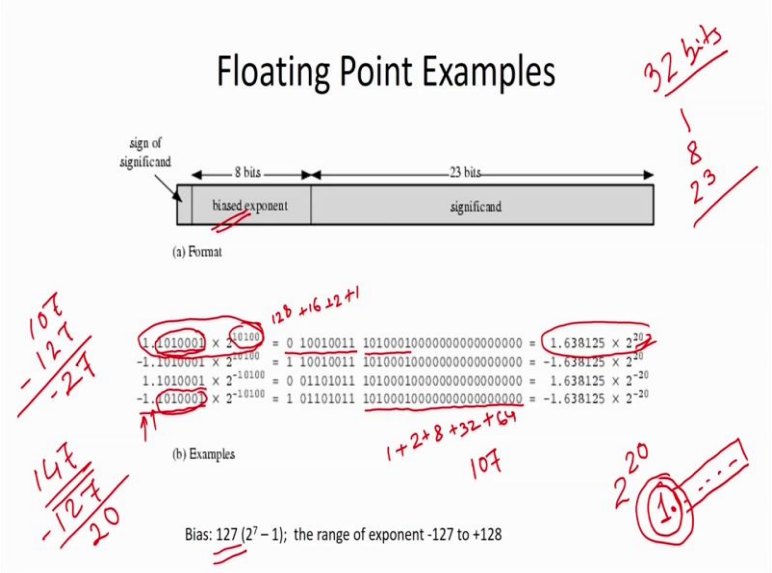
(Refer Slide Time: 51:56)



Now, just look for this particular representation. So, what is the size of this particular representation this is your 32 bits 1 bit is for sign bit, 8 bit is for your exponent and 23 bit is for significant. So, now, if we look into this number representation say 1.10100001 into 2^10100. So, if this is the number that we want to represent in our floating point represent then what will happen, in that particular case we have to see what is the significant part the significant part is your 1010001, 1010001, so 10010001. So, this is the significant part and it is having total 23 bits. So, remaining part will be all 0s.

So, this is basically same numbers positive and negative positive exponent and negative exponent now what is the exponent over here you just see here I am talking about 2 to the power. So, this is your 1 2 4 8 16, so 16 + 4, 20. So, basically we are talking about 2^20. So, this is the representation 2^20 and if I convert it it will come like that in decimal equivalent the value.

Now, in this particular case we are having 2^20, but what we have stored over here you just see what is this equivalent 1 2. So, basically this is your 1 2 4 8 16 + 32 64 128. So, finally, this is the number that you are getting, so 128 + 19. So, we are getting 147. So, what we are storing basically in the exponent part we are storing 147, but basically we are looking for 20, 2^20, but instead 20 we have storing it your 147. So, this is called biased exponent because now exponent may go positive as well as negative. So, instead of doing this thing what will happen we represent everything in the positive number and it will be biased by some numbers.

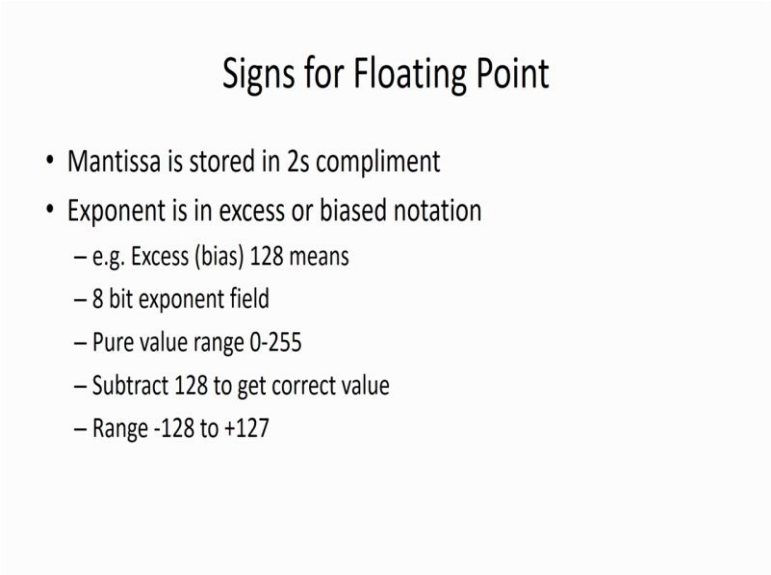
So, in this representation it is biased by 128; that means, whatever number we are storing over here that will be that to find out the exact exponent that 127 will be subtracted from that. So, 147 - 127 what I am getting this is 20, so this is the 20 that we are getting. So, this is the biased exponent.

Now, just look for the negative now this is a negative exponent this is your - 20, but for - 20 what we are storing 1 + 2 + 8, 8 + 16 + 32 + 64. So, these are the things that we are storing in the exponent. So, what I am getting 96 + 8, 104 + 3, 107.

Ok now, we are storing 107 in the biased exponent. So, what is the exact value we are storing. So, what we have to do we subtract 127 from 107, 107 - 127, so we are going to get - 27. So, this is the way we are storing our floating point numbers. So, it is having 3 component, one is your sign bit then biased exponent and significand and what we are storing you just see observe these things we want to store 1.1010001 and we have stored this particular information in the significand part. We have not stored that 1 and decimal point; that means, we are not going to store the decimal point it is implicit and what is the decimal point whatever we have stored in a significand that will be appended by 1 point that number. So, this is 1 point something then what happen we are not storing it this information whatever we have stored that will be always consider as 1 point that number.

So; that means, if I am going to give going to store some number then what will happen first we have to normalize it and the normalization will come like that decimal point will be always after 1 digit. So, this is the normalization. In decimal number system also we do the normalization. So, these are the things that we are having. So, mantissa will be stored in your 2’s complement form exponent will be stored in your biased exponent. So, this is basically it will be stored in a biased exponent and after that what happen some values we need to subtract it may be from 128 or 127 for 8 bit numbers.

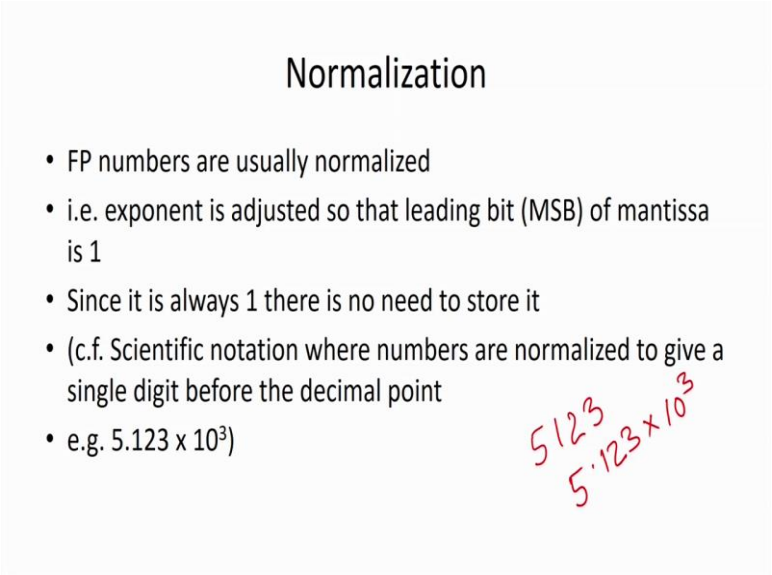
(Refer Slide Time: 56:48)



So, we must understand this 3 component while we are going to find out what is the exact value that we are storing in this particular representation. So, for this particular representation say if I want to store this particular number 1.638125×2^{20} basically in binary representation we are going to have these things. So, for that this part we are going to put in the mantissa or the significand part and whatever we are having this particular exponent it should be according to the bias that we have used here we have used bias as 127. So, whatever value we have we have to add 127 to it and store that particular number as an exponent and along with that we are going to store the sign bit.

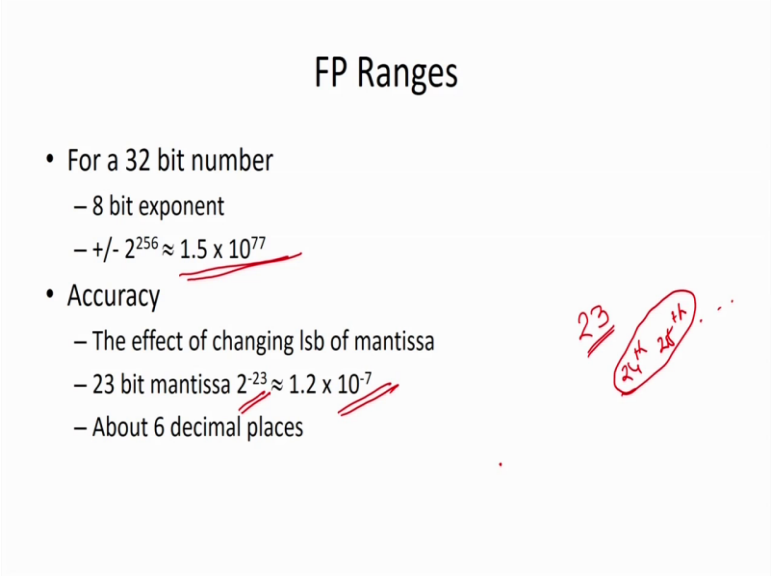
So, if you know the information then very well you can find out what is the value of this particular number and how we are going to get the value, basically it is in normalized form this particular information 1 and dot we are not storing that remaining part we are storing. So, this is the way we are going to do it.

(Refer Slide Time: 58:19)



Because this is similar thing, whatever we are talking about. So, in floating form number it should be usually normalized that is exponent is exhausted. So, that leading bit MSB of mantissa is 1. So, leading bit is 1. So, this is similar to your decimal number system say if I am going to talk about say 5123 I can look for 5.123×10^3. So, this is the way we have to look into it. So, it have to be first normalized one, the number is there is only 1 bit prior to that decimal point and after that we are having a mantissa and exponent whatever exponent we are going to store it should be in the biased form because why we are teaching the bias notation just to avoid the negative numbers represent store negative numbers in the exponent point.

(Refer Slide Time: 59:13)

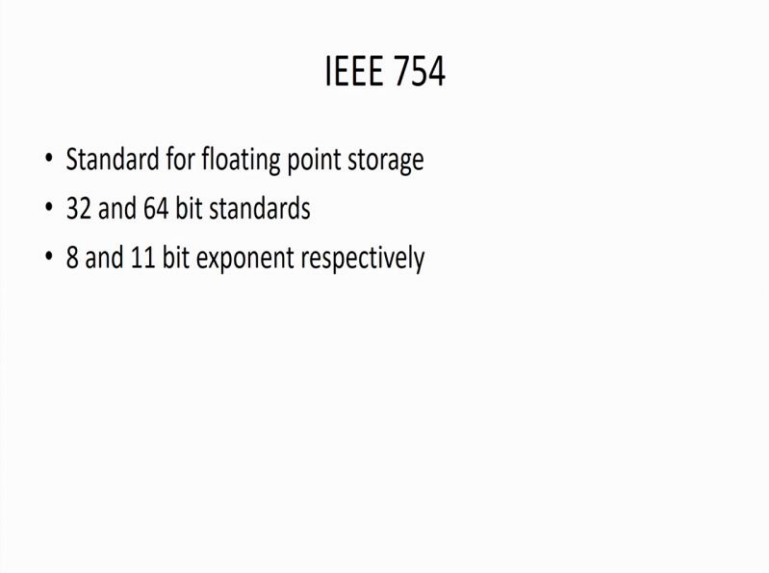


Now, what are the ranges of your floating point numbers? Again it depends on the number of bits. So, here we are talking about the 8 bit numbers and in this 8 bit numbers have to talk about what is the 8 bit exponent. So, if the exponents is 8 bit; that means, we can go up to 2^{256} so; that means, we can go up to 2^{256}; that means, if I could convert it to the decimal number you will find that we can go up to 10^{77}. So, if it is a 32 bit representation and we are going to use 8 bit for exponent then what we are going to get we are going to get range up to 10^{77}.

Now, what is the accuracy? And accuracy basically talk about what is the changes of the value if we are going to change the least significant bit. So, in that particular case we are using 23 bit in our mantissa, but when we are going to convert into binary numbers it may go we may have that 24 bit 25th bit after decimal point. So, but we cannot store it so; that means, we are losing some information, but these are the least significant bit. Ok so, up to 23 bit we can store it, but 24 bit we cannot store so that what is the changes of the number if we change this particular least significant bit and it will go from 0 to 1 or 1 to 0. So, that's why the accuracy is your 2^{-23}; that means, in decimal number it is your 10^{-7}; that means, about 6 decimal places we can have the accuracy beyond that we cannot have the accuracy.

So, you just see that if we increase the number of bits that range will increase as well as accuracy will also increase. So, these are the two issues that we are having range and accuracy in floating point number. So, we are having a standard call IEEE standard and in most of the cases we use this particular standard because we should not come up with our own number system because globally it should be accepted so we are having a body call IEEE Institute of electrical and electronics engineers is a global body and for many systems for digital computers even for your computer networks even for your communication they give the standard, they freeze the standard and after that all parties use those particular standard.

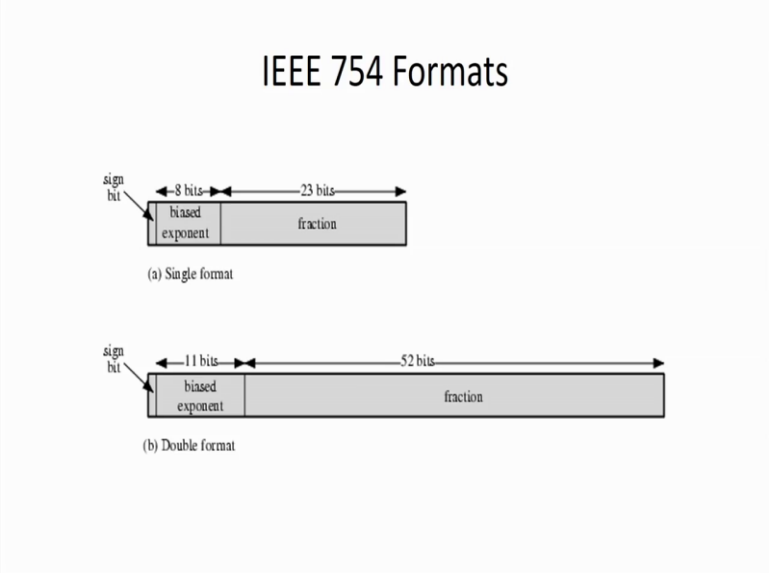
(Refer Slide Time: 61:17)



So, for floating point representation also that IEEE has given a format which is known as your 754, IEEE 754 format and in that particular format they are having two format one is your 32 bit and another one is a 64 bit because if we are going to construct the computer which we say that processor is a 32 bit processor; that means, you can represent information with the help of 32 bit. So, in that particular case you can look for the 32 bit standard. If your processor is your 64 bit then you can handle 64 bit together then what will happen when you are going to handle floating point numbers you can look for that 34 bit standard. And in that particular case what is the differences in case of your 32 bit it is your exponent is your 8 bit and in case of your 64 bit the exponent is your 11 bit. So, in case of range for 8 bit I am saying that we can go up to 10^77.

Now, when we increase the exponent from 8 bit to 11 bit similarly range will also increase now you can calculate what will be the range that we can handle when we are going to use 32 bit format and what is the range that we can use in your 11 bit format. Now, again you just see that for mantissa part also now we are increasing from 23 bit to around 54 bit it seems because 64 - 11 is your 53 - 1 is for sign bit. So, 52 we are using 52 bits for the mantissa part or significand part; that means, our accuracy is now going to increase for 22 bit we are getting the up to 10^{-6} now here we are going to get many more. So, if we are having more number of bits range is more accuracy is also more.

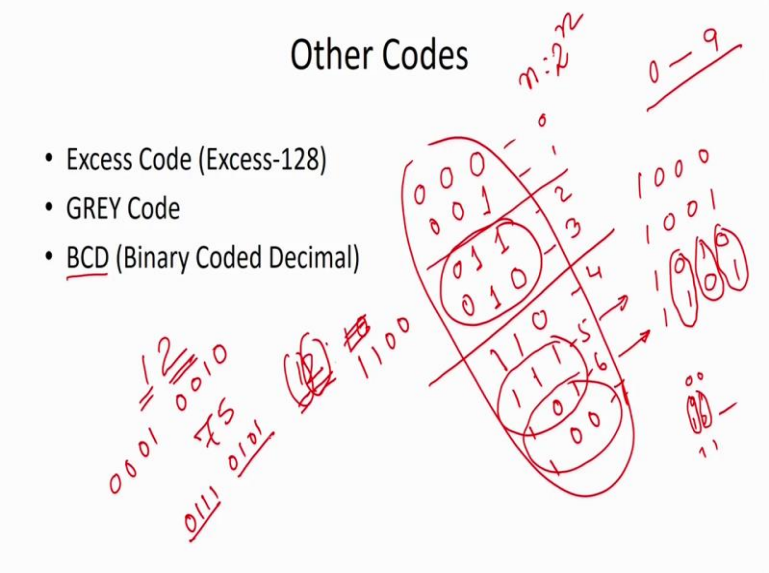
(Refer Slide Time: 63:39)



Now, this is the format, IEEE 754 format, this is your 32 bit format 1 bit is your sign bit, 8 bit is for biased exponent and 23 bit is for fraction. Similarly for your double format this is your 64 bit. So, already I said that fractional part is your 32 52 bit, 11 bit is your biased exponent and 1 is for sign bit. So, I think if we have written any program in your C language I think we are having that one of the data type is double.

So, that means, when we are going to work with that double data type it is a floating point presentation and the representation is the double format of IEEE 754 format; that means, we are going to use 64 bit to represent our number ok and another case we can use 32 bit to represent our number. So, this is the format of representing floating point number. So, we have seen how to represent numbers basically it is either integer number or real numbers. So, in case of real numbers then we will use floating point representation. There is some other code also there. So, it is your binary code. Already we have discussed about that excess code excess 128 or excess 127. So, this is basically depends on the number of bits that we are using.

(Refer Slide Time: 64:50)



So, if we are using 8 bit numbers then excess will be your excess 128 because to represent negative number as well as positive number what will do we store the positive number then we subtract something from that particular number to get the exact number.

You are having another code, code called GREY code. So, this is just for your information I am giving it. So, what will happen this is a code that will have so that 2 consecutive number is going to have a minimum signals sense. Say as for example, if I say that this is your 8 in binary and what is your 9, this is 9 and what is your 10 what is your 11. So, this is 10, this is 11 in that particular case when I am going to get next consecutive number you just see there is a change of bit pattern in 3 position. Ok in some cases it may be more also if we are going to look for more number of bits. So, GREY code is consists just to minimize this particular changes of bits when you go from one number to the next number. So, for that what basically their GREY code is going say if this is my 0 then next number is 1. So, basically 0 and 1 is going to represent decimal 0 and decimal 1. So, 00 say if I am going to have 2 bits number, now this is representation 0 this is representation 1 then I am going to have representation 2.

So, in case of representation 2 what we are going to do basically. So, in your binary number system it is your 00, 01, 10 so when I am going to from 1 to 2 there is a change of bit position in 2 different position, but here we want to minimize it. So, we want to just say that there will be changes in 1 position. Then what we are going to do we take the mirror image of these things 1 and the most significant bit we are going to get 1 over here because already two 0s are there

now we are going to have this thing. So, if I take mirror image of this thing. So, I am going to get 1 0 and 1. So, this is going to represent to this is going to represent 3. So, this is now changing. So, here 2 is 10 and 3 is 11 in binary number, but in GREY code it is different.

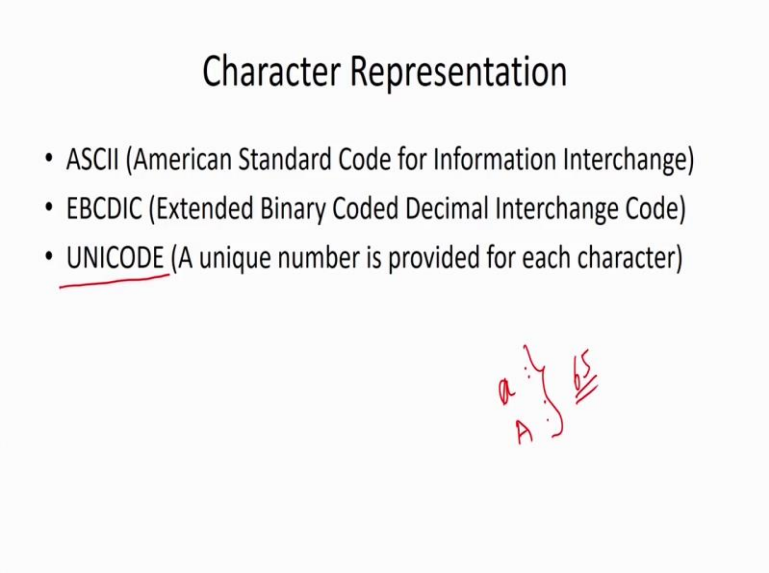
Now, when I will go for more numbers then what we are going to say 3 bits number then what I am going to get, take the mirror image of these things. That this term so, I am going to get 10. So, this is 10. So, this 1 will become 0 will be come 1 now. So, this is changing only 1 position. So, this is my 4. So, this is mirror image of 1 and 1 it is coming over here and this bit is 1 now it will be 01 1 and this is your 001. So, this is 4 5 6 7 in decimal. So, this is the concept that we use in our GREY code. So, basic principle is to minimize the number of changes of bit when we are looking for two consecutive number you just see that if we take these 2 consecutive numbers there is change in only 1 bit position if you we these 2 consecutive number this change is only bit position. If we consider these 2 consecutive number the change of bit position is only 1. So, this is GREY code. In some system we may use GREY code also.

Another one is your BCD binary coded decimal. Basically what will happen? We are accustomed with the decimal number system now when we convert something from decimal to binary it is slightly taking some time or getting a different pattern. As for example say if I am going to have 12 then bit pattern is your 10 sorry bit pattern is 1100, 8 + 4 = 12. But in binary coded decimal what we are going to do digit wise we are going to convert to the binary and for that for every digit we are having total 10 symbol 0 to 9 and to represent 10 symbol we need at least 4 bits of information you just see that it is your number of symbols and number of bit pattern is related to each other.

So, with your n bit we can go up to 2^n different representation already have mention it. So, with 3 bit I can go up to 2^3 is 8; that means, 0 to 7. So, we are having 2 more numbers 8 and 9 for that we need 1 more bit. So, we need 4 bit of information and this is basically in binary coded decimal these digit if 12 these will be converted to binary or coded to binary digit wise. So, this is 1 is going to represent 0001 and 2 will be represented by 0010. Like that if I am going to represent 75 in your BCD then will be your 0111 and 5 is your 0101 so 7 5. So, this is the binary coded decimal.

So, in some system we can use these things also and what will happen our calculation will become easier because; that means, you will feel that you are working the decimal system itself.

(Refer Slide Time: 70:25)



Now, in computers we have to work with the character also because in numbers we are doing some arithmetic operation, but sometimes we have to work with number system because now you just see that in everywhere we are using computer you are writing letters also with the help of computer we have to say how to represent A, how to represent B and like that. So, here also in this representation we are saying that character representation saying that character c h a r a like that, but inside computer when we are storing it we cannot store c as it is everything has to be stored as a binary number not only binary number it is at the high voltage and low voltage already I have mention these things.

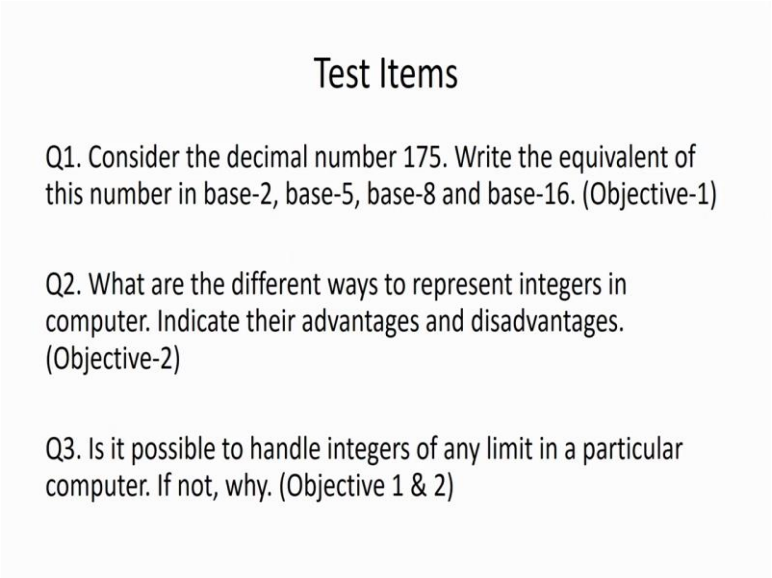
So, for every character we must have to assign some code. So, we are having several codes to represent the character the first basic code is your as ASCII, A S C I I American Standard Code for Information Interchange. So, this is a first code it is developed I think some of you may be knowing that when I am going to represent I think capital a or lower case a then we need some number I think to represent one of these numbers is your 65 in decimal numbers. So, for every number we are assigning a code and we are storing that particular code when we are going to store information. So, EBCDIC is your extended binary coded decimal interchange code this is the extension of EBCDIC ASCII because what will happen in ASCII it is basically represented with 7 bit. So, in case of 7 bit we can go up to 128 different character because we are having total 128 representation from all 0s to all 1s. So, we can go up to 128 characters.

But in case of EBCDIC, EBCDIC it is a just extended by 1 more bit it is an 8 bit code so; that means, my character size is increased by 2. So, it is becoming now 256 in ASCII it is your 128 it is your EBCDIC 256, but if you look into the entire universe entire globe there are a lot of symbols we should not only think about that English alphabets we should not think about only that numerals and the signs like + - and like that, but there are several languages and in every languages they are having several character. Like if we talk about India we can talk about the de Devanagari script or Hindi language, so they are having several character. Now we have to give unique code to each and every character.

So, this is 1 language I am talking about in India Hindi, but we are having several languages like that if you go to the down south you are going to get kannadi, you are going to get Tamil, if you come to our streets you are going to get Assamese, you are going to get Bengali, if you go to north then you are going to get some Haryana, be some Marathi language also in the west like that. So, several languages are there and several characters are there this is I am talking about in India, but globally that languages is much more.

So, finally, we want to represent each and every symbol every character to computer and we need a bigger code with 8 bit or 7 bit we cannot do it. So, for that the concept of UNICODE is coming in to picture. A unique numbers provided for each character. So, if you want to difference some character in computer then you have to approach this body then by looking into the nature of the character they will keep a unique code to this particular symbol or particular character and that is it can be used in computer. So, to character representation now the standard is your UNICODE. So, this is the different way we can represent our information, but finally, everything will be converted to the binary code only and computer works on binary. So, these are the representation issues.

(Refer Slide Time: 74:23)

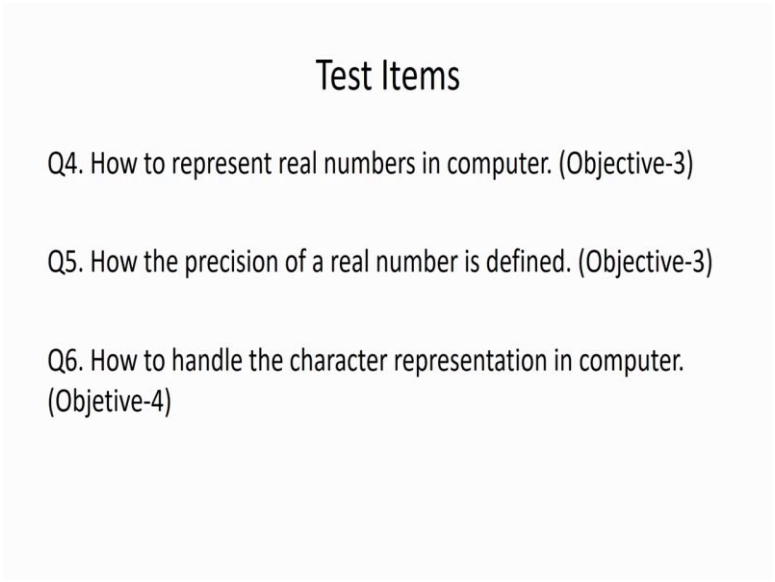


Now, just see I am giving some test items with respect to this particular representation information representation and your carry number system. Now the first test item I am giving that consider a decimal number 175 write the equivalent of this number in base 2, base 5 base 8 and base 16. So, if you can do this thing this is basically the objective 1 of this particular unit; that means, if you can solve this thing; that means, you can say that we have achieved the objective 1 of this particular module in it.

What are the different ways to represent integer in computer indicate their advantages and disadvantages. So, this is the objective 2 we are talking about the integer representation. So, already we have discussed these issues maybe this is your sign magnitude form one's complement form or two's complement form. The question 3, is it possible to handle integer of any limit in a particular computer, if not why? This is meeting objective 1 and 2, I think already I have mentioned in my lecture if I give you some tasks to work with pen and paper you can go for any limit, but in computer we are always restricted by a range.

Now why, why it is restricted by range now you just look into it I think I have mentioned in my lecture you just refer some books also and see why it is restricted.

(Refer Slide Time: 75:48)



Question 4 How to represent real numbers in computer? This is the meeting the objective 3, I think already have discussed basically you concentrate on IEEE format because this is a standard. How the precision of a real number is defined? Again we have discussed this things. So, basically number of bit pattern that we are using in the significant of mantissa depending on that we can say what is the precision. Question 6 How to handle the character representation in computer? This is the objective 4 of this particular unit already I have mentioned in my last slide.

So, here I can give you one more task or just to look into it what is the size of the UNICODE I am not mentioned over here deliberately I have not mentioned it, now you find out the information what is the number of bits that we use to represent any character in UNICODE and if there is any format is there is you look into the format also.

Ok so, these are the test item we have slightly looked into it and I hope that at least you are getting idea or having idea how we are going to represent information in computer and how we are going to work with this particular information. With that I wind up this particular lecture.

Thank you all.