

Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati
Fundamentals of Digital Computer

Lecture – 03
Information Representation and Number systems

Hello everybody, welcome back to the online course on Computer Organization and Architecture. Now we are in the module of fundamentals of digital computer now we are in unit 3 and unit 3 is basically related to information representation and number system. In this unit we are going to say how we are going to represent information for digital computer and the number system that you are going to use for our arithmetic.

So, first of all we are going to sort the objective, what is the unit objective of this particular module.

(Refer Slide Time: 00:59)

Module: Fundamental of Digital Computer

- Unit-3: Information Representation and number systems
- Unit Objectives:
 - Objective-1: Illustrate the number system of different radix system (Knowledge)
 - Objective-2: Describe the methods for integer representation (Comprehension)
 - Objective-3: Illustrate the method to represent real numbers (Comprehension)
 - Objective-4: Describe the representation of character (Knowledge)

So, the first objective is illustrate number system of different radix system. So, this is in the knowledge level. So, we are going to discuss everything in the knowledge level. So, that will have the knowledge about a number system and you can use these things in digital computer

Objective 2, describe the method for integer representation. So, this is in the comprehension level. So, when you see this things you will be able to understand how you are going to represent in this term in computer system. Objective 3, illustrate a method to represent the real numbers. This is also an comprehension level we are going to see how you are going to deal with real numbers in computer. And objective 4, describe the representation of character this is also in knowledge level we are going to see how you are going to represent character in computer because you have to deal it data processing and many a time you are going to work with string manipulation.

Now, first we are going to talk about the number system.

(Refer Slide Time: 02:01)

Decimal and Binary Numbers

- Consider the decimal number 75_{10}
 - $75_{10} = 7 \times 10^1 + 5 \times 10^0$
- Binary equivalent of 75_{10}
 - $75_{10} = 1001011_2$
 - $1001011_2 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$
- Octal equivalent of 75_{10}
 - $75_{10} = 113_8$
 - $113_8 = 1 \times 8^2 + 1 \times 8^1 + 3 \times 8^0$

I think all of you know about a decimal number system where the base of this number system is 10 or in general we say radix system. Consider an example of decimal number is 75 and I just ask you if you have written as 10, it means that a base of this number system is 10. So, now, 75 how you are going to evaluate the value of 75, it is 7 into 10 to the power 1 plus 5 into 10 to the power 0, 70 plus 5 this is 75.

Now, what is the binary equivalent of this particular 72? Now whatever we are going to do eventually we should get 75. So, I am not going to discuss about conversion just I am giving an example. So, 75 in decimal number system will be represented it 1 0 0 1 0 1 1 in binary number system. So, when we talk about a binary number system then the radix of base of the number system is 2 which is even its a subscript over here.

Now, in base 10 system it is of radix 10 and this is a multiple of 10, but in case of your binary number system it is multiple of 2. So, in that particular case now way we are say that fifth is the 0th position, 7 is the unit position of our decimal number system. So, similarly here also we are going to have this position this 1 is the 0th position like that 1 2 3 4 5 6. So, this 1 is in the 6th position. So, ultimately what is the equivalent value of this binary number? $1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$. So, this is 2^6 is equal to 64, 2^3 is 8, $64 + 8 = 72$, 2^1 is 2, $72 + 2 = 74$ and 2^0 is 1, so plus 1, 75. So, eventually we are getting the decimal equivalent 75. So, in this way we can represent every numbers every decimal numbers in binary representation.

Like that we are having different radix system and as for example, I am giving another base system which is your 8 base is 8. So, now if you consider this 75 in decimal number system that equivalent octal number system, base 8 number system is known as your octal number system it is 1 1 3 so that means, $1 \times 8^2 + 1 \times 8^1 + 3 \times 8^0$. So, 8^2 is 64 plus 8 64 plus 3 75. So, this is the way we can have a conversion from 1 number system to another number system.

Decimal to any other number system is very easy very simple what we can do, simply divide that number by that particular base. So, in that particular case 75, this is in radix 10 or base 10 or you want convert it to octal number system. Then what we are going to do? We will divide it by 8, $8 \times 9 = 72$ and remainder is 3 again we are going to divide 9 by 8, $8 \times 1 = 8$ and remainder is 1 again we are going to repair divided by 8 then result is 0 remainder is 1. So, in that particular case we are going to get 1 1 3 in octal number system. So, 75 in decimal number system is equivalent to 1 1 3 in octal number system. This is the way we can represent or we can convert every decimal number to any other radix system. So, similarly when we are going for a binary number system than 75 will be divided by 2 and in that particular case now where you are divided consider this is coming in the unit position and like that it will increase. So, it is 1 1 3.

(Refer Slide Time: 06:22)

Decimal and Binary Numbers

- Consider the decimal number 75_{10}
 - $75_{10} = 7 \times 10^1 + 5 \times 10^0$
 - $1001011_2 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$
 - $113_8 = 1 \times 8^2 + 1 \times 8^1 + 3 \times 8^0$
- Least significant bit and Most Significant Bit
- Number of symbols for a particular number system

Handwritten notes:
decimal: 10
0, 1, 2, ..., 7, 8, 9
MSB
LSB
Bit
Hexadecimal
16
0 & 1
 $D_{16} = 13 \times 16^2 + 1 \times 16 + 2 \times 16^0$

So, now when we are going to talk about a number system we are having 2 terms call least significant bit and most significant bit MSB and LSB this is basically when you are going to talk about that bit it basically talk about the binary number system you are going to take single bit. But for any radix system we can say it is a most significant digit and least significant digit. As for example, in your 1 1 3 in best 8, 3 is the least significant digit and 1 is the most significant digit in this particular case because you just see that this 1 is multiplied by 8 square. So, weightage is more and these 3 is multiplied by 8 to the power 0 which is one. So, weightage is less, so in that case we having that least significant digit as 3 and most significant digit as 1 for 1 1 3 in octal number system. Similarly when we are going to talk about the binary number system which say it is least significant bit and most significant bit.

So, in this particular case the 1 or bit that we are having a most right hand side is known as your least significant bit because it is weight at least 2 to the power 0 and the 1 that we are having in the left hand side most left hand side is your most significant bit because the weightage is more over here 2 to the power 6. So, this is the conjunction about the least significant bit and most significant bit.

Now, how many numbers or how many symbols we required for a particular number system. So, it is basically related to the base of the number system. So, if we are using a decimal number system decimal number system in that case base is 10. So, to represent

these things we need 10 different symbol and we know that those symbol we are using 0 1 2 3 like up to 9 we are using. So, these are the 10 different symbol that you are using in decimal number system. So, in case of binary number system since base is 2 we need only 2 symbols and that 2 symbols that we are using in binary number system is a 0 and 1. So, the binary number is represented with the help of 0s and 1 only.

When you come to octal number system in that particular case the base is 8. So, we need 8 different symbol. So, what are the symbols that we are using? We are using from 0 to 7 these are the 8 different symbol we are using in octal number system. Like that we can go for any radix or any base. One important number system that we are having is your hexadecimal. So, in computer system sometimes we represent our information in hexadecimal form. So, in hexadecimal the base is 16, so we need 16 different symbols. So, in hexadecimal system the base is 16 and mean it 16 different symbol to represent it. Now, what are those 16 different symbols that we use in our hexadecimal number system?

So, from 0 to 9 we are using those symbol to represent 0 to 9, 0 1 to like that since it is having base 16. So, we can represent 16 different numbers in this particular number system. So, 0 to 9 10 is gone now you are having 6 more number 10 11 12 13 14 and 15. So, to represent those number in hexadecimal system we use the letter from an alphabet and these letters are you A B C D E and F. So, this is the symbols that we use in our hexadecimal number system, 0 to 9 10 digit and A to F 6 characters, total 16 character.

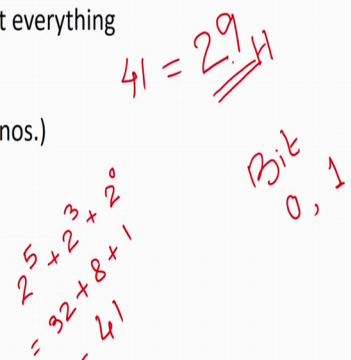
So, if we write some numbers like that cell D 12. So, this is an hexadecimal number and we are using (Refer Time: 10:38). What is the decimal equivalent of these things? This is your d is your 10 11 12 13. So, this is 13 into 16 square plus 1 into 16 to the power 1 plus 2 into 16 to the power 0. So, whatever you are going to get this is the decimal equivalent of this particular hexadecimal number. So, conversion from any number system to decimal number system is very easy. So, like that from decimal also we can convert to any number system. So, as defined by the base of that number. So, this is the way we can just represent our numbers.

Now, we are going to see how you are going to represent integers. So, this is 0s and 1s only. So, we will say whatever you are representing will say it is a bit and bit is nothing, but binary digit; that means, you are representing 0 or 1. So, these are the binary digit.

(Refer Slide Time: 11:33)

Integer Representation

- Bit: Binary digit
- Only have 0 & 1 to represent everything
- numbers stored in binary
 - e.g. 41 = 0010 1001
- No minus sign (for negative nos.)
- No period (for real nos.)
- Range of numbers
- Negative Numbers
 - Sign-Magnitude
 - Two's complement



And already we have seen how you are going to write numbers in binary number system. So, for example, here we are giving 41. So, in that particular case, 41 is having binary equivalent 0 0 1 0 1 0 0 1 so that means, we can say that this is these are the 3 ones are coming. So, these are going to have going to give has the effect. So, it is basically 2 to the power 5 plus 2 cube plus 2 to the power 0 0 1 2 3 4 5. So, 2 to the power 5 is 32, 2 cube is a 8 plus 2 to the power 0 is one. So, this is equal to 41. So, this is the way we are going to represent our integer in binary number system.

So, there are some issues now you have to see how to give the minus sign to have the negative numbers, how to put a decimal point for real numbers and secondly, you have to see what is the range of numbers. So, basically say if I ask you to do something with pen and paper then you do not have any limitation, but when you are going to work to our digital system or you said at you are going to work with digital computer then always you are restricted by a ranks and these ranks depends on the bit number of bit that you are using to represent the number.

So, in this particular case of 41, in this particular case we are using 8 bits. So, when we are using 8 bits; that means, we are restricted by a number up to a particular (Refer Time: 13:24) you can go. So, if we increase the number of bit then you can go for more ranges and. Secondly, we have are going to see how you are going to represent a negative

numbers there are been 2 way of doing it 1 is sign magnitude and second one is your two's compliment as an example I am giving here is an size of my data.

(Refer Slide Time: 13:44)

Information Representation

- Size of Data:

SIZE	BINARY
8	0000 0000 1111 1111
12	0000 0000 0000 1111 1111 1111
16	0000 0000 0000 0000 1111 1111 1111 1111
20	0000 0000 0000 0000 0000 1111 1111 1111 1111 1111
32	00000000 11111111

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

$$128 \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

$$255$$

$$0 - 255$$

So, if you are walking with your size A; that means, you are working with the data size of 8 bit. So, in that particular case the combination may be all 0s to all 1s. So, in that particular case all ones is going to represent can you calculate it this is your 0 to 7; that means, 2 to the power 7 plus 2 to the power 6 plus 2 to the power 5 plus 2 to the power 4 plus 2 cube plus 2 square plus 2 to the power 1 plus 2 to the power 0. So, this is your 1 2 4 8 16 32 64 – 128. We have to wait he have to a custom in this particular 2 to the power something because many a time we are going to look it. So, this is I am going to get 128 64 32 16 8 4 2 1 and if I add a all those thing I am going to get 255. So, 1 you have to work with 8 bit numbers then my range will go from 0 to 255.

Similarly if I am using a 12 bit numbers now my range is going to increase like that if I am going for 32 bit numbers; that means, you are going to work with 32 bit at a time. So, nowadays is to said my computer my processor is a 32 bit processor; that means, I can work with a 32 bit numbers. I say that my computer is or my processor is a 64 bit processor then we can handle 64 bit at a time.

(Refer Slide Time: 15:27)

Information Representation

- Size of Data:

SIZE	BINARY	DECIMAL
8	0000 0000 1111 1111	0 - 255
12	0000 0000 0000 1111 1111 1111	0 - 4095
16	0000 0000 0000 0000 1111 1111 1111 1111	0 - ($2^{16} - 1$)
20	0000 0000 0000 0000 0000 1111 1111 1111 1111 1111	0 - ($2^{20} - 1$)
32	00000000 11111111	0 - ($2^{32} - 1$)

$2^8 = 256 - 1$

So, in that particular case we can say that these are binary representation for 8 bit this is the minimum number this is the maximum number if you are going to work with a positive line only and if this is 32 bit this is the minimum 1 this is the maximum one. So, my range is in decimal is of 8 bit numbers it is from 0 to 255 for 12 bit number it will 0 to 4 0 9 5. So, this is 4 0 9 5 and now I am going for 16 bit number it is 0 to 2 to the power 16 minus 1 basically 255 is nothing, but I can say 2 to the power 8 minus 1 2 to the power 8 is a 256 minus 1, 255 and if it is your 20 bit then into go up to 2 to the power 20 minus 1 and in case of 32 bit it will go to 2 to the power 32 minus 1.

(Refer Slide Time: 16:19)

Information Representation

- Size of Data:

SIZE	BINARY	DEC	HEXA
8	0000 0000 <u>1111 1111</u>	0 - 255	00 - FF
12	0000 0000 0000 1111 1111 1111	0 - 4095	000 - FFF
16	0000 0000 0000 0000 1111 1111 1111 1111	0 - ($2^{16} - 1$)	0000 - FFFF
20	0000 0000 0000 0000 0000 1111 1111 1111 1111 1111	0 - ($2^{20} - 1$)	00000 - FFFFF
32	00000000 11111111	0 - ($2^{32} - 1$)	00000000 - FFFFFFFF

$4\text{-bit} = 0$
 $0000 \dots 1111 = 15$
 $15 \times 16 + 15 = 255$
 $= 255$

So, the similar information I am representing in another number system which is a hexadecimal. Now, you just say what I am saying this is your 8 bit number all 0s to all ones. So, all 1 in decimal it becomes 255 in hexadecimal it is your FF FF means 15 F is represents 15, 15 into 16 to the power 1 plus 15 into 16 2 the power 0. So, in that particular case you will get that this is nothing, but 255 in decimal.

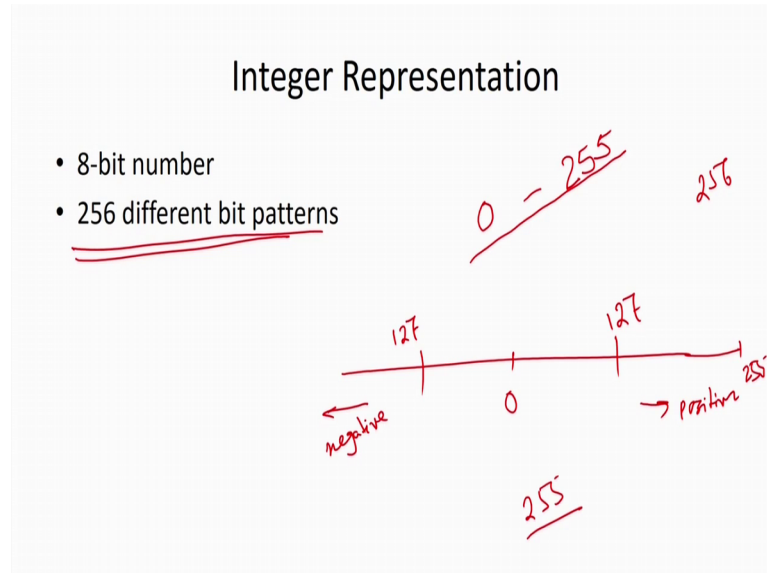
Now, when we are going to discuss that representation of number in computer or in your binary system of a binary digit most of a time we go we are going to take help of hexadecimal number system because it is helping on from days. So, when I am going to work with 32 bits number I have to write 32 bits all 32 bits 0 or maybe combination on 0s and 1 which is slightly difficult. Similarly when you are going to work with 8 bits you have going to write 8 bits all 0s to all 1. Now, again if I see you how to do it in your decimal number then we have to have slight your distance 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 done this is your 0 and if it is all 1 you can say that 8 plus 4 12, 12 plus 2 14, 14 plus 1 15. So, this is your 15 se 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, while you having a binary representation we simply group them in the 4 bits. So, in that particular case what will happen? We are going to set a 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 symbol needed for hexadecimal representation.

So, if I go back to 1 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 to 1001 this is basically 8 plus 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 0010 1001. 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 2 symbol 2 digits. So, in most of a time we are going to represent our information in hexadecimal for understandability and for reliability, but 1 you think about how it is working in the computer basically it

working with this particular binary digit only in bit pattern 0s and 1, but for readability we can write something in a hexadecimal notation.

(Refer Slide Time: 19:59)



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

So, if you are going to represent only positive number 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 numbers also this is the number 1 this is 0 and this is your positive side and this is your negative side. So, when you are going to work with a negative number then what will happen whatever 256 different bit patterns you are having some of the bit patterns need to be deserve for negative numbers also; that means, in positive number when you 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 plus 127 254 with 0 to 255. So, we are having total 256 symbols, but here we are using 255 one more symbols

are still remaining. We will see how you 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 you are going to handle negative numbers then range will reduce it will go from some negative 127 to positive 127 or may be plus minus 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)

8: 1 7
151 Magnitude

16 1 15

127
+127
-127
254

00000000
10000000

Now, for that we are having 2 ways of representing in 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? So, for 8 bit numbers 1 bit will go from your sign and 7 bit will go for magnitude. So, if it is a 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 happen what convention we are using 0 means the positive numbers and name 1 means the negative number. So, in that particular case for example, you consider plus 18. So, this is the magnitude 1 0 0 1 0 this is 2 to the power 4 plus 2 to the power 1 16 plus 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 plus 18 and this bit more significant bit is 1 1 represent the negative numbers. So, this

bit are going to give me minus 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, which 0 I can go up to plus 127 and with most significant bit 1 we can go up to 127. So, this is 127 plus 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 depth valued numbers is 255 we are having 2 representation of 0. So, in a look to eliminate this also because why we should use 2 bit pattern to represent the same number, so for that we may go for some other representation which is known as a two's compliment.

(Refer Slide Time: 25:03)

Two's Compliment

<ul style="list-style-type: none"> • +3 = 00000011 • +2 = 00000010 • +1 = 00000001 • +0 = 00000000 • -1 = 11111111 • -2 = 11111110 • -3 = 11111101 	$\begin{array}{r} 11111100 \\ 11111101 \\ \hline 255 \\ 254 \\ 253 \\ 252 \\ \vdots \end{array}$	$\begin{array}{r} 0 \\ 1 \\ 2 \\ 3 \\ \vdots \end{array} \begin{array}{l} -1 \\ -2 \\ -3 \\ -4 \\ \vdots \end{array}$
---	--	---

So, in this two's compliment just I am giving 1 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 plus 1 is simple that magnitude 1 1 0 1 1 like that if I go for positive 4 then it will be 1 0 0 like that and negative number is coming as minus 1 is representing

as your all 1s then minus 2 is coming as your all 1 and 0 and minus 3 is coming as your all 1 after; that means, 6 1 0 1.

So, basically what happens? It is basically the maximum number is a 255 or 8 bit representation that 255 is going to represent my minus 1 now you just come down and go down was then 54 is going to give me minus 2 253 will give me your minus 3 then 252 will give me minus 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 compliment form you are going to represent in this particular way.

(Refer Slide Time: 26:31)

Two's Complement

- 1's complement and 2's complement

$$\begin{array}{r}
 00000011 \\
 11111100 \\
 \hline
 11111101 \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 11 \\
 + 1 \\
 \hline
 100
 \end{array}$$

$$\begin{array}{r}
 37 \\
 63 \\
 \hline
 100
 \end{array}$$

$$\begin{array}{r}
 37 \\
 + 63 \\
 \hline
 100
 \end{array}$$

Now, how to do it? You have been 1 is called ones complement and two's complement. So, if I consider a particular number say 0 0 0 0 0 0 1 1 this is a number that I am considering then what is the ones complement ones complement basically it says that you complement all the bits of this particular number. So, I am going to get 1 1 1 1 1 1 0 0. So, this is the one's complement of this particular number. And what is two's complement you change that in case of two's complement what about you are getting in 1 complements at 1 to it. So, what I am going to get 1 1 1 1 1 1 0 1. 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 nothing in digital system we can do it by using a non gate

or maybe using a not gate that we have discussed these things all 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 taking about we are taking this particular number one. So, if I am going to take the ones compliment I am going to get 1 1 1 1 1 0 0 now if I add 1 to it then I am going to get 1 1 1 1 1 0 1. Now, you just see whatever I am getting or taking the number after looking into the two's compliment I am getting this representation.

So, generally you are saying that this is my positive 3 and other one is my negative 3. So, after taking the ones two's compliment we are going to get the negative representation of the given number. So, this is the trust that we are going to set and why we are coming this particular number system and why we call it compliments you just see that I am taking this particular number 0 0 0 0 0 1 1. Now, I am taking the two's compliment what I am getting 1 1 1 1 1 0 1. Now, in that particular cases I am going to add this 2 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, you are getting this number.

Now, what will happen we are working with 8 bit numbers; that means, you are having provision to store this particular 8 bit something 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 one. So, finally, what I am getting the after adding this number we are getting 0. So, when we add 2 numbers and if the result is 0 then what we say that 1 is the negation of the other because if I add 2 numbers a plus minus a then we are going to get result is 0. So, this is the principle that we are using. So, in two's compliment form we can use we get the negative of a given number or if you are giving a negative number then you are going to get the positive of that particular number.

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

Now, you add a 37 and 63 what you are going to get 0 9 10. Now, you are working with 2 digit only, now we are having working with 8 bits we are using 2 bit 2 digit only. So, this digit cannot be stored. So, eventually what will happen heading this 2 number is going to give me the result 0 so; that means, we can say that 1 is the regression of the other if we are using the number system about 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 a 9's complement the way we are getting the ones complement this is the 9's compliments.

And after adding 1 we are going to get the tens complement. So, this is the principle we are using to represent a negative number. So, already I have explain 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 = 0000011	
– Boolean complement gives	1111100
– Add 1 to LSB	1111101

Now, what already have send?

(Refer Slide Time: 32:27)

Negation Special Case 1

- 0 = 00000000
- Bitwise not 11111111
- Add 1 to LSB +1
- Result 100000000
- Carry-out is ignored, so:
- -0 = 0 ✓

*-127 to +127
2 representation
of 256*

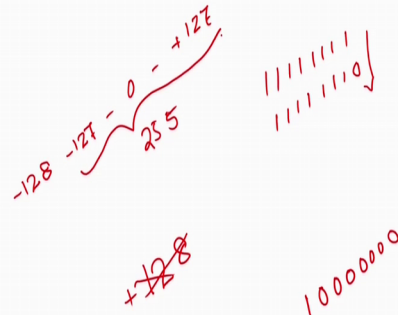
This is a special case already you have seen that now in that particular case negative 0 is equal to a positive 0; that means, you having only 1 representation of 0. So, in sign magnitude form what will happen we are having minus 127 to plus 127 total 256 number along with that 2 representation of 0. So, total 256 symbols.

Now here in this particular case after adding it since you are going to discuss this particular carry out with because you cannot store in to the you walking with 8 bit number. So, finally, positive 0 and negative 0 is becoming same. So, we are having only 1 representation of 0.

(Refer Slide Time: 33:14)

Negation Special Case 2

- ~~-128 =~~ 10000000
- bitwise not 01111111
- Add 1 to LSB +1
- Result 10000000
- So:
- $-(-128) = -128$ X
- Monitor MSB (sign bit)



Other one is you just see what will happen now in that particular case if I am going to look for this particular combination because it is coming do not know what I am saying that you start with all 1 then 1 1 1 1 1 1 0 like that you decrease it and finally, we are going to get this particular number. So, for this particular number if I take the one's complement then I am going to get this representation and 2's complementation I am going to get this particular number. So, basically two's compliments will give me the negation of that particular number. So, if this is a particular representation then we should be the negation of this thing, but both are standing up to be same. So, this is a special case and you are going to handle it as like that this representation will be treated as your minus 127 oh sorry minus 128.

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

Since this bit pattern is starting with one. So, if we going to treat this as a negative numbers and we say that it is going to represent minus 128 not plus 128; that means, now my range is go from minus 128 to plus 127 total 256 different representation. So, if you 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 is why when we are coming for this particular special case 1 all 0s that will represent a negative number the negative number is your minus 128. Now, what is the range of number?

(Refer Slide Time: 35:54)

Range of Numbers

- 8 bit 2s compliment
 - +127 = 01111111 = $2^7 - 1$
 - -128 = 10000000 = -2^7
- 16 bit 2s compliment
 - +32767 = 01111111 11111111 = $2^{15} - 1$
 - -32768 = 100000000 00000000 = -2^{15}

(-2ⁿ⁻¹) to (2ⁿ⁻¹ - 1)

0 - 255

n bit

Already I have mentioned it. So, in that particular case, for integer sorry for positive number we can go from 0 to 256 255 if it is an 8 bit number. So, for negative numbers if you are going to handle a complete range of integers positive and negative then for 8 bit numbers the range will go from minus 128 to plus 127.

And for 16 bit number it will go from 32768 to 32767. So, minus 32768 to plus 32767 and basically it is in 2 to the power 15 minus 1 this is 2 to the power 7 minus 1 because you 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 to the power n minus 1 to minus 2 to the power 15. So, if you are going to work with n bit number then my range will be your minus 2 to the power n minus 1 to 2 to the power n minus 1 minus 1. So, this is the range that you are going to have when you are going to use two's compliment 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 minus 1 will represent at the all 1 and this decremented and finally, minus 8 will be your 1 0 0.

(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

Now, look for some operations say I am giving that 7 plus 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 0 1 1 1, 0 1 1 1. So, this is 7 and 7 now if I add them then what I am going

to get 0 1 1 1 this is the things that I am getting. Now you just say that minus 7 plus minus 7 now what is the representation of minus 7 is two's complementation 1 0 0 1, 1 0 0 1. So, this is your 1 0 0 1 and 1 0 0 1. So, if I add them what I am going to get 0 1 0 0 1. So, this is a carry out we cannot consider it because we are working with 4 bit number.

Now, considered that 7 plus minus 1, 7 is your 0 1 1 1 and what is minus 1, minus 1 is this all 1 1 1 1 1, now this is your 0 1 1 0 1 again I cannot considered these things and 1 plus minus 7. So, 1 is your 0 0 0 1 and what is my minus 7, 1 0 0 1 1 0 0 1. So, this is your 0 1 0 1. Now, you just see what we are getting actually. So, in this particular case this is your plus 7 this is your plus 7 now you are working with two's compliment. So, 1 1 1 0 what does it means 1 1 1 0 is your minus 2 here this is your minus 7 minus 7, now 0 0 1 0 what I am getting 2 these are decimal equivalent this is your 7 and this is your minus 1.

So, what I am getting 0 1 1 0 which is your 6 this is your 1 and this is minus 7. So, what I am getting 1 0 1 0. So, what is that 1 0 1 0 is your minus 6. So, I am getting minus 6. So, just I am calculating it just show it now what will happens say when I use minus 7 to minus 1 should get minus 6 and I am getting it. So, this is a correct result when I am using this things minus 7 and minus 1 I am getting plus 6. So, this is also correct result, but when I am using minus 7 plus minus 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 plus 7 and plus 7 I should get plus 14, but what I am getting over here I am getting minus 2.

So, in this particular case, whatever result we are getting these are not correct because we have walking with 4 bit numbers and you can handle it 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 minus 8 to plus 7. So, this is the range, but these are the valid numbers after adding I am getting a bigger number if it is a positive lines say which is your 14 no doubt about it, 8 plus 4 plus 2, but since you are going to handle negative number. So, you can actually it is represented as minus 2. So, these are the 2 result at I am getting which is not correct and this situation is basically known as overflow because we cannot handle plus 14 over here this is the range. So, this is your overflowing the number. So, this is the overflow situation, but in this 2 cases I do not have any problem and what I am getting that correct resultant these are the 2 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 a digital system is doing it and essentially it is giving 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 4 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 with this carry out is 0. So, in that particular case what will say that these 2 operation generating some carry for me, but one is your correct result second one is not correct result. So, we are having some situation call, but 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 is a see that I have give a, one more question that you perform 7 minus 5. So, in that particular case how I am going to perform these operation 7 minus 5 I know that this can be done with your 7 plus minus 5. What is 7? A binary representation 0 0 0 1. And what is the representation of minus 5? 1 0 1 1, 1 0 1 1. If you add these 2 things then what you are going to get 0 1 0 0 1 7 minus 5 is 2. So, I am getting 5 only. So, this is your 7 and this is your minus 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 address architecture 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 you are using two's complement representation basically advertising it can be used to evaluate the subtraction also. So, in this particular case 7 minus 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 caring to the most significant bit and this is the carry out this is the carry into the most significant bit and this is the

carry out. So, these are overflow situation this is the carry into the most significant bit and this is the carry out of the operation. Here also this is the carry into the most significant bit and this is the final carry out now you just see in this 2 situation these are my correct result. Here also I am having now the observation is like that if the carry into the most significant bit and the final carry out if they are same then we do not have any overflow say these are the same situation.

These are the same situation. So, we do not have overflow, but in case of overflow these 2 bit 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 2 situation is 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 over flow 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.

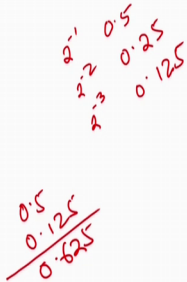
So, these are the situations. So, what I am saying that here I am talking about overflow. So, this is the way I can detect the overflow. I have seen whether it has generated carry or not. So, with this carry out bit I can check whether carry has been generated or not and sign bit what is the sign of the result basically this most significant bit is going to give me the sign bit. So, I will just look into this particular position and say what is a number is positive or negative. So, since it is 1, it is a negative number and this going to represent minus 2 since this bit is 0. So, it is going to represent positive numbers. So, this is plus 2. So, like that these are the 3 information you 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 to you 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?



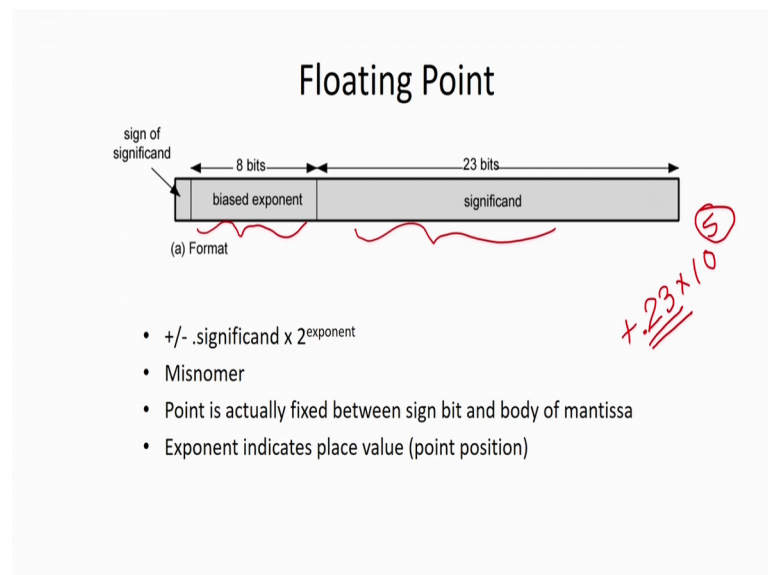
So, how you are going to represent real number it is again everything we have to deal with your binary number system only 0s and 1 and you should know what is the decimal equivalent of a real numbers if you are going to represent it in your binary number system.

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 you are going to get it this number is your 2 to the power 4 plus 2 to the power 0 and this are decimals. So, this is plus 2 to the power minus 1, 2 to the power minus 2, 2 to the power minus 3 and 2 to the power minus 4. So, this is your 4 plus 1 this is 9. So, in now 1 0 0 1 is 9 and 2 to the power minus 1 is your 0.5, this is your 2 to the power minus 1 what is your 2, to the power minus 2 this is your 0.25 half of this, 1 2 to the power minus 3 this is half of this, so 0.125. So, this is basically 0.5 plus 0.125. So, this is your 0.625. So, finally, my number is you 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 minus 7 point something to minus 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 these things we will say this is a fixed point we have found 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 is exhausted 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.

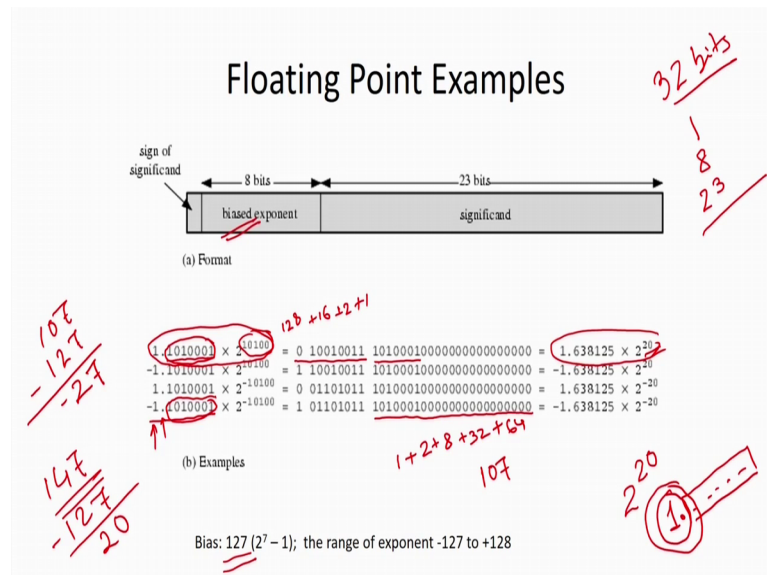
(Refer Slide Time: 50:44)



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 bisector of exponents we will see and we are having the significant part; that means, if I am going to represent in decimal number systems say plus 23 into 10 to the power 5 then what will happen, this is the exponent part and this is the significant part or the man mantissa part. So, this is the number presentation if they having plus minus then decimal point sorry here decimal point will put it over here and into 2 to the 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.

(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 you look into this number representation say 1.10100001 into 2 to the power 1 0 1 0 0. So, if this is the number that we want to represent in our floating point representation then what will happen, in that particular case we have to see what is the significant part this significant part is 1 0 1 0 0 0 1, 1 0 1 0 0 0 1, so 1 0 0 1 0 0 0 1. So, this is the significant part and it is having total 23 bit. 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 plus 4, 20. So, basically we are talking about 2 to the power 20. So, this is the representation 2 to the power 20 and if I convert it if you come like that in decimal equivalent the (Refer Time: 53:25).

Now, in this particular case we are having 2 to the power 20, but what you have stored over here you just see what is this equivalent 1 2. So, basically this is your 1 2 4 8 16 plus 32 64 128. So, finally, this is the number that you are getting, so 128 plus 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 bias 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 bias buy some numbers. So, in this representation it is biased by 128; that means, whatever number you are storing over here that will be that to find out the exact exponent that 127 will be subtracted from that. So, 147 minus 127 what I am getting this is 20, so this is the 20 that we are getting. So, this is the bias exponent.

Now, just look for the negative now this is a negative exponents this is your minus 20, but for minus 20 what we are storing 1 plus 2 plus 8, 8 plus 16 plus 32 plus 64. So, these are the things that we are storing in that exponent. So, what I am getting 96 plus 8 1 0 4 5 6 1 0 7.

Now, we are storing 1 0 7 in the bias exponent. So, what is the exact value we have storing. So, what we have to do we subtract 127 from 107, 107 minus 127, so we are going to get minus 20. So, this is the way we are storing a floating point numbers. So, it is having 3 component, one is your sign bit then bias exponent and significant and what we are storing you just see observe these things you want to store 1.10100001 and we have store this particular information in the significant 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 significant that will be offended by 1 point that numbers. So, this is 1 point something then what happen you 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 you 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 are will be stored in your 2's complement form exponent will be stored in your bias exponent. So, this is basically it will be stored in a bias exponent and after that what happen some values we need to subtract it may be from 128 or 127 for 8 bit number.

(Refer Slide Time: 56:48)

Signs for Floating Point

- Mantissa is stored in 2s compliment
- Exponent is in excess or biased notation
 - e.g. Excess (bias) 128 means
 - 8 bit exponent field
 - Pure value range 0-255
 - Subtract 128 to get correct value
 - Range -128 to +127

So, we must understand this 3 component while you 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 into 2 to the power 20 basically in binary representation we have going to have these things. So, for that this part we are going to put in the mantissa or the significant 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 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 you 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)

Normalization

- FP numbers are usually normalized
- i.e. exponent is adjusted so that leading bit (MSB) of mantissa is 1
- Since it is always 1 there is no need to store it
- (c.f. Scientific notation where numbers are normalized to give a single digit before the decimal point
- e.g. 5.123×10^3)

$$\begin{array}{l} 5123 \\ 5.123 \times 10^3 \end{array}$$

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 m s b 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 into 10 cube. 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 I should be in the bias form because why you are teaching the bias notation you just about the negative numbers represent store negative numbers in the exponent point.

(Refer Slide Time: 59:13)

FP Ranges

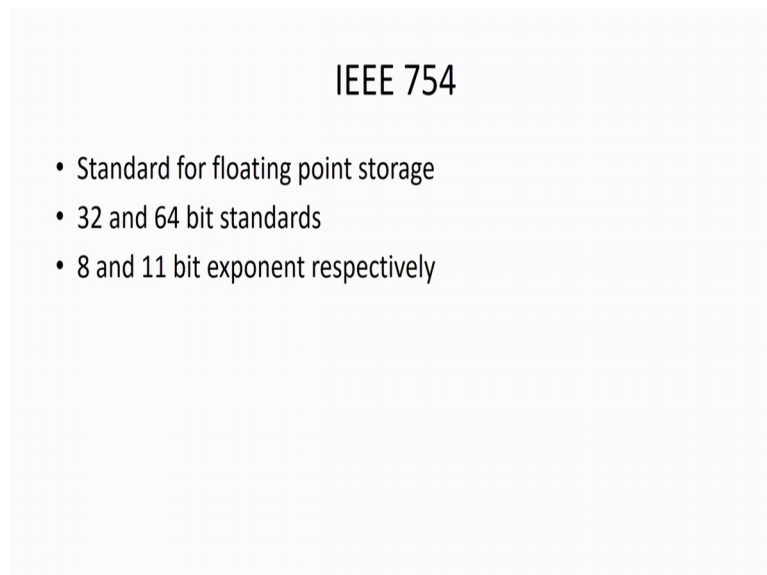
- For a 32 bit number
 - 8 bit exponent
 - +/- $2^{256} \approx 1.5 \times 10^{77}$
- Accuracy
 - The effect of changing lsb of mantissa
 - 23 bit mantissa $2^{-23} \approx 1.2 \times 10^{-7}$
 - About 6 decimal places

Now, what are the ranges of 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 exponents. So, if the exponents is 8 bit; that means, you can go up to 2 to the power 256 so; that means, you can go up to 2 to the power 256; that means, if I going to convert it to the decimal number you will find that you can go up to 10 to the power 77. So, if it is a 32 bit representation and we are going to use 8 bit for exponent then what you are going to get you have going to get range up to 10 to the power 77.

Now, what is the accuracy? And accuracy basically talk about what is the changes of the value if you are going to change the least significant bit. So, in that particular case we are using 23 bit you our mantissa, but when you 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. So, up to 20 3 bit you cannot stored it, but 20 4 bit you cannot store so that what is the changes of the number if you change this particular least significant bit and it will go from 0 to 1 or 1 to 0. So, that is why the accuracy is your 2 to the power minus 23; that means, in decimal number it is your 10 to the power minus 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 you increase the number of bits that range will increase as well as accuracy 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 instead 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)



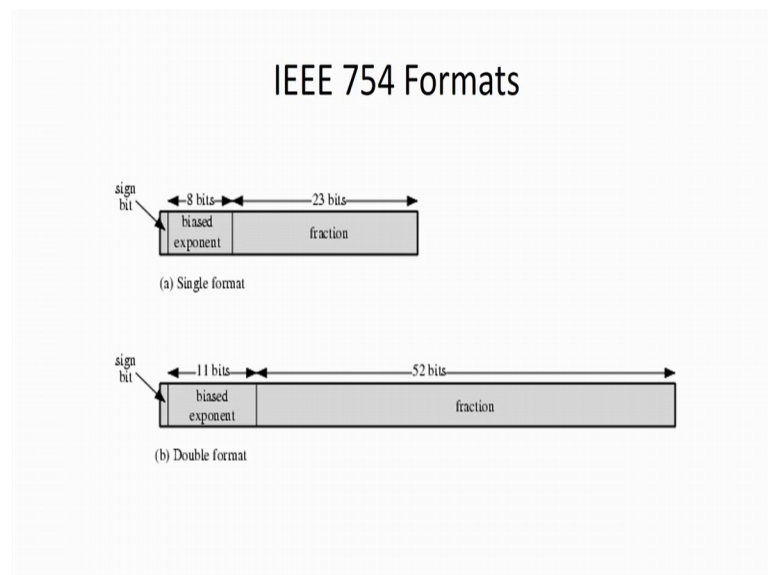
IEEE 754

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively

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 2 format one is your 32 bit and another one is a 64 bit because if you 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 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 to the power 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 you 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 you are increasing from 23 bit to around 54 bit it seems because 64 minus 11 is your 53 minus 1 is for sign bit. So, 52 we are using 52 bits for the mantissa part or significant part; that means, our accuracy is now going to increase for 22 bit you are getting the up to 10 to the power minus 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)

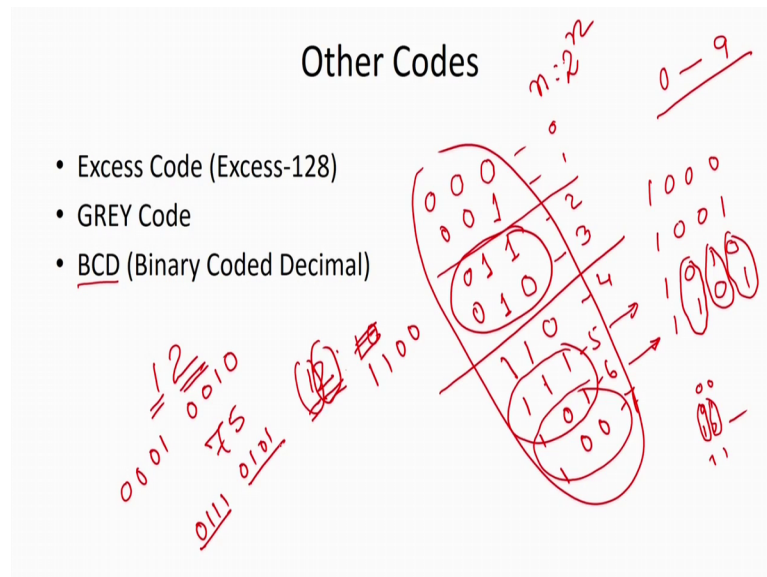


Now, this is the format, IEEE 754 format, this is your 32 bit format 1 bit is your sign bit 8 bit is for bias exponent and 23 bit is for fraction. Similarly for your double format this is your 64 bit. So, already I said that fractional part is your 32 52 bit 11 bit is your bias exponent and 1 is for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

So, that means, when you 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 on another case we can use 32 bit to represent our number. So, this is the format of representing floating point number. So, you have seen how to represent numbers basically it is either integer number or real

numbers. So, in case of real numbers then will we use floating point representation. Thus some other code also there. So, it is a binary code. Already you 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 you 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 this information I am giving it. So, what will happened this is a code that allowed. So, that 2 consecutive number is going to have that query 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 conjugative number you just see that here is a range of bit pattern in 3 position. In some cases it may be more also if you are going to handle look for more number of this. So, GREY code is consists just to minimize this particular ranges 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, 0 0 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 0 0 0 1 1 0 when I am going to form 1 to 2 there is a range of bit position in 2 different position, but here we want to minimize it. So, we want to, say that there will be ranges. So, we want 1 position then what we are going to do we take the mirror image of this things 1 and the most significant bit we have going to get 1 over it because already 2 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 ranging. So, here 2 is 1 0 and 3 is 1 1 in binary number, but in grey code it is different.

Now, when I go for more numbers then what will happens say 3 bits number then what I am going to get take the mirror image of these things that this one. So, I am going to get 1 0. So, this is 1 0. So, this 1 will become 0 will be come 1 now. So, this is range in 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 0 1 1 and this is your 0 0 1. 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 changes of bit on we are looking for 2 conjugative number you just see that if you take these 2 consecutive numbers this change is only 1 bit position if you take this 2 consecutive number this change is only bit position. If you 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 accustom 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, if I am going to have 12 then bit pattern is your 1 0 sorry bit pattern is 1 1 0 0 8 plus 4 12. But in binary coded decimal what we are going to do digit wise you are going to convert to the binary and for that for every digit we are having total 10 symbol 0 2 9 and to represent 10 symbol we need at least 4 bit 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 to the power n different representation already have mention it. So, with 3 bit I can go up to 2 to the power 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 0 0 0 1 and 2 be represented by 0 0 1 0. Like that if I am going to represent 75 in your BCD then will be your 0 1 1 1 and 5 is your 0 1 0 1, so 7 5. So, this is the binary coded decimal.

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

(Refer Slide Time: 70:25)

Character Representation

- ASCII (American Standard Code for Information Interchange)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- UNICODE (A unique number is provided for each character)



Now, in computers we have to work with the character also because in numbers you are doing some arithmetic operation, but sometimes you have to work with number system because now you just see that in every where 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 you have saying that character representation saying that character c h a r a like that, but inside computer when you are storing it we cannot store c as it is everything has to be stored as a binary number not only binary numbers 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 code 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 develop 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 1. 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 on think about that English alphabets we should not think about only that numerals and the sign as like plus minus and like that, but there are several languages and in every language there are having several character. Like if you 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 (Refer Time: 73:08), 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 language much more.

So, finally, we want to represent each and every symbol every character through 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 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 not 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)

Test Items

Q1. Consider the decimal number 175. Write the equivalent of this number in base-2, base-5, base-8 and base-16. (Objective-1)

Q2. What are the different ways to represent integers in computer. Indicate their advantages and disadvantages. (Objective-2)

Q3. Is it possible to handle integers of any limit in a particular computer. If not, why. (Objective 1 & 2)

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 6 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 receive the objective 1 of this particular module in it.

What are the different ways to represent integer in computer indicate that advantages and disadvantages. So, this is the objective 2 we are talking about the integer representation. So, already have discuss those issues maybe this is your sign magnitude form ones complement, form two's compliment. 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 my legs are if I give you some tasks to work to with pen and paper you can go for any limit, but in computer we are always restricted by runs.

Now why, why it is restricted by runs 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)

Test Items

Q4. How to represent real numbers in computer. (Objective-3)

Q5. How the precision of a real number is defined. (Objective-3)

Q6. How to handle the character representation in computer.
(Objective-4)

Question 4 How to represent real numbers in computer? This is the meet in 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 discuss 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 line 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 mention about here deliberately I have not mention 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 right to look into the format also.

So, these are the test item we are slightly look into it and I hope that at least your getting idea I having idea how we are going to represent a information in computer and how we are going to work with this particular information. With that I end with this particular lecture.

Thank you all.