

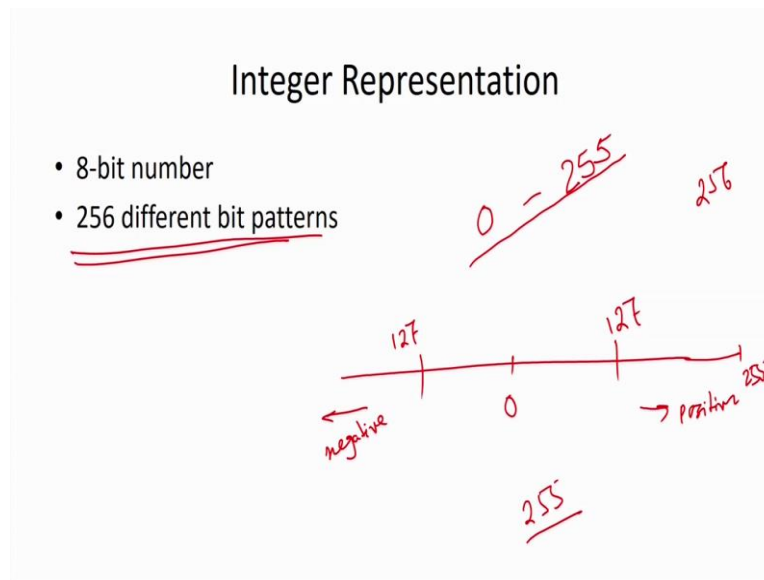
So, the similar information I am representing in another number system which is your hexadecimal. Now, you just see what I am saying this is your 8 bit number all 0s to all 1s. So, all 1 in decimal it becomes 255 in hexadecimal it is your FF, FF means 15 F represents $15 \times 16^1 + 15 \times 16^0$. So, in that particular case we will get that this is nothing but 255 in decimal.

Now, when we are going to discuss that representation of number in computer or in your binary system or binary digit most of the time we are going to take help of hexadecimal number system because it is having one advantage. Say, when I am going to work with 32 bit number I have to write 32 bits ok all 32 bits 0 or maybe combination of 0s and 1 which is slightly difficult. Similarly when we are going to work with 8 bits we are going to write 8 bits all 0s to all 1. Now, again if I see how to do it in your decimal number then we have to have slight your these things calculation now this is the calculation we have to do which is slightly time consuming for any number.

So, in that case for hexadecimal representation this very simple, you just see if I am having a 4 bit number then all 0 if I am having then this is your 0 and if it is all 1 you can say that $8 + 4 = 12$, $12 + 2 = 14$, $14 + 1 = 15$. So, this is your 15 so we need total 16 different symbol 0 to 9 and A to F, F is 16 and A is 10. So, now, to represent this number the maximum number of bits that we need is your 4 bit. So, when we are having a binary representation we simply group them in the 4 bits. So, in that particular case what will happen? We are going to say that this all 1 is going to represent F and all 1 is going to represent F. So, the range is from 00 to FF, when we are having 12 bit numbers then it will go from 000 to FFF. So, number of bit divided by 4 is going to give me the number of symbols needed for hexadecimal representation.

So, if I go back to one of my earlier slide. So, in 41 I am saying that this is the binary representation now what is the hexadecimal representation 0010 this is your representing 2, 1001 this is basically $8 + 1 = 9$, so 29. So, 41 in decimal number is equal to 29 in hexadecimal number system or in binary number system we can say 00101001. So, if you are going to work with 8 bit numbers we have to write 8 symbols in binary presentation, but in hexadecimal we are going to use only two symbol two digits. So, in most of the time we are going to represent our information in hexadecimal for understandability and for readability, but when you think about how it is working in the computer basically it working with this particular binary digit only in bit pattern 0s and 1s, but for readability we can write something in our hexadecimal notation.

(Refer Slide Time: 19:59)



Now, how to represent integers? So, in that particular case we are talking about the number system. Now, say if I am having 8 bit numbers. So, I am having a combination of 0s and 1s and 8 bits 8 symbols. So, in that particular case we are having 256 different symbols or combinations, so this combination will go from 0 to 255.

Ok, so if we are going to represent only positive numbers then what will happen I can use all those 256 character to represent positive numbers from 0 to 255, but if you are going to consider about negative number also this is the number line this is 0 and this is your positive side and this is your negative side. So, when we are going to work with a negative number then what will happen whatever 256 different bit patterns we are having some of the bit patterns need to be reserved for negative numbers also; that means, in positive number when we are on the dealing with positive number then we are going from 0 to say 255. But when you are coming for negative numbers then some of those symbols have to be used for negative numbers also; that means, gradually my range is going to reduce in the positive side maybe it will be half over here. So, I can say this is your 127 and this side I may have 127. So, this is basically $127 + 127$ 254 with 0 255. So, we are having total 256 symbols, but here we are using 255 one more symbols are still remaining. We will see how we are going to deal with that particular representation.

So, basically we have with 8 bit numbers we can handle 256 different numbers. So, if it is your only positive numbers we are going to deal from 0 to 255, but if we are going to handle negative

numbers then range will reduce it will go from some negative 127 to positive 127 or may be +
- so on.

(Refer Slide Time: 22:12)

Sign-Magnitude

- Left most bit is sign bit (MSB)
 - 0 means positive
 - 1 means negative
- +18 = 00010010
- -18 = 10010010
- Problems
 - Need to consider both sign and magnitude in arithmetic
 - Two representations of zero (+0 and -0)

Zero

8: 1 7

16 1 15

127

+127

-127

254

00000000

10000000

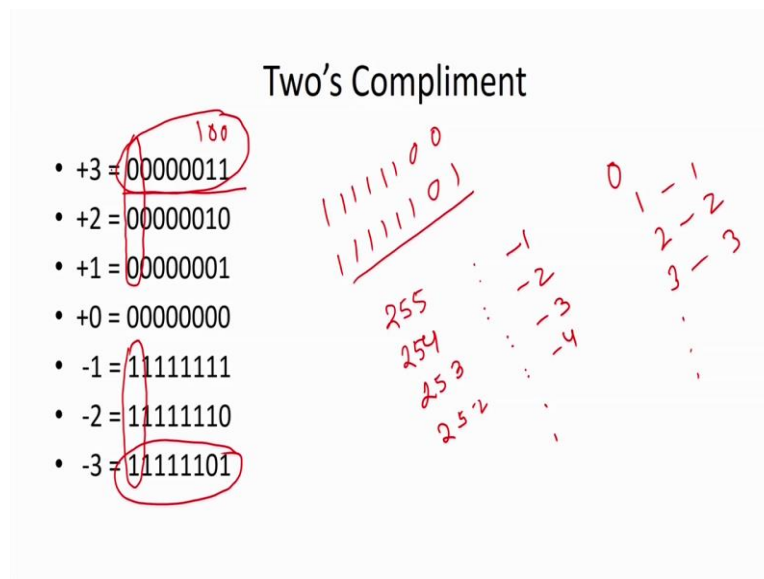
1st Magnitude

Now, for that we are having 2 ways of representing this number one is your sign magnitude in that particular case what will happen whatever bit pattern we have it will be divided into two part one part is known as your sign and other part is your magnitude. So, in that particular case what will happen? Say, for 8 bit numbers one bit will go for your sign and 7 bit will go for magnitude. So, if it is an 8 bit numbers. If it is a 16 bit numbers then 1 bit will go for sign and 15 bit is going to represent the magnitude of that particular number. So, here what happened what convention we are using 0 means the positive numbers and 1 means the negative number. So, in that particular case for example, you just see + 18. So, this is the magnitude 10010 this is $2^4 + 2^1 16 + 1 18$. So, similarly this is the magnitude of the number again it is 18, but now we are saying that 0 is my indicating the sign with it is a positive number. So, it is + 18 and this bit more significant bit is 1, 1 represent the negative numbers. So, this bit we are having going to give me - 1. So, in that particular case now we can represent the numbers positive number as well as negative numbers.

Now, in that particular case now what will happen, if you take this particular 7 bit then it can the value that numerical that it can have go up to 127. So, with 0 I can go up to +127 and with most significant bit 1 we can go up to 127. So, this is $127 + 127$. So, total 254 now we are having total 256 bit pattern what is the remaining bit pattern you just see that if you look into

it what we are getting we are having that bit pattern 1 2 3 4 5 6 7 all 0 this is representing 0 or maybe one. So, this is also if you look into this magnitude, magnitude is always 0, but with this bit 0 and 1 we are getting positive 0 and negative 0; that means, there are 2 representation of 0. So, in this particular case we are representing 256 different numbers, but out of that valid numbers is 255 we are having 2 representation of 0. So, there we may look to eliminate this also because why we should use two bit pattern to represent the same number, so for that we may go for some other representation which is known as your two's complement.

(Refer Slide Time: 25:03)



So, in this two's complement just I am giving one example just only getting phone numbers now in that particular case what will happen this is the representation if I am having 0 then these are all 0s, + one is simple that magnitude 1, 10, 11 like that if I go for positive 4 then it will be 100 like that and negative number is coming as - 1 is representing as your all 1s then - 2 is coming as your all 1 and 0 and - 3 is coming as your all 1 after that six 1's 01.

So, basically what happens? It is basically the maximum number is a 255 or 8 bit representation that 255 is going to represent my -1 now you just count down and go downwards then 254 is going to give me -2, 253 will give me your -3, then 252 will give me -4 like that and from 0 if you go in upward direction say 1 is going to represent 1, 2 is going to represent 2, 3 is going to represent 3 like that. So, this is the simple way we can visualize it. So, if we are having a number in two's complement form we are going to represent in this particular way.

(Refer Slide Time: 26:31)

Two's Complement

- 1's complement and 2's complement

The diagram illustrates the process of finding the two's complement of a number. It starts with the binary representation of a positive number, 00000011 (decimal 3). The first step is to find the 1's complement by inverting all bits, resulting in 11111100. The second step is to find the 2's complement by adding 1 to the 1's complement, resulting in 11111101. A carry-over is shown for the addition of 1. The diagram also includes a box showing the addition of 37 and 63 to get 100, and a calculation 37 + 63 = 100.

Now, how to do it? We are having one is called one's complement and two's complement. So, if I consider a particular number say 00000011 this is a number that I am considering then what is the one's complement, one's complement basically it says that you complement all the bits of this particular number. So, I am going to get 11111100. So, this is the one's complement of this particular number. And what is two's complement it says that in case of two's complement whatever we are getting in one's complement add 1 to it. So, what I am going to get 11111101. So, this is the two's complement.

Now, if this is my some number given number I can get the one's complement just complementing all the bits and I think in digital system we can do it by using a NAND gate or maybe using a NOT gate that we have discussed these things our digital block. So, after that we are going to add on to it finally, we are going to get it. Now if I go back to my previous slide you just see that we are talking about we are taking this particular number one. So, if I am going to take the ones complement I am going to get 11111100 now if I add 1 to it then I am going to get 11111101. Now, you just see whatever I am getting or taking the number after looking into the two's complement I am getting this representation.

So, generally we are saying that this is my positive 3 and other one is my negative 3. So, after taking the ones two's complement we are going to get the negative representation of the given number. So, this is the task that we are going to see and why we are coming with this particular number system and why we call it complements you just see that I am taking this particular

number 00000011. Ok, now I am taking the two's complement what I am getting 11111101. Now, in that particular cases I am going to add this two numbers then what I am going to get 1 and 1, 0 we having a carry 1, 1 and 1 0. So, I am getting a carry one. So, and finally, 1 carry out. So, we are getting this number.

Now, what will happen we are working with 8 bit numbers; that means, we are having provision to store this particular 8 bits only and this is a carry out which is not a part of my number because I can represent the number with the help of 8 bit only. So, carry cannot be incorporated. So, we are not going to consider this particular 1. So, finally, what I am getting the after adding this number we are getting 0. So, when we add two numbers and if the result is 0 then what we say that one is the negation of the other because if I add two numbers $a + (-a)$ then we are going to get result is 0. So, this is the principle that we are using. So, in two's complement form we can use we get the negative of a given number or if we are giving a negative number then we are going to get the positive of that particular number.

So, we are going to use this number. This is related to any number system see what we can do if I look into these things that binary decimal number system also in that particular case also we can have this particular concept of complement. So, in that particular complement here we are saying that just take the complement that means 0 will be replaced by 1, 1 will be replaced by 0. This is basically the subtract the number from the maximum digit that you have in that number system. So, this is basically $1 - 0$ is going to give 0. So, now, if I take any decimal number system say 27 then what we are going to do, we are going to subtract this number from the maximum digit which is 9. So, $9 - 7$ is going to give me 2, yes I am going to say that this is 37. So, this is your $9 - 3$ is 6, ok 62 now if I am going to add 1 to it then I am going to get 63.

Now, we add up 37 and 63 what we are going to get 0 9 10. Now, we are working with two digit only the way here we are working with 8 bits we are using two bits two digit only. So, this digit cannot be stored. So, eventually what will happen adding this two number is going to give me the result 0 so that means, we can say that one is the negation of the other if we are using the number system your 10's complement. So, for decimal number system also we can use that complement 10's complement in case of 10's complement first we are going to get the 9's complement the way we are getting the one's complement this is the 9's complement and after adding 1 we are going to get the 10's complement. So, this is the principle we are using to represent the negative numbers. So, already I have explained these things what is the

benefits. So, in this particular case first take the complement of given number then add 1 to the LSB then we are going to get the negation of that given number.

(Refer Slide Time: 32:13)

Benefits

- One representation of zero
- Arithmetic works easily
- Negating is fairly easy
 - $-3 = 00000011$
 - Boolean complement gives 11111100
 - Add 1 to LSB 11111101

Now, what already we have seen?

(Refer Slide Time: 32:27)

Negation Special Case 1

- $0 = 00000000$
- Bitwise not 11111111
- Add 1 to LSB $+1$
- Result $1/00000000$
- Carry-out is ignored, so:
- $-0 = 0 \checkmark$

*-127 to +127
2 representation
of 3200*

This is a special case already we have seen that now in that particular case negative 0 is equal to a positive 0; that means, you having only one representation of 0. So, in sign magnitude form what will happen we are having -127 to +127 total 256 number along with that two representation of 0 2. So, total 256 symbols.

Now here in this particular case after adding it since we are going to discard this particular carry out bit because we cannot store it because we are working with 8 bit numbers. So, finally, positive 0 and negative 0 is becoming same. So, we are having only one representation of 0.

[illegible]

Now, why we are taking -128 why we are not taking these things your +128 because this is an extra symbol know we are getting -127 to 0 and to +127, now these are the 255 different bit pattern we are using now this is the bit pattern it is remaining there. So, we can use this particular bit pattern to represent another number and we are saying that we are representing it as a -128, but why not +128. So, in that particular case you will find that just concentrate on this particular most significant bit it is 1. So, in the representation when we are going to represent the number with 2's complement we will find the positive number should start with

0; that means, most significant bits are 0 and negative numbers will start with 1; that means, most significant bit is a 1.

Since this bit pattern is starting with 1 so, if we are going to treat this as a negative number then we say that it is going to represent -128 not +128; that means, now my range is go from -128 to +127 total 256 different representation. So, if we come back to this particular slide you see that for negative numbers this most significant bit is always 1 and for positive number this most significant bit is 0. So, that's why when we are coming for this particular special case 1 all 0s that will represent a negative number and the negative number is your -128. Now, what is the range of number?

(Refer Slide Time: 35:54)

Range of Numbers

- 8 bit 2s complement
 - +127 = 01111111 = $2^7 - 1$
 - -128 = 10000000 = -2^7
- 16 bit 2s complement
 - +32767 = 01111111 11111111 = $2^{15} - 1$
 - -32768 = 10000000 00000000 = -2^{15}

$0 - 2^{n-1}$
 $2^{n-1} \text{ to } (2^n - 1)$
 (-2^{n-1})

Already I have mentioned it. So, in that particular case, for integer sorry for positive number we can go from 0 to 255, 255 if it is an 8 bit number. So, for negative numbers if we are going to handle a complete range of integers positive and negative then for 8 bit numbers the range will go from -128 to +127.

And for 16 bit number it will go from 32768 to 32767. So, -32768 to +32767 and basically it is in $2^{15} - 1$ this is $2^7 - 1$ because we are using 7 bit to represent my magnitude and this bit again a going to give me the indication about negative and positive number. So, it is basically $2^n - 1$ to -2^{15} . So, if we are going to work with n bit number then my range will be your

$-(2^n - 1)$ to $2^{n-1} - 1$. So, this is the range that we are going to have when we are going to use two's complement form.

So, this is just in a tabular form we are writing this two's complement number in 4 bit numbers.

(Refer Slide Time: 37:16)

4-bit numbers

- 4-bit numbers in 2's complement form

Decimal	Binary	Decimal	Binary
0	0000		
1	0001	-1	1111
2	0010	-2	1110
3	0011	-3	1101
4	0100	-4	1100
5	0101	-5	1011
6	0110	-6	1010
7	0111	-7	1001
		-8	1000

So, this is your 0 then 7 this are the positive number and -1 will represent at the all 1 and this decremented and finally, -8 will be your 100.

(Refer Slide Time: 37:33)

Sing, Carry & Overflow

- 4-bit numbers in 2's complement form
- Perform the operations:
 - $7+7$, $(-7)+(-7)$, $7+(-1)$, $1+(-7)$, $7-5$

The slide contains several handwritten diagrams illustrating 4-bit operations in 2's complement form:

- 7 + 7:** $0111 + 0111 = 1110$ (14). A carry of 1 is shown from the MSB.
- (-7) + (-7):** $1001 + 1001 = 0010$ (-14). A carry of 1 is shown from the MSB.
- 7 + (-1):** $0111 + 1111 = 0110$ (6). A carry of 1 is shown from the MSB.
- 1 + (-7):** $0001 + 1001 = 1010$ (-6). A carry of 1 is shown from the MSB.
- 7 - 5:** $0111 - 0101 = 0010$ (2). A carry of 1 is shown from the MSB.

Arrows labeled "Carry (out)" and "Carry (in)" indicate the flow of carries between bits. The term "overflow" is written near the final results.

Now, just look for some operations say I am giving that $7 + 7$ now how we are going to edit 7 it is a 4 bit numbers just remember that we are having a 4 bit numbers. So, 7 is a nothing, but 0111, 0111. So, this is 7 and 7 now if I add them then what I am going to get 0111 ok this is the things that I am getting. Now you just see that $-7 + -7$ now what is the representation of - 7 its two's complementation 1001, 1001. So, this is your 1001 and 1001. So, if I add them what I am going to get 01001. So, this is your carry out we cannot consider it because we are working with 4 bit number.

Now, considered that $7 + -1$, 7 is your 0111 and what is -1, -1 is this all 1 1111, now this is your 01101 again I cannot consider these things and $1 + -7$ so, 1 is your 0001 and what is my - 7, 1001 1001. So, this is your 0101. Now, you just see what we are getting actually. So, in this particular case this is your +7 this is your +7 now we are working with two's complement. So, 1110 what does it means 1110 is your -2 here this is your -7, -7. now 0010 what I am getting 2 these are decimal equivalent this is your 7 and this is your -1.

So, what I am getting 0110 which is your 6 this is your 1 and this is - 7. So, what I am getting 1010. So, what is that 1010 is your - 6. So, I am getting - 6. So, just I am calculating it just show it now what will happens say when I use - 7 to - 1 should get - 6 and I am getting it. So, this is a correct result when I am using this things - 7 and - 1 I am getting + 6. So, this is also correct result, but when I am using $-7 + -7$ what should be my result my result should be 14, but as a result I am getting 2 only. Again when I am using $+7$ and $+7$ I should get + 14, but what I am getting over here I am getting - 2.

Ok so, in this particular case, whatever result we are getting these are not correct because we have working with 4 bit numbers and you can handle 4 bits only. So, in that particular case these are not the correct result because we know that for 4 bit numbers my range is from - 8 to + 7. So, this is the range, but these are the valid numbers but after adding I am getting a bigger number if it is a positive result say which is your 14 no doubt about it, $8 + 4 + 2$, but since we are going to handle negative number. So, eventually it is represented as - 2. So, these are the two result at I am getting which is not correct and this situation is basically known as our overflow because we cannot handle + 14 over here this is the range. So, this is your overflowing the number. So, this is the overflow situation, but in these two cases I do not have any problem and what I am getting that correct result these are the two correct result.

So, this is basically we should talk about the overflow situation; that means, we are trying to perform some operation computer is doing it or digital system is doing it and essentially it is giving me some result, but this result is wrong I cannot consider it. So, we are going to say these are the overflow situation. Secondly, if I look for this particular four calculation one thing you just see that here it is having a carry out this one is also having a carry out it is generating some carry, but other 2 combinations are not generating any carry bit this carry out is 0. So, in that particular case what we will say that these two operation is generating some carry for me, but one is your correct result second one is not correct result. So, we are having some situation called whether after performing some operation whether it generates carry or not or secondly, whether it is a valid result or not, if the valid result is not valid then we say it is an overflow situation. So, these are the terms we are going to use while going to design our computer.

Now, another important issue you just see that I am giving, one more question that you perform $7 - 5$. So, in that particular case how we are going to perform this operation $7 - 5$ I know that this can be done with your $7 + - 5$. What is 7? A binary representation 0001. And what is the representation of $- 5$? 1011, 1011. If you add these two things then what you are going to get 01001. Ok $7 - 5$ is 2. So, I am getting 5 only. So, this is your 7 and this is your $- 5$ because we know that result is 2 I am getting it.

Now, what is the observation over here? You just see when we are representing in two's complement form to do the subtraction we can use the adder circuit itself just take the negative representation of the number and after that add them together and this is the way we used to do. So, in two's complement form if we are using two's complement representation basically adder circuit can be used to evaluate the subtraction also. So, in this particular case $7 - 5$ we are getting result is 2 and you just see that it is also generating 1 carry. So, carry generated, since it is a valid result so it is not an overflow. So, this is also valid result.

Now, when we are going to say that it is an overflow just you observe this particular result and from that we can conclude it actually you just see that this is having carry 0 and this is your carry out is 0. So, in that particular what I can say that carry into the most significant bit and the final carry out. So, this is the carrying to the most significant bit and this is the carry out this is the carry in to the most significant bit and this is the carry out. So, these are overflow situation this is the carry in to the most significant bit and this is the carry out of the operation. Here also this is the carry in to the most significant bit and this is the final carry out now you just see in these two situation these are my correct result. Here also I am having this ok now

the observation is like that if the carry in to the most significant bit and the final carry out if they are same then we don't have any overflow say these are the same situation.

Ok these are the same situation. So, we don't have overflow, but in case of overflow these 2 bits are different say it is 0 and final carry out is 1 it is your carry in is your 1 and final carry out is 0. So, if these two situations are different then we are going to get an overflow and this is nothing, but my exclusive or scenario; that means, with the help of an exclusive OR gate I can check whether overflow occurs or not. So, what I can say this is the carry out and this is the carry out final and this is the carry in to MSB most significant bit which is a sign bit. So, with the help of this exclusive OR gate we can detect whether overflow occurs or not.

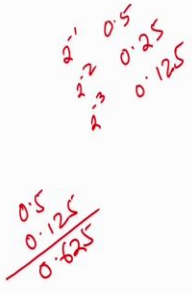
So, these are the situations. So, what I am saying that here I am talking about overflow. So, this is the way I can detect the overflow. I have seen whether it has generated carry or not. So, with this carry out bit I can check whether carry has been generated or not and sign bit what is the sign of the result basically this most significant bit is going to give me the sign bit ok. So, I will just look into this particular position and see whether the number is positive or negative. So, since it is 1, it is a negative number and this going to represent - 2, since this bit is 0 so it is going to represent positive numbers so this is + 2. So, like that these are the three information we can collect when I am going to perform addition operation in two's complement form and we require those information while implementing of computer.

So, we have seen how to represent positive numbers how to represent integers; that means, positive as well as negative. Now we are going to see how we are going to represent the real numbers.

(Refer Slide Time: 48:04)

Real Numbers

- Numbers with fractions
- Could be done in pure binary
 - $1001.1010 = 2^4 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Where is the binary point?
- Fixed?
 - Very limited
- Moving?
 - How do you show where it is?



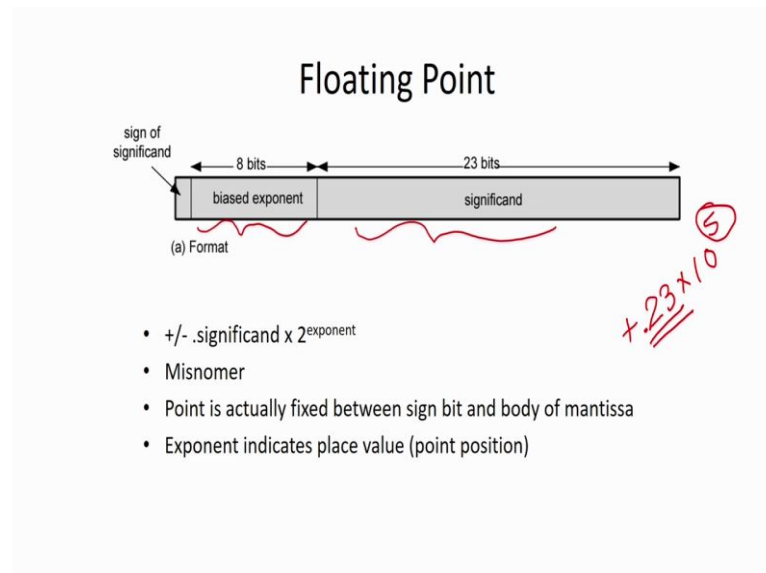
So, how we are going to represent real number it is again everything we have to deal with your binary number system only 0s and 1s and you should know what is the decimal equivalent of a real numbers if we are going to represent it in your binary number system.

So, in that particular case just look for the particular simple example. So, this is the decimal point, so that is why these are the fractional point. So, how we are going to get it this number is your $2^4 + 2^0$ and this are decimals. So, this is $+ 2^{-1}$, 2^{-2} , 2^{-3} and 2^{-4} . So, this is your 4 + 1 this is 9. So, now 1001 is 9 and 2^{-1} is your 0.5, this is your 2^{-1} , what is your 2^{-2} this is your 0.25 half of this, 2^{-3} this is half of this, so 0.125. So, this is basically $0.5 + 0.125$. So, this is your 0.625. So, finally, my number is your 9.625. So, this is the decimal representation and finally, we are in this binary representation. So, everything we have to do binary representation.

Now, main issue is now where I am going to keep this particular decimal number. So, if it is like that I am having a 8 bit numbers we are keeping 4 bits for before decimal and 4 bits after before decimal then what will happen then I can go up to 15 only it is your positive number and if it is your negative number then I can go from - 7 point something to - 7 point something and what is that point something we can simply add those particular numbers and we can get it. So, if I am going to have this things we will say this is a fixed point we have finding or we are putting the position of the decimal point as a fixed one and it will always appear at a fixed position. But it is very limited in computer system I am not using it, but another one is moving means it is according to our numbers we will adjust the particular decimal point which is

basically known as floating point number fixed point representation and floating point representation. In case of moving it is basically floating point representation and I think if you are writing some problem in your C language I think you know about the floating point number. So, this is the floating.

(Refer Slide Time: 50:44)



Now, how we are going to do it? This is basically in floating point number we use this particular representation. So, most significant bit is always used to indicate the sign bit whether it is positive numbers or negative numbers 0 means positive 1 means negative then we are having some exponent part we say it is a biased exponent we will see and we are having the significant part. Ok that means, if I am going to represent in decimal number system say $+ 23 \times 10^5$ then what will happen, this is the exponent part and this is the significant part or the mantissa part. So, this is the number presentation if they having $+$ - then decimal point sorry here decimal point will put it over here and into 2 power exponent. So, always decimal is having over here.

Now, this is the way we are going to represent now in negative binary number system also those things will be represented in my binary number system. So, it is having exponent part and significant or mantissa part and along with that sign part.