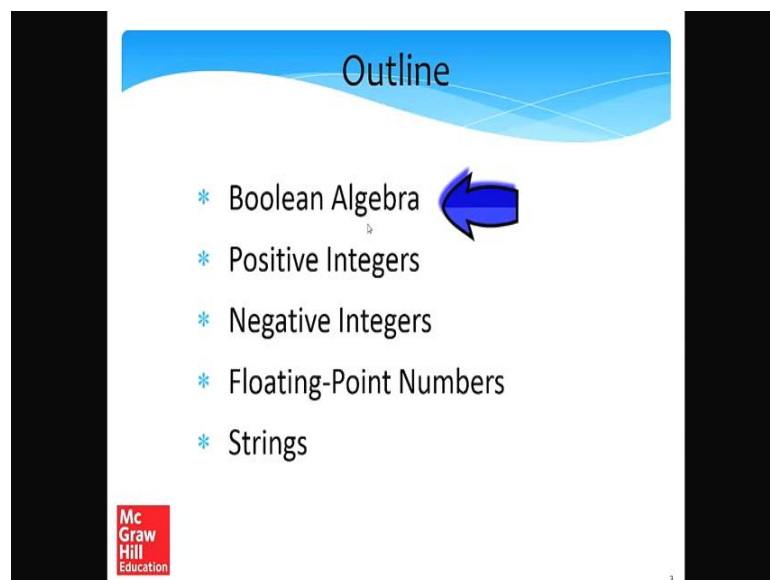


Computer Architecture
Prof. Smruti Ranjan Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture - 02
The Language of Bits Part- I

Welcome to Chapter 2: The Language of Bits. In this chapter we will study everything about bits which is the basic 0 or 1, and what they can achieve. Again these slides are part of the book; computer organization architecture, publish by McGraw Hill in 2015.

(Refer Slide Time: 01:27)



So, here where the 5 main point that will cover Boolean algebra. So, Boolean algebra is the algebra basic bits. How to represent positive integers with 0s and 1s, how to represent negative integer with 0 and 1s, how to represent number to the decimal point, call floating points numbers with 0 and 1s, and how to represent pieces of text, namely strings with 0 and 1s.

(Refer Slide Time: 02:03)

What does a Computer Understand ?

- * Computers do not understand natural human languages, nor programming languages
- * They only understand the language of bits

Java/C++
↓ compiler
0s & 1s

Bit	0 or 1
Byte	8 bits
Word	4 bytes
kiloByte	1024 bytes $2^{10} = 1024$
megaByte	10^6 bytes $2^{10} \times 2^{10} = 2^{20}$

Mc Graw Hill Education

So, what does the computer understand the computer is not smart enough to understand languages that we human beings can understand, and that is language such as English or Arabic or French .Computers are not even smart enough to understand programming languages; such as java and C++; that is the reason it is typically necessary to use a compiler, which can convert a java program or C++ or C program into a sequence of basic 0s and 1s, which the computer can understand. So, this processes as we have discussed in chapter one, is known as compilation, and this is typically done by a compiler.

So, let us define the terms with regards to 0s and 1s, say bit mean basic variable which can take only two values, which is the 0 or a 1. Byte is the sequence of 8 bits, which means 8 numbers for each number can either be the 0 or one. Word is the sequence of 4 bytes, or 32 bits, and a kilo bytes. So, kilo here means 1000, is defined as 1024 bytes. Why 1024 is interesting. So, as will see 1024 is an interesting number in the senses two raise to the power 10 is equal to 1024, and this gives us some interesting property is to reason about such systems. So, typically you know when to use a word kilo, you will find mix references in the senses, some places it is use for 1000 and some cases its use for 1024.

Typically in computer science we would use kilo, the term kilo as in kilo byte, as 1024 bytes, because 1024 is 2 raise the power 10, and as we shall see in the latter half of the

chapter this gives us some interesting properties. Similarly, we can define mega. So, mega is kilo times a kilo, which is roughly 1000 times 1000 or a million. And in computer science mega is typically 2 raise to the power 10 multiplied by 2 raise to the power 10, which is 2 raise to the power 20.

So, this is what million is, a mega byte, and this is close to 1 million 10 to the power 6. So, as a result you will find both the number being used, and which variant is being used, needs to be in fort from the context.

(Refer Slide Time: 05:07)

Review of Logical Operations

bit \rightarrow Boolean variable

* $A + B$ (A or B)

A \rightarrow 0 (false)
A \rightarrow 1 (true)

Truth table

A	B	A + B
0	0	0
1	0	\rightarrow 1
0	1	\rightarrow 1
1	1	\rightarrow 1

* $A.B$ (A and B)

A	B	A.B
0	0	0
1	0	0
0	1	0
1	1	1

Mc Graw Hill Education

So, now let us take a look at some basic logical operations 1 bits. So, let us. So, bits bit is also called a Boolean variable in owner of gorge bull, say bit is also called a Boolean variable. So, Boolean variable can only take values 0 or 1. So, let us define some simple operations. So, let us first defined the or operation. The or operation means either. So, if before that let me defined the context, say a can take only two values 0 or 1. So, typically 0 is also called falls, which means that is a statement is not correct and one also means true. So, in the contest of this we can interpreted the or operation.

The or operation says either A or B which been, it either one of them is true. So, let us take a look at what this means in the stable over here, which is also called a truth table, because it truth table typically as all combinations of the input values and the final function. So, as we can see from here, if A and B both are 0, which basically means it both the values are falls say A or B. So, the also a mathematical symbols or is plus say A

or B is also 0. So, in terms of common sense it can say that look if two friends; the name of one friend is A and the name of the second friend is B if both of them are saying something, which is falls then we can say that the combination of A and B both of them together are also saying something; that is falls or nobody is telling the truth.

Now, if it is possible that a is saying something that is true and b saying something, that is falls we can say that you know the combination of A and B at least one of them is speaking the truth, and this logic, is the, this or logic, which means either a is true or b is true is actually correct, it is true, because one of them is 1. So, the final output is also one over here. Similarly a symmetric situation holes over here; that if A is 0 and B is 1, at least one of them is one say A or B here is also equal to 1.

Finally if A is 1 and B is 1, at least one of them is 1 is correct. So, the final output is 1. So, this is the or logic. The or logic basically says that only if A is equal to 0 and B is equal to 0, is the final output equal to 0. Otherwise is it is equal to 1 and all cases. Similarly let us defined the and operator on Boolean variables or Boolean bits, and typically the symbol for and is a dot, as I have shown over here. So, let us take a look.

So, let us again assume two friends A and B, we want only both of them are speaking the truth which means one, do you want to make A and B equal to 1. Otherwise it is 0 and all cases. So, we can see in a sense this is similar to multiplication as well. So, if A is 0 and b is 0, we see that A and B is 0, because in, let us say in the example of speaking the truth, both of them defiantly not speaking the truth, if a is 1 and b is 0 both of them are also not speaking the truth, because in this case B is saying something that is falls. So, you can see that 1 and 0 is equal to 0.

Similarly, is symmetric situation also over here where 0 and 1 is equal to 0? So, vennius an and function positive, it is positive in only one case, which is the last case that I am showing over here, where A is equal to 1 and B is equal to 1, this is when both are speaking the truth. So, let me give an example let us assume that you have two friends right, and simplicity they are names are A and B. And let us assume that they go and you want to ask a that have you had an ice cream, if a speaks the truth then you gave a 1 point otherwise give a 0 points. Similarly we ask the same question to b, and if b speaks the truth, b gets one point; otherwise b gets 0 points.

So, in the case of, you know in this particular case, we have if you want to have an and logic. You say that the function A and B, the function A and B is true. If both A and B individually it is speaking the truth, which means that a is equal to 1 and b is equal to 1 then A and B is equal to 1; otherwise in all the other cases the output is equal to 0 as we can see over here

We will be using these two basics functions the or and the and function in detail. See you can take a look at the book for more details or another book on Boolean logic in Boolean algebra, to get a slightly deeper perspective on the or and functions. We can have a similar function NAND function.

(Refer Slide Time: 11:12)

Review of Logical Operations - II

A	B	A NAND B
0	0	1
1	0	0
0	1	0
1	1	0

A	B	A NOR B
0	0	1
1	0	0
0	1	0
1	1	0

$NAND = \overline{AND}$
 $\overline{0} = 1$
 $\overline{1} = 0$

- * NAND and NOR operations
- * These are **universal operations**. They can be used to implement any Boolean function.

So, an NAND function is actually, the Boolean inverse. So, let us first define at this compliment operator of a NAND. So, what is the compliment? Compliment is very simple, the compliment or 0 is equal to 1, is essentially inverts. So, if you have a 0 the compliment of this is one, and the compliment 1 is equal to 0. So, a NAND function whenever the and function yields a 1 NAND yields a 0, and when and yields a 0 NAND yields a 1, it is like a inverse of it. So, as we can see take a look at the truth table over here, let me the show the previous one we have 0 0 0 and 1. So, this is the exact inverse of a where, we have 1 1 1 and 0.

So, this is the exact inverse of a NAND, it just a compliment or not of the and function. Similarly the or function has a inverse also, which is the NOR function over here n o r.

and if I show you the or function if you take a look if 0 1 1 1, here you have a reverse have a 1 over here and the remaining 3 entries are 0s.

So, it turns out that both NAND as well as NOR are universal operations, in the sense they can be used to implement any Boolean function. We are not at the point where you can prove this result, but simply an interesting result for students to remember, that NAND and NOR operations can implement any kind of logic, any kind of functions. So, you can take a look at the book, you can try proving yourself, or look at sources on the web or other books, for a very detailed proof of this result.

(Refer Slide Time: 13:19)

The slide is titled "Review of Logical Operations" and focuses on the XOR operation. It includes a truth table for $A \oplus B$ with handwritten notes in red ink. The notes include "Exclusive OR" with an arrow pointing to the title, a "2" above the table, and a calculation $2^4 = 16$ next to the table. The truth table has four rows, with the first three rows grouped by a bracket on the right.

	A	B	A XOR B
1	0	0	0x
2	1	0	1
3	0	1	1
4	1	1	0x

Handwritten notes on the slide include "Exclusive OR" with an arrow pointing to the title, a "2" above the table, and a calculation $2^4 = 16$ next to the table. The first three rows of the truth table are grouped by a bracket on the right.

Let us now discuss the XOR operation, or the exclusive or operation. So, it has a different truth table. So, one thing the readers might have realized up to now that if I have two variables, I can create a lot of different kinds of truth tables. So, how many truth tables that I can create? Actually I can create a 6teen truth tables, the reason is like this that every truth table that I have over here, will have precisely 4 rows right, as you can see over here it will have precisely 4 rows; row 1 2 3 and 4. Each row will have a combination of, a unique combination of the different variables. So, it will have 0 0 1 0 1 and 1 1. And for each combination we will store what is the output, that this function is suppose to give.

See in this case we have two variables. So, for two variables we have 4 rows, and for each row, we have a choice, that the output can either be 0 or 1. So, essentially we have

two choices per row. So, how many total choices do we have. We have 2 to the power 4 choices, or 16 possible truth tables we can build, or 16 possible functions we can build if we have two variables, and the XOR function is one such function. So, let us take a look at its properties. So, the XOR that is how it is called. The X stands for exclusive or.

Exclusive or basically means, that it is true only when one of the variables is 1, and the other is 0 or a is 0 and b is 1, in the rest of the cases it is equal to false. As we can see in the first row both A and B are 0 1 0. So, A XOR B is 0. Similarly in the last row both A and B are 1 and 1; so A XOR B is also 0, but in the middle two rows, the second row and the third row A is equal to 1 and b is equal to 0, or alternatively in the third row A is 0 and B is 1. So, as a result in these two rows A XOR B is equal to 1 and in the rest of the two cases A XOR B is equal to 0. So, this is one more kind of function as we have seen.

(Refer Slide Time: 16:05)

Review of Logical Operations

- * ^(complement) NOT operator
 - * Definition: $\overline{0} = 1$, and $\overline{1} = 0$
 - * Double negation: $\overline{\overline{A}} = A$, NOT of (NOT of A) is equal to A itself
- * OR and AND operators
 - * Identity: $A + 0 = A$, and $A \cdot 1 = A$
 - * Annulment: $A + 1 = 1$, $A \cdot 0 = 0$

8

So, just to make a review, we saw the basic not operator. So, not operator or. So, not is also called the complement operator. So, what it does, is that given a Boolean variable you just place a bar on top of it. So, if its value is equal to 0 after a not, the combination will become 1 and 1 bar or not a 1 will become 0

So, one property of the not operator is that the double negation, in a sense, is not of a not, will essentially reduce it to the same variable. So, not of not of a is equal to a itself. Then we saw the or and NAND operators. So, or and NAND operators the truth

table basically for the or cells. If one of the variables is equal to 1 the output is equal to 1, and the truth tables for and Cells that all the variables need to be 1.

So, let us take a look at some of their basic properties. So, we have the property of identity the property of identity is like this, that A or 0 is equal to a, this is very easy to verified A is equal to 0, then output here is 0. If A is equal to 1, then the output here is 1. Similarly we can verified the same property for the and operator, if a is 0 0 and 1 is 0, if a is 1 1 and 1 is 1. So, this is called identity, which basically means that any Boolean variable odd with 0 is equal to the variable itself, and any Boolean variable and 1, is equal to the variable itself.

We have a similar property in the same way in call annulment, which basically says that A or 1 is equal to 1. So, let us see why. If A is 0 0 or 1 is 1, if a is 1 1 or 1 is 1. You can similarly verify the property annulment this for any number ended with 0 is 0, say any any Boolean number. So, in this case away A is 0 0 and 0 a 0, even if a is 1 1 and 0 is 0. So, as we see we can use the similar logic to prove properties for, you know all kinds of Boolean formulae.

(Refer Slide Time: 18:53)

*** Idempotence:** $A + A = A$, $A.A = A$, The result of computing the OR and AND of A with itself is A.

*** Complementarity:** $A + \bar{A} = 1$, $A.\bar{A} = 0$

*** Commutativity:** $A + B = B + A$, $A.B = B.A$, the order of Boolean variables does not matter

*** Associativity:** $A+(B+C) = (A+B)+C$, $A.(B.C) = (A.B).C$, similar to addition and multiplication.

*** Distributivity:** $A.(B + C) = A.B + A.C$, $A+ (B.C) = (A+B).(A+C) \rightarrow$ Use this law to open up parantheses and simplify expressions

NOT \bar{A}
OR $(A+B)$
AND $(A.B)$

A	B	$A.B$
0	0	0
0	1	0
1	0	0
1	1	1

$x(y+z) = xy + xz$

So, let us take look at some more, I will be discussing some of this properties, I put it on the reader to actually consider more details text. So, it can be this book, or it can be other books that are fully devoted to Boolean algebra, to take a look at these properties.

But they are simple properties. So, it is not necessary to consult a lot of text for this, you know a simple, if you just set down in work it out its good enough. So, let us take a look at this property called idempotence. It is say at any number odd with itself is a same number you can easily verify 0 or 0 is 0 1 or 1 is 1. Similarly any number ended with itself is equal to the number itself. This also can be easily verified. the property of complementarity is interesting, with says that any number odd with this compliments is equal to 1, easy to verified considered 0 then a or it. So, the not of 0 is 1 0 or 1 is 1, then you considered 1 and 1 will get, 1 complement is 0. So, 1 or 0 is equal to 1. And similarly we have this property over here, which is A and A bar is equal to 0 can easily we verified in the same fashion.

The next property of interest is commutativity. Here A or B is equal to b or a , means of the order a variable does not matter. Similarly A and B is equal to b and a , the order again does not matter. We also have similar to normal decimal addition and multiplication, we have associativity, which is A or B or C , B or C being here in brackets is same as A or B or C C . It does not matter where you place the brackets, we can place them here and here, or you can place them here and here for Boolean logic, it does not matter. We have the similar setup rules for the and operation as well.

So, we in for example, here we can place b and C inside a bracket, or we can place A and B inside the bracket, does not matter. So, you can, readers can manually put in different values of a B and C can verify, but the result will be as it is. Then we have distributivity. Since distributivity what this means. It similar to what we have in regular arithmetic, like x multiplied with y plus z , is equal to x y , x multiplied by y plus x z . So, this is the long distributivity that we get to see in regular math exactly the same thing also Boolean algebra as well. So, what we see over here, is that A and B or C is exactly the same as A and B or A and C .

So, this particular formula over here, should be verified by the students, a nice way of doing it is just create a simple truth table, have a b c , and then consider the results, consider the result of A and B or C , and then consider this expression over here, just bring it here, and Computed. So, you will see that for each and every row the values match, so the values match for all the rows, it means that both the formulae are the same.

Say exactly a similar formula holds if I interchange and an or say, A or B and C in bracket is equal to A or B and A or C. So, many such rules can be used. These are all the basic rules that we have, and with that we can construct complex formulae. So, I request the readers to go to the book, read all the rules of handling bits of Boolean algebra. What is Boolean algebra? It is the rules for handling bits and operations with bits. What are the basic operations? Well is the not operation is a example, this is a naught, it is the or operation something like A or B when is A or B true. When at least one of A or B is equal to true or equal to 1. Similarly as the and operation.

(Refer Slide Time: 24:12)

De Morgan's Laws

* Two very useful rules

$$\overline{A+B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

A	B	A+B	A.B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

prove this

McGraw Hill Education

10

When is the A and B equal to true, when both a is true and b is true. So, here are two very useful rules, they are call the De Morgan's laws. So, some of the students of the familiar with De Morgan's laws, they were there in set theory. So, what this say in; this is exactly the formula that was there in set theory. It is essentially come to Boolean logic wire set theory. So, this says that A or B compliment is a compliment and b complement. So, let us verify this, you know let us see if this De Morgan's rule is actually correct or not. So, let us you know verify this by. So, let us have A or B compliment, and a compliment and b compliment.


So, let us consider all possible values. So, 0 and 0 A or B, when both are 0, this is 0, the compliment of 0 is equal to 1. So, a compliment is 1, b compliment is 1. So, 1 and 1 is equal to 1. So, both of the match which is fantastic. Now let us consider A to B 1 and b

to be 0, say A or B is 1 and 1 compliment is 0. Similarly here 1 compliment is 0, any number and it with 0 is 0. So, here also the numbers match. And since this case is exactly symmetric to the case above, and the expressions here are symmetric, we can also write 0 and 0, but if you do not trust me let us work it out. Say a is 0 b is 1 A or B is 1 1 compliment is 0, a is 0 0 compliment is 1 and b is 1 the compliment of 1 is 0. Any number ended with 0 is 0. So, we write it over here.

Now, let us consider the last row 1 a is 1 b is 1 A or B is 1 1 compliment is 0. Similarly here we have the compliment of a is 0. So, it does not matter what the value of b is the result is 0. So, one thing that we can see is that all are rows match; since all are rows match both the formulae are absolutely equivalent, they are the same formulae they are equivalent. So, what we have done is that by the truth table we are verified, this result is a very important result, that the compliment of A or B is equal to a compliment and of b compliment. We have a similar result, where we just replaced the or by the and, and the and by the or. So, A and B compliment, is a compliment or b compliments, I will not be proving this. So, this is left to the student that you need to prove, this with a very similar approach that we have used.

(Refer Slide Time: 27:28)

Consensus Theorem



* Prove :


$$* X.Y + \bar{X}.Z + Y.Z = X.Y + \bar{X}.Z$$

$$\Rightarrow X.Y + \bar{X}.Z + Y.Z.1 = X.Y + \bar{X}.Z + Y.Z.(X + \bar{X})$$

$$= X.Y + \bar{X}.Z + Y.Z.X + Y.Z.\bar{X}$$

$$= X.Y.(1+Z) + \bar{X}.Z.(1+Y)$$

$$= X.Y.1 + \bar{X}.Z.1$$

$$= X.Y + \bar{X}.Z$$


.Now, it is time to take a look at Boolean relationship, which is slightly difficult and, but we can work this out. So, this is called the consensus theorem, and this is the statement of the theorem. This is the l h s and this r h s, and you will see many of such burger I can

throughout the presentation, say burger means food for thought. So, you are suppose to think. So, in this case what you are suppose to do, is prove that this expression on the left hand side, is equal to the expression on the right hand side. So, if you want to try you can always pause the video and think what the solution will be, but let me never the less work out the solution. So, the first thing that you need to see, is you need to take a look at this term, this term is present on the left side, and is not present on the right side.

So, this basically means, we need to find a way of eliminating this term from the expression. So, how would we do that, the way that we would do that is something like this. Now, we expand this term something like this; that any number ended with itself is equal to itself. So, what we essentially do over here is that we end it with 1. So, so let me just expand the l h s side, I will ignore the r h s side. So, basically we still have the same terms

So, in this case the term one can be replaced with x or \bar{x} ,. So we are essentially doing this to ensure that y and z can be eliminated. So, what have we done, we have essentially said that any number ended with one is itself. Any number meaning any Boolean number, and now we have replaced the term one with x or \bar{x} , after now we can open up the expression slightly, and use the distributive rule to open it up. So, keeping the rest of the terms we can have.

So, now what we need to do is, we need to rearrange the terms. So, let me rearrange them. So, let me consider the first and the third term. So, since any number ended with one itself. So, we can write. Now let me consider the third term over here. So, I am using the commutative rule to in for the z and \bar{x} is the same as \bar{x} and z . So, now, we can go up, say any number odd with 1 is 1. So, we can replace this with

Which is exactly what is there in the right hand side or the r h s. So, this term is the same as this term, and this term is the same and this term. So, essentially we have proven or first complex Boolean relationship or Boolean equation, using the simple formulae that we have learned up till now. So, mind you this is a very short presentation of Boolean logic. It is possible to work out much more complex formulae, but for that I would request the readers to consult the text book, and if they want more information they can look at the book by Zecovi on switching theory.

(Refer Slide Time: 32:31)

Outline

$b \in \{0,1\}$ NOT, AND, OR
XOR

- * Boolean Algebra
- * Positive Integers
- * Negative Integers
- * Floating Point Numbers
- * Strings

McGraw Hill Education

12

So, let us move forward. So, we have covered some amount of Boolean algebra. What have we learnt, we are learnt a Boolean variables. So, Boolean variable Let us say b you can take only two values and. So, it is a part of the set 0 or 1, and the basic operations that we defined on Boolean variables is not and and or. Then we as some complicated operation like NAND and NOR. Now we also have the x or exclusive or operation. So, these are, may be these 4 operations can be considered to be basic, and we looked at sudden simple rules of manipulating Boolean variables.

So, now the question arises, why did we learn Boolean algebra? This is something that I did not tell you, but it clearly has some implication for the rest of the 4 topics. So, let us see.

(Refer Slide Time: 33:29)

Representing Positive Integers

- * Ancient Roman System

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

- * Issues :
 - * There was no notion of 0
 - * Very difficult to represent large numbers
 - * Addition, and subtraction (**very difficult**)

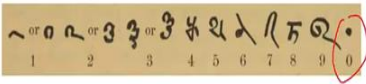
Mc
Graw
Hill
Education

13

So, let me slightly digress at this point, and then we will stitch everything together. So, let us take a look at the ancient roman system, how they were representing numbers. So, they had roman number of the form 1 was a single I 5 was represented with V type symbol 10 by X 50 by 100 by C 500 by D 1000 by M. So, they did not follow a place value system the way that you know modern numbers and made. So, it was very difficult to represent large numbers and addition and subtraction was fairly difficult in the system, but in an ancient civilization they did not also have the requirement of representing very large numbers. So, this is something that went in the favor.

(Refer Slide Time: 34:23)

Indian System



Bakshali numerals, 7th century AD

- * Uses the place value system

$$5301 = 5 * 10^3 + 3 * 10^2 + 0 * 10^1 + 1 * 10^0$$

Example in base 10

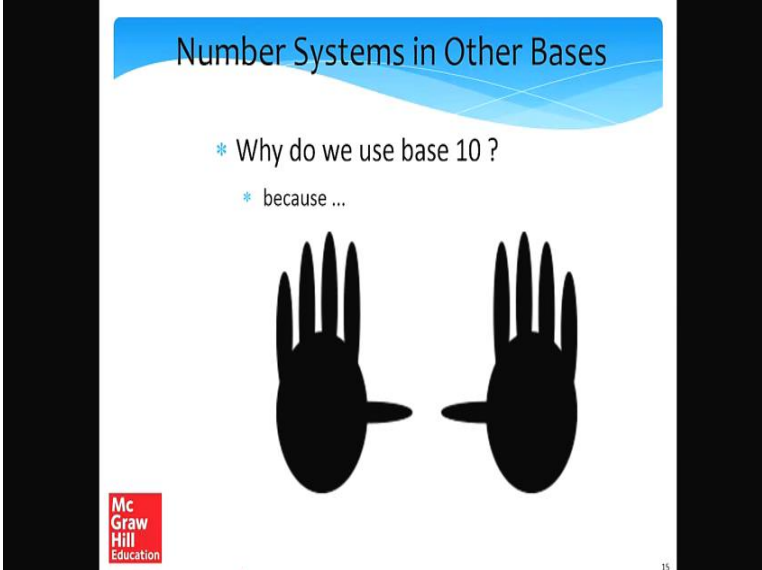
Mc
Graw
Hill
Education

14

So, in comparison the Indian system which later on begins into Arabic numeral system had the number 0 which is the crux of the idea right. So, this is an old script recovered from Bakshali which is on modern day Pakistan actually, from the 7th century A.D. so it uses the place value system.

So, what is the place value system, any number of this form is 5 times 10 to the power 3 plus 3 times 10 square 0 times 10 to the power 1 plus 1 times 10 to the power 0. So, the number is 10 over here can be considered to be the base. So, this particular example is an example in base 10

(Refer Slide Time: 35:16)



The slide features a blue header with the title "Number Systems in Other Bases". Below the header, there is a question: "* Why do we use base 10 ?" followed by an answer: "* because ...". Underneath the text, there are two black silhouettes of hands, palms facing each other, representing ten fingers. In the bottom left corner, there is a red logo for "Mc Graw Hill Education". In the bottom right corner, there is a small number "15".

So, why do we use base 10? Well, simple we have 10 fingers that is how we learnt counting in the first place, that we count with a fingers from 1 to 10.

(Refer Slide Time: 35:29)

What if we had a world in which ...

- * People had only two fingers.



Mc
Graw
Hill
Education

16

What if we go to another planet? In a planet which is far away, where people have only two fingers rights. So, the entire planet all the people there very intelligent, but they have only two finger.

(Refer Slide Time: 35:44)

Binary Number System

- * They would use a number system with base 2.

Number in decimal	Number in binary
5	101
100	1100100
500	111110100
1024	1000000000

$5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$3 = 2^1 + 2^0 = (11)_2$

binary

$19 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (10011)_2$

Mc
Graw
Hill
Education

17

So, instead of base 10, they would actually use based two. So, in base two how would you represent a number? The way we would do that, is you considered 5, 5 would be represented as 1 times. So, 2 square plus 0 times 2 to the power 1 plus 1 multiplied by 2 to the power 0. So, you know sense the number would be 1 0 1, which is exactly what we

get to see over here. So, similarly what would the number 3 be? The number 3 would be 2 to the power 1 plus 2 to the power 0

Which can be represented as 1 1, and let us say base 2 will use the subscript for the base. So, this is also called the binary notation; or are the base two notation. So, what we see over here is that it is not necessary to use a base of 10 to represent a number. We can use the different ways as well, and we use 10 because we have 10 fingers, but you can use base two if you have just two fingers, and any numbers I have shown many examples, can be represented in the base two notation. Let me again one more example lecture number 19 how would you represent that. So, the its very simple its 1 times 2 to the power 4 which is 16 plus 0 times 2 to the power 3 plus 0 times 2 square plus 1 times 2 to the power 1 plus 1 times 2 to the power 0. So, this would be equal to 1 0 0 1 1 in base 2. So, this is. So, 1 0 0 1 1, is the way that we would represent the number 19 in the base 2 system or the binary system.

(Refer Slide Time: 38:02)

The slide features a blue header with the title "MSB and LSB". Below the header, there are two bullet points: a blue one for MSB and a green one for LSB. The MSB point includes a red bracket under the first '1' of the binary number "1110". The LSB point includes a red bracket under the last '0' of "1110". The slide also contains the McGraw Hill Education logo in the bottom left and a small page number "18" in the bottom right.

MSB and LSB

- * **MSB (Most Significant Bit)** → The leftmost bit of a binary number. E.g., MSB of 1110 is 1
- * **LSB (Least Significant Bit)** → The rightmost bit of a binary number. E.g., LSB of 1110 is 0

McGraw Hill Education

18

So, now let us define to terms the MSB the most significant bit and the least significant bit. So, the leftmost bit of a binary number; for example, number 1 1 1 0 the numbers shown over here, the most significant bit is 1, and the least significant bit is the rightmost bit, and in this case it is 0.

(Refer Slide Time: 38:26)

The slide is titled "Hexadecimal and Octal Numbers" and contains the following content:

- * Hexadecimal numbers**
 - * Base 16 numbers – 0,1,2,3,4,5,6,7,8,9, ¹⁰A, ¹¹B, ¹²C, ¹³D, ¹⁴E, ¹⁵F
 - * Start with 0x $0x9 = (9)_{10}$
- * Octal Numbers**
 - * Base 8 numbers – 0,1,2,3,4,5,6,7
 - * Start with 0 $042 = (4 \times 8 + 2) = (34)_{10}$

Handwritten notes on the right side of the slide:

- base 10: 0...9
- base 2: 0, 1
- base 16: 0...9, A, B, C, D, E, F

McGraw Hill Education logo is visible in the bottom left corner of the slide.

So, similarly we can define numbers in other basis as well. So, it is very common in computer science and Computer design, to represent numbers in base 16. So, these are called hexadecimal numbers. So, so what is the problem in representing numbers in base 16. So, let us first take a look at base 10. So, in base 10 what are our digits? Our digits are 0 till 9, so we have 10 digits.

In base 2 how many digits do we have? We have two digits 0 and 1. So, in base 16 we need to have 16 digits from 0 to 15, but the problem is, that till 9 we have digits define, but beyond 10 right we do not have a digits define. So, that is the reason what we do is, we use the set from 0 to 9, and then we say that a represents 10 in decimal of course, b represents 11, and in a same vein f represent 15. So, a b c d f are the 6 extra digits that we create, to represent a number in base 16 or an hexadecimal number. And these 6 extra digits represent the numbers 10 to 15.

So, the way we represent a number hex number is that we typically add 0 x before it. For example, 0 x 9, is actually 9 in base 10. Similarly 0 x 32 is equal to 3 times 16 plus 2 times 1 which is actually equal to 50 in base 10. So, I would like to request the reader at this point of time, to take as many numbers as she wants, and Convert them to base 16, and Convert them to any base of their choice, and just see how this works out. So, similarly we can have octal numbers. So, in octal number is in base 8, and here we will have 8 digits 0 to 7. So, which is fine, we do not need to introduce any new digit, and

octal number start with the 0. So, maybe the number 0 4 2, will be equivalent to 4 times 8 plus 2. So, this is equal to 34 in base 10. So, we can work too many more examples, but I would suggest that you know we move forward, and the reader spend some time at this particular, you know at this slide ,which is slide 19 to convert numbers between arbitrary basic.

(Refer Slide Time: 41:30)

Examples

0s and 1s

binary

Convert 110010111 to the octal format: $\underline{110} \underline{010} \underline{111} = 0623_8$

6 2 7

Convert 111000101111 to the hex format: $\underline{1110} \underline{0010} \underline{1111} = 0xC2F$

a₀ ... a_k ∈ {0,1}

$$N = a_0 \times 2^0 + a_1 \times 2^1 + \dots + a_k \times 2^k \quad \text{binary } a_k \dots a_0$$

$$= (a_0 + 2 \times a_1 + 4 \times a_2) + 2^3 (a_3 + 2 \times a_4 + 4 \times a_5) + 2^6 (\dots) + \dots$$

$$= \underbrace{b_0}_{b_0} \times 8^0 + \underbrace{b_1}_{b_1} \times 8^1 + \underbrace{b_2}_{b_2} \times 8^2 + \dots + \underbrace{b_k}_{b_k} \times 8^k$$

(b_k, ..., b₂, b₁, b₀)

Let us now discuss why we choose the binary the octal and the hex formats. Well the binary is very straight forward computers understand the language of only 0s and 1s. So, as a result if you have a number system that represents everything in terms of 0s and 1s, which is a binary or a based 2 number system, this is information that the computers scan process. So, what about base 8 and base 16. Well base 8 and base 16, will turn out to be short cut for the binary form, but how that is the case that is slightly more complicated. So, let us work it out. So, consider any number right rather number be n. So, we can say there the number is the form.

So, we can sort of break it down into sums of powers of two, and each of these numbers a 0 to a k can be a binary number in 0 and 1, which is exactly are binary representation. So, the binary number would essentially b. So, this is something that we already know. Now we can alternatively say that this is equal to.

So, what we have essentially done, is that we have group that terms into groups of 3 terms, and the way that we have group them is something like this, that we take 3 terms

to the first term is a 0 and 2 to the power 0 is 1, then we take 2 times a 1 and 4 times a 2, which is also 2 to the power 1 and 2 to the power 2. Similarly when we come to the next term

So, we again group the terms. So, the third term over here is a 3 multiplied by 2 to the power 3. So, what I have done, is that I have brought 2 to the power 3 outside the bracket, and similarly we have something similar, sorry this is the plus a 3 plus 2 times a 4 plus 4 times a 5 and so on. So, the interesting thing is that if you consider each of these numbers a 0 a 1 a 2, all of these numbers still a k. essentially all of these numbers are either 0 or 1, they have Boolean bits. So, if I consider this term the maximum value of this term is 4 times 1 plus 2 times 1 plus 1 which is 7, and the minimum value for each of these terms this one or this one or this one is 0, when all 3 are the coefficients are 0.

So, alternatively the way that I can write is that this entire coefficient, I can think of this as b_0 . So, I can write this as. So, what we have essentially done is that we have expanded this number in terms of parts of two, then group together 3 terms, and then essentially we have expanded the number in powers of 8. So, for those who have not been, who have not had that magical bulb light inside you up till now. The magical bulb is like this, that we have essentially broken down n, into some of parts of 8 which is nothing, but a number in the octal notation.

So let us assume that the last number in this series is k dash times 8 to the power k dash. So, essentially the number, the octal number will be equal to. So, this is the number in base 8. So, what is the important thing that we learn over here? The important thing that we learn over here is that if we group terms into groups of 3, and each of them is essentially an octal digit right. So, basically if we let us say group them into 4, each of them is hexadecimal digit. So, this gives us a very nice and short form of writing binary numbers in other bases that are a power of 2 for example, if you have this number over here, we can group it into blocks of 3, and each such block we can convert it into an octal digit. So, this number, I am sorry this is the mistake over here it should be 7. So, this number is a 7, this is a 2, and this is the 6. So, this is 0 6 2 7.

Similarly we can take this number over here, and if you want to convert it to base 16 that is also very easy. We group 4 binary digit, so we 1 1 1 1 is an f 0 0 1 0 is 2 and 1 1 1 0 is, well so 1 1 1 0 is 12 which is the hex digit c. So, what we have is that we have a very

compact representation for 12 binary bits, using 3 hexadecimal digits. Hexadecimal is abbreviated as hex. So, what do we see over here? What we see is a binary the octal and the hex formats are essentially the same. The binary format is varies parts, the other formats are slightly more dense. So, this in a sense is desirable if you want to represent binary bits, instead a representing them as 1s and 0s. We can at least file writing or while programming we can represent them as hexadecimal digits, where we group digits into blocks of 4, and each block is replaced by a hexadecimal digit.

Similarly, when we need to expand a number in the hexadecimal notation to a number in the binary notation, all that we need to do is, take each of these digits and replace them with 4 binary numbers and we will have the binary notation.