation of a floating-point number consists of three fixed-size fields:

Diagram illustrating the fields:

- Sign (indicated by a blue oval and arrow pointing to a small green box)
- Exponent (indicated by a yellow box)
- Significand (indicated by a pink box)

- This is the standard arrangement of these fields.

*Note: Although "significant" and "mantissa" do not technically mean the same thing, many people use these terms interchangeably. We use the term "significant" to refer to the fractional part of a floating point number.*

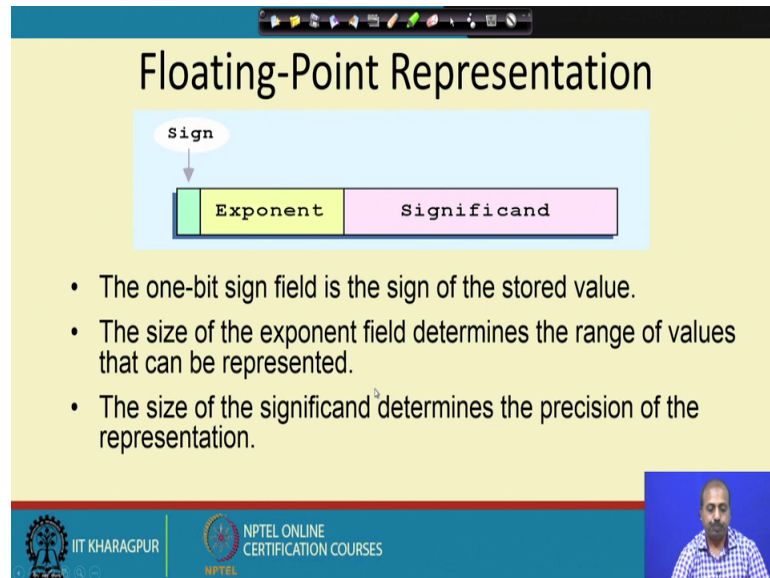
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Computer representation of floating point numbers so, they consists three fixed size field. The sign field is definitely there so, that is the that is a sign then the exponent part that is the power of 10 that we are talking about so, the exponent part and the significant part.

So, significant part is the in the previous example where you when you are talking about mantissa. So, that is actually told as significant here. So, exponent is same. So, this is a very standard arrangement of this field that is the after sign the exponent will come and then the significant will come. So, depending upon the number of bits that you allocate to different portions, the sign will definitely be 1 bit, but this exponent and significant. So, you can assign different number of bits to tell like how many up to what power we can represent or up to how many digits after decimal point so, we can represent in the significant form.

So, although this significant and mantissa they do not technically mean the same thing mean. So, they can be used interchangeably ok. So, we will see why they are not same.

(Refer Slide Time: 11:25)



The slide is titled "Floating-Point Representation". It features a diagram of a floating-point format. At the top, a light blue box contains the word "Sign" with a downward arrow pointing to a small green box. Below this, a larger box is divided into two sections: a yellow section labeled "Exponent" and a pink section labeled "Significand". Below the diagram, there are three bullet points:

- The one-bit sign field is the sign of the stored value.
- The size of the exponent field determines the range of values that can be represented.
- The size of the significand determines the precision of the representation.

At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. A small video inset of a man is visible in the bottom right corner.

So, the one-bit field is the sign value that is stored here, the size of the exponent field it is determines the range of values that can be represented, as I have already said. The size of the significant it determines the precision of the representation.

So, if I say that this significant is 32 bit, then after decimal point we are going to represent 32 up to 32 bit we are taking the precision. So, if you if you if more number of bits are available for the significant part. So, I will have better precision and less number of bits means, I will have less precision. So, that way this exponent mantissa and significant parts are there to this term.



(Refer Slide Time: 12:08)

**Floating-Point Representation**

sign

Exponent      Significand

- We introduce a hypothetical “Simple Model” to explain the concepts
- In this model:
  - A floating-point number is 14 bits in length
  - The exponent field is 5 bits
  - The significand field is 8 bits

IIT KHARAGPUR      NPTEL ONLINE CERTIFICATION COURSES

So, we will take a very simplistic example ok. So, for our discussion so, a hypothetical simple model to explain the concepts. So, we will think that the floating point number altogether I am giving it 14 bits ok. So, this entire from this entire structure that I have so, that is given 14 bits. So, out of that sign will take 1 bit.

So, we are left with 13 bit out of the 13. So, exponent part we are giving 5 bits and the significant part we are giving 8 bits. So, this is the this is our simplistic model a 14 bit representation of floating point numbers.

(Refer Slide Time: 12:49)

**Floating-Point Representation**

sign

Exponent      Significand

$5.52 \times 10^2$   
 $\downarrow$   
 $0.552 \times 10^4$

- The significand is always preceded by an implied binary point.
- Thus, the significand always contains a fractional binary value.
- The exponent indicates the power of 2 by which the significand is multiplied.

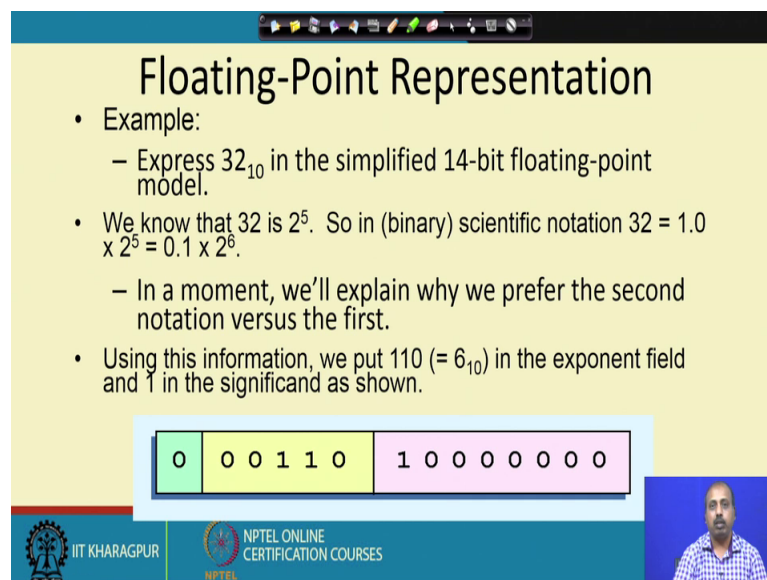
IIT KHARAGPUR      NPTEL ONLINE CERTIFICATION COURSES

So, what can we do with this? The significant is always preceded by an implied binary point. So, it is suppose I am going to represent a number say a 5.5 into 5.52 into 10 power say 3.

So, what we will do. So, we will not represent it like this. So, we will take it as 0.552 into 10 to the power of 4 ok. That that the, that is why the significant is always preceded by the implied binary. So, binary point is always at this at this point. So, at the very first one is the binary point ok. So, we do not need to store it explicitly, meaning that it starts with the binary point. So, significant always contains a fractional binary value like here it is 0.552 as I was talking about.

The exponent part so, it is a power of 2. So, instead of since I am representing in a computer. So, instead of 10 power. So, I will be calling it 2 power ok. So, it is a power of 2. So, it is. So, it is 5.52 into 2 to the power something. So, 10 to the power 3 means 1000. So, it is may be 2 power 10 is 1024. So, it will be close to that ok. So, that way I can so, I that the exponent part is power of 2 instead of power of 10 that we are that we are looking into.

(Refer Slide Time: 14:18)



**Floating-Point Representation**

- Example:
  - Express  $32_{10}$  in the simplified 14-bit floating-point model.
- We know that 32 is  $2^5$ . So in (binary) scientific notation  $32 = 1.0 \times 2^5 = 0.1 \times 2^6$ .
  - In a moment, we'll explain why we prefer the second notation versus the first.
- Using this information, we put 110 ( $= 6_{10}$ ) in the exponent field and 1 in the significand as shown.

0 00110 10000000

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, suppose I am trying to represent the number 32 in decimal system into this 14 bit floating point notation that we have into introduced. So, 32 is 2 to the power 5 so, in binary notation by scientific notations. So, 32 will be 1.0 into 2 to the power 5. So, scientific notation means, before decimal point we have got 1 digit and in case of this

floating point representation that is followed in the computer system. So, we have got a decimal point it starts with the decimal point.

So, this is 32 in scientific notation is  $1.0 \times 2^5$ , in our notation it is  $0.1 \times 2^6$  ok. So, why do we prefer the second one? So, we will see slightly later. =So, this 6 ok so, 6 is 110. So, in the exponent part we put 110 and total 5 bits are allocated to the exponent part out also in this in this 5 bit location, we stored the value 6110, and then on the significant part.

So, we have to keep only this 1 that we have so, because this point is implied here. So, we have to I have to keep this 1 and 1 is represented like this. So, this is so, 1 0 00 0 like that. So, it is 1 0 0 0 0. So, it will be  $0.1 \times 2^6$ . So, that represent again 32. So, so this is. So, we put 1 1 0 in the exponent part and 1 in the significant part as we have shown here.

(Refer Slide Time: 16:03)

**Floating-Point Representation**

- The illustrations shown at the right are *all* equivalent representations for 32 using our simplified model.
- Not only do these synonymous representations waste space, but they can also cause confusion.

0	00110	10000000
0	00111	01000000
0	01000	00100000
0	01001	00010000

IIT KHARAGPUR
 NPTEL ONLINE CERTIFICATION COURSES

So, these are all equivalent like say the. So, this is the representation they are all representing 32. It is  $32 \times 2^0$  it is  $1 \times 2^6$ . So, it is  $1 \times 2^6$ . So, this is point 0 1. So, this is  $0.01 \times 2^7$  then it is  $0.001 \times 2^8$ . So, that way these multiplying by 2 will be shifting it by 1 position in the binary representation.

So, that way all these representations, they have the same value which is 32. So, they are they are synonymous ok. So, they, but they can also cause some confusion. So, we need to have some standardization, because otherwise what will happen is that different computers may be storing information in different format. So, as a result understanding may become confusing ok. So, we have to use some fixed notation.

(Refer Slide Time: 17:06)

**Floating-Point Representation**

Sign

Exponent      Significand

- Another problem with our system is that we have made no allowances for negative exponents. We have no way to express  $0.5 (=2^{-1})!$  (Notice that there is no sign in the exponent field.)

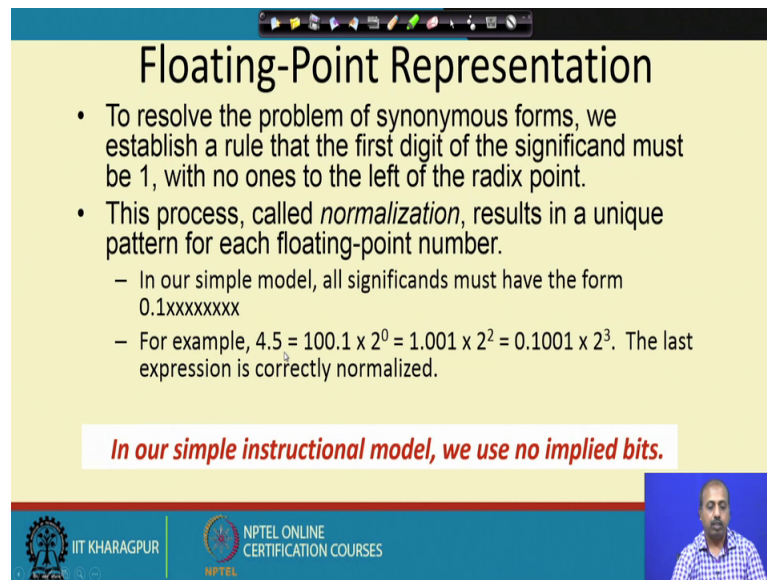
**All of these problems can be fixed with no changes to our basic model.**

IIT KHARAGPUR      NPTEL ONLINE CERTIFICATION COURSES

So, another problem with the system is that we have no we have no allowance we have for the negative exponent. So, we have got this sign bit, but that is for the entire number, but suppose I have to represent say 2 power minus 3. So, I do not have any of the any technique by which I can represent minus 3.

So, we have no way to express this 0.5. So, that is 2 power minus 1 why? Because this I cannot have I cannot tell that my exponent part is minus 1. So, I can take tell that exponent part is 1, but I cannot tell it to be minus 1, there is no sign bit. So, this problems can be fixed with no changes to our basic model. So, we can keep this exponent both positive and negative.

(Refer Slide Time: 17:53)




**Floating-Point Representation**

- To resolve the problem of synonymous forms, we establish a rule that the first digit of the significand must be 1, with no ones to the left of the radix point.
- This process, called *normalization*, results in a unique pattern for each floating-point number.
  - In our simple model, all significands must have the form 0.1xxxxxxx
  - For example,  $4.5 = 100.1 \times 2^0 = 1.001 \times 2^2 = 0.1001 \times 2^3$ . The last expression is correctly normalized.

*In our simple instructional model, we use no implied bits.*

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, how do we do this? We will see slowly. So, to resolve this problem of synonymous forms that is for the same number we have got multiple representation so, we have to have a rule ok.

So, some rule has to be followed and the rule is that, the first digit of the significant must be 1 with no ones to the left of the radix point because it is a binary representation. So, it is zeros and ones. So, I will say that my significant can never start with a 0. So, if you just look into this slide. So, you see that this representation we are telling ok, but these three representations we are telling that they are not correct because my as per my convention. So, this significant part must start with a 1. So, it cannot start with a 0 ok.

So, that way we put we make a rule, that the first digit of the significant must be 1 and to the left of it to the and there will be no 1 to the left of the radix point. So, before radix point there is no digit or no bit, but after radix point we have always it always starts with a 1. So, this is called normalization ok. So, this process is called normalization like say 4.5.

So, 4.5 when you are representing so, this is 100.1 into 2 to the power 0. So, we can tell it like this. So, it can also be like 1.001 into 2 to the power 2, it can also be 0.1101 into 2 to the power 3. So, we will say that the this representation is correct because it does not have any one to the left of the radix point, to the left of the radix point there is only 0.

Similarly, here to the left of the radix point we have 1. So, this representation is not correct, but this is correct and in the right side. So, its it starts with a 1. So, the in the here if you do. So, it is right side this starts with 0. So, it is not correct. So, here it is starting with a 1. Here also its starts the right side starts with a 1, but on the left side we have got some digits ok. So, that is not correct.

So, there should not be any digit to the left of the radix point. So, the only this representation is correct in as per our convention. So, all significands must have the form 0.1 xx xx like that. So, its starts it must start with it the bit before decimal point should be 0, and after decimal point the first bit should be 1. Accordingly we have to adjust the exponent part ok. So, we can adjust the exponent part and that way we can do it.

So, in this model so, we have we use no implied bits ok. So, but it is it is 0 directly, but the implied bits we will see later. So, this is known as the process of normalization.

(Refer Slide Time: 20:46)

### Floating-Point Representation

- To provide for negative exponents, we will use a *biased exponent*.
- A bias is a number that is approximately midway in the range of values expressible by the exponent. We subtract the bias from the value in the exponent to determine its true value.
  - In our case, we have a 5-bit exponent. We will use 16 for our bias. This is called *excess-16* representation.
- In our model, exponent values less than 16 are negative, representing fractional numbers.

5  $\rightarrow$  32 pattern  
 $\downarrow$   
 16  
 $\downarrow$  +16  
 16  
 +++++

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, how do we go for this negative exponents ok? So, negative for the for the purpose of negative exponent. So, the technique that is used is known as biased exponent. So, a bias is a number that is approximately midway in the range of values expressible by the exponent, and we subtract the bias from the value in the exponent to determine its true value.

So, we have got in our example we have got 5 bit exponent. So, 5 bit exponent means. So, it can go from. So, if I if I am using negative numbers also. So, it can go from a minus 16 to plus 16. So, 5 bit is all the bits may be 1 ok. So, it may be 1. So, by so, I may be so, it is minus 16 to plus 16. So, this out of this total the in 5 bits. So, I can have 32 different patterns 32 patterns. So, diff 32 different exponents can be stored.

So, I can make it I can I can divide this range on the negative side minus 16 and the positive side say plus 16 and then what we do? We um with the value with we stores add another 16 ok. So, we add another 16 what whatever be the result that is actual actually stored. So, this is called excess 16 representation. So, we will take an example that will make it clear.

(Refer Slide Time: 22:29)

**Example 1**

- Example:
  - Express  $32_{10}$  in the revised 14-bit floating-point mode
- We know that  $32 = 1.0 \times 2^5 = 0.1 \times 2^6$ .
- To use our excess 16 biased exponent, we add 16 to 6, giving  $22_{10}$  ( $=10110_2$ ).
- So we have:
  - $00000 \Rightarrow 0 \rightarrow 31$
  - $11111$

*5 → 32 patterns*  
*↓ +16*  
*↓*  
*16*

*11111*

0	1 0 1 1 0	1 0 0 0 0 0 0 0
---	-----------	-----------------

IIT KHARAGPUR    NPTEL ONLINE CERTIFICATION COURSES

See this one says we are again coming back to example of 32. So, 32 representation we have seen that we are following this notation  $0.1$  into  $2$  to the power  $6$ . So, in our previous case the a significant part was a storing this  $1$  and the exponent part was storing this  $6$ , but we do not store  $6$  there. So, what we do with this  $6$  we add  $16$ . So, that way it becomes  $22$ . So, we will be storing  $22$  in the exponent.

So, here if you look into the all possible bit pattern so, you can if you look into sorry if you look into all possible bit patterns here, then it is all  $0$  to all  $1$  so; that means, I can go from  $0$  to  $31$ . So, in that range what I do. So, I add  $16$ . So, if I add  $16$ , then this will become this  $6$  will become  $16$  plus  $6$   $22$ . So, I stored the pattern of  $22$  here knowing that

when we are calculating the value. So, this is in this is. So, 16 extra has been added. So, I subtract 16 from here and I will get back 6 ok.

So, if I have a negative number if I have a negative number. So, the negative can go till say minus minus 16. So, that way that even 16 will be added to that. So, that number will become 0 the exponent will become 0. So, this way we do not have to store the negative exponents separately. So, in the following this excess notation so, this mid value will be added to the exponent and that will be stored. So, the negative numbers will also come as positive numbers only.

So, in the storage so, that is known as the biased exponent concept; so, exponent has been biased by the middle of the range. So, that everything becomes positive.

(Refer Slide Time: 24:35)

**Example 2**

- Example:
  - Express  $0.0625_{10}$  in the revised 14-bit floating-point model.
  - We know that  $0.0625$  is  $2^{-4}$ . So in (binary) scientific notation  $0.0625 = 1.0 \times 2^{-4} = 0.1 \times 2^{-3}$ .
  - To use our excess 16 biased exponent, we add 16 to -3, giving  $13_{10}$  ( $=01101_2$ ).

0	0 1 1 0 1	1 0 0 0 0 0 0 0
---	-----------	-----------------

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, next we have another example suppose it is 0.0625 in the binary in the in the decimal representation. So, this 0.0625 is 2 power minus 4 because. So, in so, this is 2 power minus 4. So, in the scientific notation that we are having so, this will be represented as 1.0 into 2 power minus 4 or in our notation. So, it is as per our rule. So, it is there should be only 1 digit after there should be no digit before decimal. So, this is 0.1 into 2 power minus 3.

So, this is the value that we are trying to store finally, in the computer for example, now what we do we take an excess 16 notation. So, with this minus 3, 16 is added. So, the



value becomes 13. So, ultimately in the exponent part we store 13 we do not store minus 3, but we store 13 knowing fully well that when we are looking back for the value. So, from this whatever is stored there. So, I have to subtract 16. So, it was excess 16. So, 16 was added while storing it. So, while getting it back. So, I have to subtract 16 and then only I can get it back ok.

So, that way I can do we I can go back to 2 power minus 3 and this part this a significant part remains unaltered. So, this is known as the biased exponent concept.

(Refer Slide Time: 25:58)

**Example 3**

- Example:
  - Express  $-26.625_{10}$  in the revised 14-bit floating-point model.
- We find  $26.625_{10} = 11010.101_2$ . Normalizing, we have:  $26.625_{10} = 0.11010101 \times 2^5$ .
- To use our excess 16 biased exponent, we add 16 to 5, giving  $21_{10} (=10101_2)$ . We also need a 1 in the sign bit.

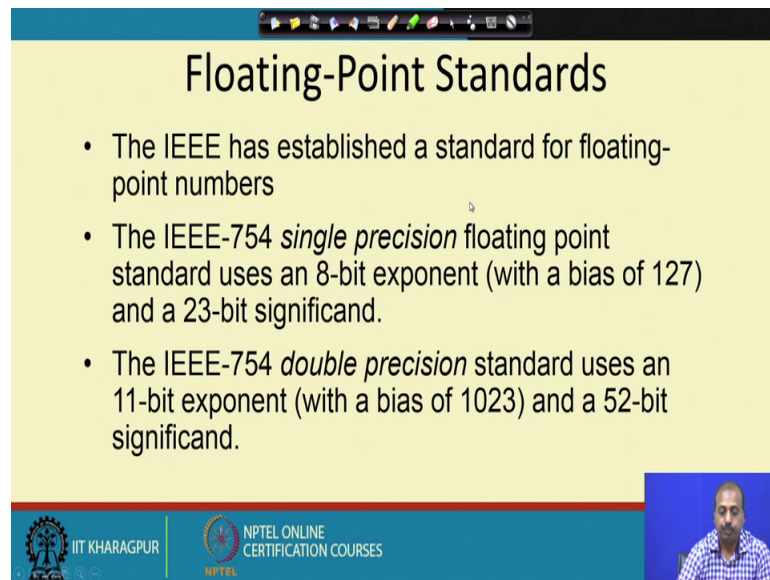
1	1 0 1 0 1	1 1 0 1 0 1 0 1
---	-----------	-----------------

IIT KHARAGPUR     NPTEL ONLINE CERTIFICATION COURSES

Another example so, suppose and to we represent the number minus 26.625 in this 14 bit representation. So, since this is minus. So, this sign bit should be 1 now this 26.625. So, if you represent it in binary notation. So, it is like this 11010.101. So, we do normalization. So, after doing normalization so, it is like this. So, this is the format we want to store in the computer, but. So, this significant part is storing it 1 1 0 1 1 0 1, but for the exponent part for the exponent part. So, we take again excess 16.

So, with this 5 16 will be added. So, this becomes 21. So, this 21 is stored in the exponent part. So, this way we can store the numbers in the system.

(Refer Slide Time: 26:51)



The slide is titled "Floating-Point Standards" and contains the following text:

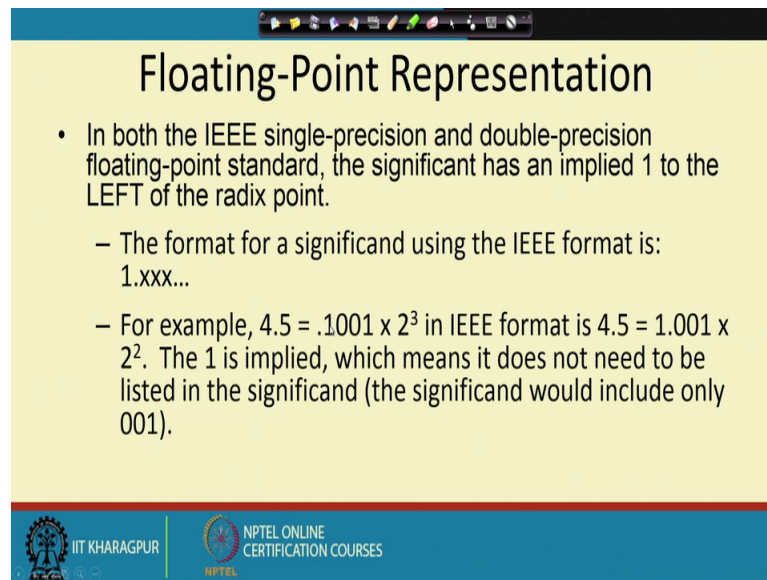
- The IEEE has established a standard for floating-point numbers
- The IEEE-754 *single precision* floating point standard uses an 8-bit exponent (with a bias of 127) and a 23-bit significand.
- The IEEE-754 *double precision* standard uses an 11-bit exponent (with a bias of 1023) and a 52-bit significand.

The slide footer includes the IIT Kharagpur logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset in the bottom right corner shows a man speaking.

So, now so, this is the standard that we were we have developed the model that we have the 14 bit representation of model that we were using. So, IEEE has got some standards for this floating point numbers. So, there it is IEEE 754 standard and they have got one standard for single precision number and another standard for double precision number.



So, for 60 for a single precision number so, it uses 8 bit exponent with a bias value of 127 and a 23 bit significant; On the other hand this a double precision number, it has got eleven bit exponent with a bias of 1023 and 52 bit significant. So, that is a very very huge number. So, that way we can represent very large numbers in double precision format with higher precision values.

(Refer Slide Time: 27:42)



## Floating-Point Representation

- In both the IEEE single-precision and double-precision floating-point standard, the significant has an implied 1 to the LEFT of the radix point.
  - The format for a significand using the IEEE format is:  
1.xxx...
  - For example,  $4.5 = 1.001 \times 2^2$  in IEEE format is  $4.5 = 1.001 \times 2^2$ . The 1 is implied, which means it does not need to be listed in the significand (the significand would include only 001).

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

So, both single precision and double precision use the significant has an implied 1 to the left of the radix point. So, in our model to the left of the radix point we have only 0, but in case of this float this IEEE standard so, of to the left we have got a 1. For example, this 4.5 is actually this one, in our notation it will be like this, but in IEEE format so, it will be 1.001 into 10 to power 2 because that will be so, the this 1 is implied. I do not need to store this 1 explicitly I in the in the storage I will only have this 001 in the in the significant part, but this 1 is implied.


So that means, so, in our case what was happening is this 1 we were storing separately and, but that way we were actually wasting a bit ok. So, we were wasting 1 bit of storage, but using this 1.001 or 2 power 2. So, we can we so, that we can say 1 bit ok. So, that way we can have a better representation.

(Refer Slide Time: 28:53)

## Floating-Point Representation

- Example: Express -3.75 as a floating point number using IEEE single precision.
- First, let's normalize according to IEEE rules:
  - $-3.75 = -11.11_2 = -1.111 \times 2^1$
  - The bias is 127, so we add  $127 + 1 = 128$  (this is our exponent)
  - The first 1 in the significand is implied, so we have:

(implied)  
Since we have an implied 1 in the significand, this equates to  
 $-(1).111_2 \times 2^{(128-127)} = -1.111_2 \times 2^1 = -1.111_2 = -3.75$ .




So, this for example, this minus 3.75 so, this floating point representation so, it will be like this. So, minus 3.75 is minus 11.11. So, minus 1.111 into 2 power 1.

Now, this bias is 127. So, 127 is added to 1 so, it becomes 128. Now the sign is negative. So, the so, this is the sign now, this part is so, before decimal. So, this is this is the exponent. So, 127 and then we have got this implied 1 and after that we have got this 1 1 1 0 0 0. So, this 1 is already implied. So, this 1 this 1 need not be stored separately.

(Refer Slide Time: 29:35)

## FP Ranges

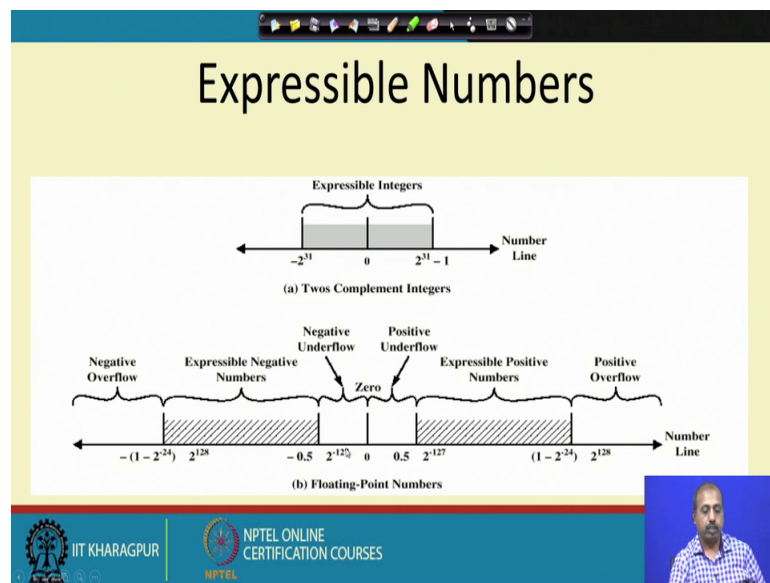
- For a 32 bit number
  - 8 bit exponent
  - $\pm 2^{256} \approx 1.5 \times 10^{77}$
- Accuracy
  - The effect of changing lsb of significand
  - 23 bit significand  $2^{-23} \approx 1.2 \times 10^{-7}$
  - About 6 decimal places



Floating point number range for 32 bit number with 8 bit exponent and plus minus 2 power 256. So, that is about 1.5 into 10 power 77 a huge number.

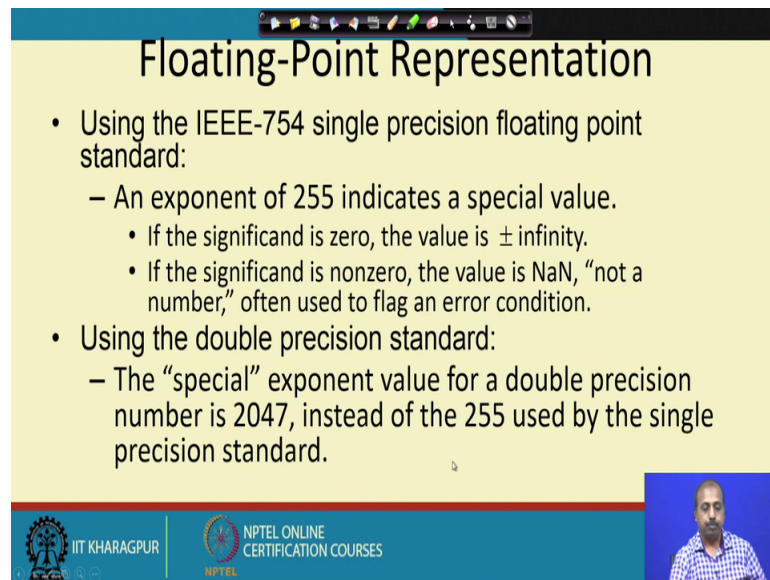
Now, accuracy so, this is the change that will come if you change the LSBs of the significant part. So, if I have got 23 bit significant then the accuracy is 2 power minus 23. So, that is 1.2 into 10 power minus 7. So, that is up. So, if this turns out to be 6 places after decimal. So, that is a huge number accuracy is good.

(Refer Slide Time: 30:08)



The numbers that you can express so, that way you can see that in 2's complement range. So, it is minus 2 power 31 to plus 2 power 31 minus 1, but in floating point numbers. So, you have got a huge range ok. So, this is minus 1 minus 2 power minus 24 to 1 minus 2 power 24. So, that way so, they that is a huge number of numbers that you can store here ok.

(Refer Slide Time: 30:35)



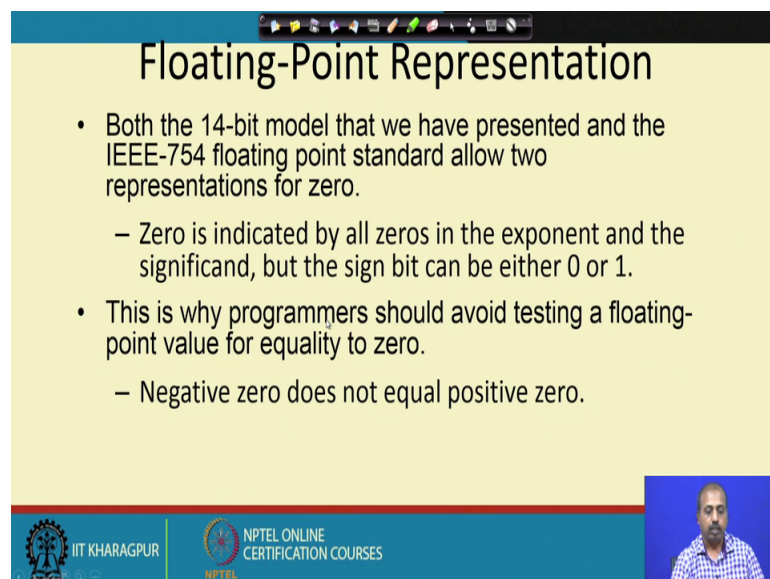
**Floating-Point Representation**

- Using the IEEE-754 single precision floating point standard:
  - An exponent of 255 indicates a special value.
    - If the significand is zero, the value is  $\pm$  infinity.
    - If the significand is nonzero, the value is NaN, “not a number,” often used to flag an error condition.
- Using the double precision standard:
  - The “special” exponent value for a double precision number is 2047, instead of the 255 used by the single precision standard.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, similarly we can have this floating point representation for this single precision floating point. So, these are this representation can be there and this flow 0 0 over this 14 bit model that we have used.

(Refer Slide Time: 30:38)



**Floating-Point Representation**

- Both the 14-bit model that we have presented and the IEEE-754 floating point standard allow two representations for zero.
  - Zero is indicated by all zeros in the exponent and the significand, but the sign bit can be either 0 or 1.
- This is why programmers should avoid testing a floating-point value for equality to zero.
  - Negative zero does not equal positive zero.

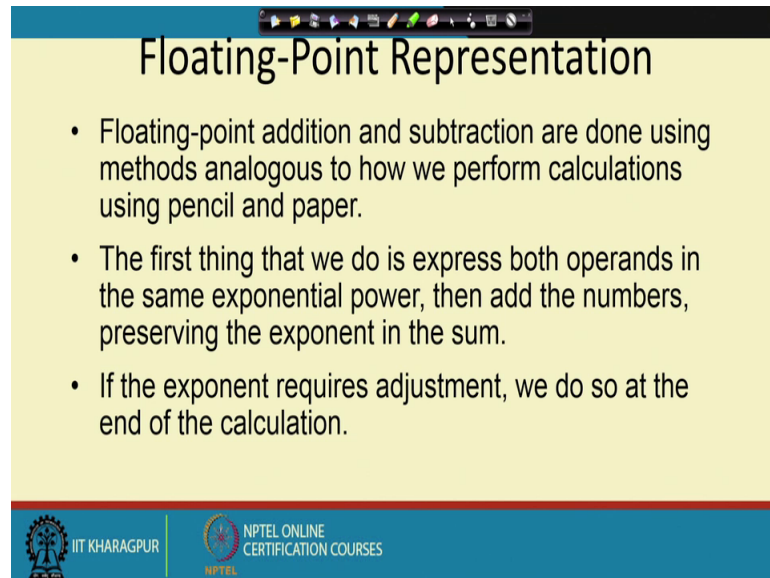
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, that is that that will also have similar such thing, but this 0 with this IEEE 754 so, this allows two representation of zeroes.

So, we have got minus zero indicated by all zeroes in the exponent part and the significant part, but the sign bit can be either 0 or 1. So, as a result there is a plus 0 and a

minus 0 in the in our representation as well as IEEE 754 presentation. So, this suggests that we should not have this floating point we should not check whether the value is 0 or not, because of the sign bit mismatch. So, it may not be giving the correct result in a program. So, you should not try to check whether  $x$  equal to 0, where  $x$  is the floating point number and this negative 0 does not include equal to positive 0.

(Refer Slide Time: 31:41)



**Floating-Point Representation**

- Floating-point addition and subtraction are done using methods analogous to how we perform calculations using pencil and paper.
- The first thing that we do is express both operands in the same exponential power, then add the numbers, preserving the exponent in the sum.
- If the exponent requires adjustment, we do so at the end of the calculation.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this floating point representation. So, it can we can do addition subtraction etcetera. So, that will be doing that we will do using a similar to integer arithmetic. So, this we will see in the next class.