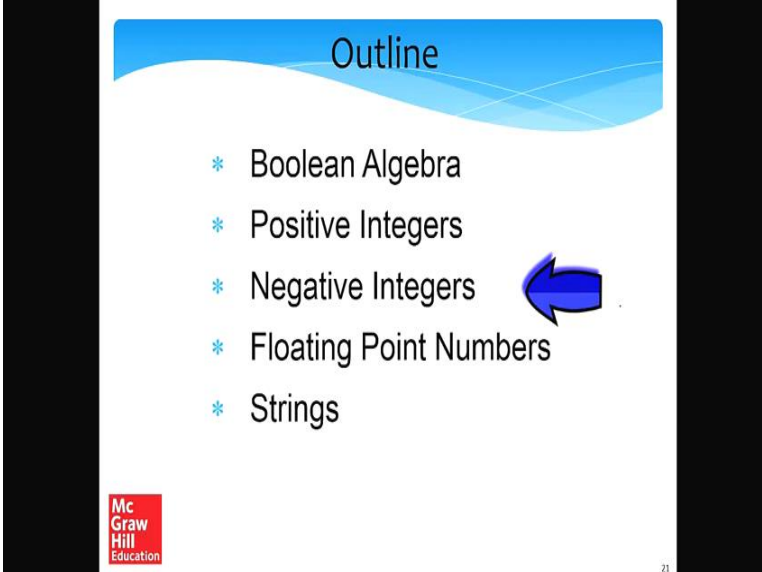


Computer Architecture
Prof. Smruti Ranjan Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture - 03
The language of Bits Part-II

(Refer Slide Time: 00:26)



Outline

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

Mc
Graw
Hill
Education


21

So now we have discussed about representing positive integers in our system, which is actually very easy. So, let us now discuss how to represent negative integers in our system using 0s and 1s.

(Refer Slide Time: 00:41)

Representing Negative Integers

- * Problem
 - * Assign a ^{0s & 1s} binary representation to a negative integer
 - * Consider a negative integer, S
 - * Let its binary representation be : $X_n X_{n-1} \dots X_2 X_1$
($X_i = 0/1$)
^{msb}^{lsb}
 - * We can also expand it to represent an unsigned, +ve, number, N
 - * If we interpret the binary sequence as :
 - * An unsigned number, we get N .
 - * A signed number, we get S .



22

So, what is the problem? What we want to do is, that given a negative integers, we want to assign a binary representation to it in terms of just in a 0s and 1s. So, consider a negative integer S , assume that its binary form is X_n to X_1 , where X_1 is the least significant bit, and X_n is the most significant bit. And hence forth we will actually start counting from 1 not from 0. So, now, we can also, you know given this number over here which we are representing we can also expanded to represent an unsigned. Unsigned means without a sign assumed to be positive number N .

So, we can interpret this binary sequence here in two ways; one is we can interpret it as an unsigned number, so we will get N , or we can represent it as a number with the sign and we will get S .

(Refer Slide Time: 01:45)

- * We need a mapping :
- * $F: S \rightarrow N$ (mapping function)
- * $S \rightarrow$ set of numbers (both positive and negative - signed)
- * $N \rightarrow$ set of positive numbers (unsigned)

Mc
Graw
Hill
Education

23

So, what we need to do is, we need to create a mapping, from the set of all the numbers with a sign, which can include both positive numbers as well as negative numbers to a set of binary unsigned numbers called N. So, this mapping would essentially tell us how to represent a number be it positive or negative using a set of binary bits. So, let the set of positive and negative numbers be called as, and let each number irrespective of it is sign positive or negative, be mapped through a sequence of binary numbers and we can alternatively interpret this number as a positive number, and let us call this set as N; if you find a mapping will be able to represent both positive as well as negative numbers using a binary representation. For the positive numbers we already have some idea from a previous discussion, but a previous discussion does not cover negative numbers. So, this needs to be covered over here.

(Refer Slide Time: 02:57)

Properties of the Mapping Function

- * Preferably, needs to be a **one to one** mapping
- * **All the entries** in the set, S, need to be mapped
- * It should be **easy to perform addition and subtraction** operations on the representation of signed numbers
- * Assume an n bit number system $x_n \dots x_1$

$$\text{SgnBit}(u) = \begin{cases} 1, & u < 0 \\ 0, & u \geq 0 \end{cases}$$

So, let us now take a look at properties of the mapping function. So, preferably you need a one to one mapping, which means that given a certain signed number, it needs to have only one binary representation. So, also the thing is that, all the entries in the set S that we need to map should be mapped, it should be easy to perform arithmetic operations on them such as addition and subtraction and. So, these all are 3 main assumptions. So, let us look at one of the simplest possible ways of doing it.

See assume an n bit number system and number system in which we have n bits. So, n bits basically means that in our number system, we will represent every number by a sequence of n bits right n binary bits, where they can be from x_n to x_1 . Every number will be mapped with n bits. So, we also want to introduce this function sign bit u, which if u is less than 0 if the number is less than 0, this sign bit will return 1, if u is greater than equal to 0 it will return 0. So, essentially the number is positive or 0 the sign bits 0, if the number is negative the sign bits 1. So, this is a function that we want to introduce.

(Refer Slide Time: 04:32)

Sign-Magnitude Base Representation

$$F(u) = \text{SgnBit}(u) * 2^{n-1} + |u|$$

* Examples :

- * -5 in a 4 bit number system : 1101
- * 5 in a 4 bit number system : 0101
- * -3 in a 4 bit number system : 1011

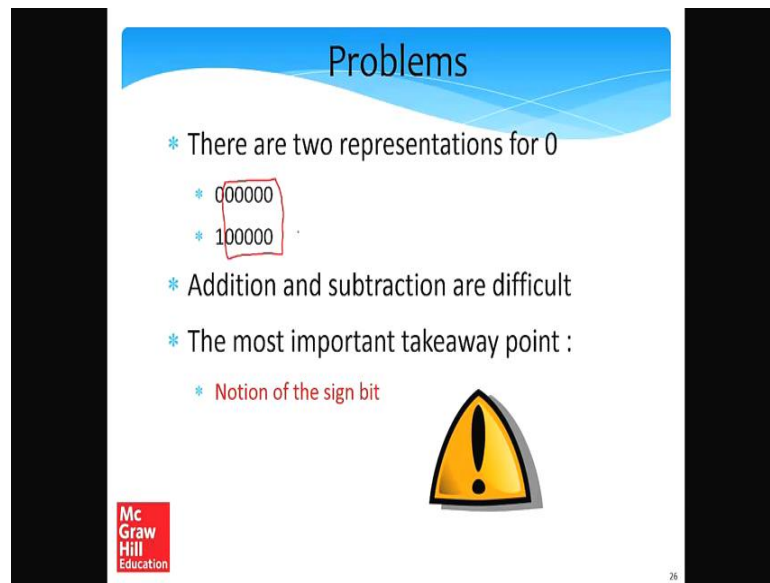
Mc
Graw
Hill
Education

25

So, let us first look at a very simple kind of representation called as sign magnitude base representation. So, here what we do is that the way we define the mapping is something like this, that we define a function $F(u)$, where u is in a , where u can be a negative number or a positive number. So, we consider the sign bit multiply it with 2 to the power n minus 1 and added with the absolute value of u , or alternatively we can think that the absolute value of u is the least significant portion of the representation, and the sign bit which is negative, which is 1 if the number is negative, the M S B of the representation. So, this number system is actually very simple.

So, let us look at an example, let us consider minus 5. So, minus 5, we first write take the binary representation of plus 5 which is 1 0 1, and it is sign bits equal to 1, because a number is negative. So, this is the representation that we have, if we consider plus 5 that is also very easy we take 1 0 1, and we treat the most significant bit as 0. So, one thing that we can immediately see over here is, at this number that we have in this representation a positive number is being represented in exactly the same way as we had discussed, for a negative number we just add a 1 before it; consider one more example minus 3. So, we first write the representations of plus 3, and then we add a 1 as the M S B digit, because a number is negative.

(Refer Slide Time: 06:26)



The slide is titled "Problems" in a blue header. It contains the following text:

- * There are two representations for 0
 - * 000000
 - * 100000
- * Addition and subtraction are difficult
- * The most important takeaway point :
 - * Notion of the sign bit

There is a yellow warning triangle icon with a black exclamation mark inside. The McGraw Hill Education logo is in the bottom left corner, and the number 26 is in the bottom right corner.

So, what are the problems? There are two representations for 0. So, 0 can you know both of them are, both of these are valid representations, because the rest of the bits are all 0, and the sign bits either 0 or 1. So, you know it does not matter, but still there is a problem in the sense that you are having a positive 0, and a negative 0. So, addition and subtraction are difficult, we can clearly see if you just regularly add them as binary numbers, ten there is a problem. So, the most important takeaway point is, yes we introduced a sign bit, the sign bit did have some advantages no doubt. However, it introduced more problems than it could solve for example, there are 2 representations of 0, which is something that we do not want.

(Refer Slide Time: 07:30)

1's Complement Representation

$$F(u) = \begin{cases} u, & u \geq 0 \\ \sim(u) \text{ or } (2^n - 1 - |u|), & u < 0 \end{cases}$$

* Examples in a 4 bit number system

- * 3 → 0011
- * -3 → 1100
- * 5 → 0101
- * -5 → 1010

Mc
Graw
Hill
Education

21

So, now let us look at the 1's complement representation, which is very simple. Let us first look at the examples. So, if you have plus 3 gets represented in the same way as other positive number is 0 0 1 1. If we consider minus 3 what we do is, that we just invert every single bit of 0 0 1 1 to get 1 1 0 0. Similarly if we consider plus 5, we take the regular unsigned representation of plus 5 which is 0 1 0 1, and if we consider minus 5, then what we basically do is that we invert every single bit of plus 5 to get minus 5. Here also you will notice that the notation of the sign bits also come, in the sense that for numbers that are positive the M S B, the most significant bits 0, and for numbers that are negative the most significant bit 1. So, in a sense the notation of the sign bits also come that positive numbers will have, start with 0 and negative numbers will start with a 1, which helps us distinguish between positive and negative, but it does not confer any more mathematical properties.

(Refer Slide Time: 09:02)


Problems

- * Two representations for 0
 - * 000000 —
 - * 111111 —
- * Easy to add +ve numbers
- * Hard to add -ve numbers
- * Point to note :
 - * The idea of a complement

$$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$$

3 + (-2) = 1

$$\begin{array}{r} 0011 \\ + 1101 \\ \hline 0000 \end{array}$$



Mc
Graw
Hill
Education

28

So, let us look at the problems of this representation. The problems of this representation are like this, that there are two representations for 0, which is something that we wanted to avoid. So, 0 0 0 is also 0 and 1 1 1 is also 0 and this is not in our best interest in the sense, we did not want. It introduces ambiguity in to the number system; it is easy to add positive numbers. How do you add positive numbers? Let us give a simple example, let us assume number sys, a number of the form 3 0 0 1 1 and you want to add this 3 plus 2. Well you can do simple binary addition as you do decimal addition, see add 1 and 0 you get 1, you get 1 and 1. So, 1 plus 1 is 2. So, 2 and binary is 1 0. So, you write 0 here you have one as the carry, see again write one over here.

So, this is 0 1 0 1 which is 5 this is absolutely fine. It is easy to add positive numbers, unfortunately it is hard to add negative numbers in this particular representation, and that is the sad part of doing this, that adding negative numbers. So, negative numbers cannot really be added the way that we added positive numbers, and that will create some problems for us, which is not something that we wanted. So, if we want to see. So, let us consider an example. So, let us see we want to add 3 plus minus 2, which is 3 minus to the, answer should be 1. So, 3 is 0 0 1 1 minus 2 is. Well let us consider plus 2. So, this is plus 2. So, invert every bit of it 1 1 0 1 then let us add them. So, 1 plus 1 is 0 and the carry is 1, 1 plus 1 is 2. So, the carry is 1, 1 plus 1 is again 0 1 1 plus 1 again 0 1 again 0.

So, clearly this answer is not correct, this answer is wrong, and this produces an output carry a 1 which is again you know not correct. So, the answer should have been equal to 1 over here, which is clearly not the case. So, we do not have a number system, where it is easy to perform; arithmetic operations on both positive and negative numbers. So, when nevertheless the point to notice the idea of a compliment; that we can take the compliment of every bit and try to represent that as a number. So, there can be some merit is to it, but well. So, we have seen 2 ideas; the idea a sign bit on a compliment individually, these ideas are not that strong.

(Refer Slide Time: 12:17)

1's Complement Representation

$$F(u) = \begin{cases} u, u \geq 0 \\ \sim(|u|) \text{ or } (2^n - 1 - |u|), u < 0 \end{cases}$$

* Examples in a 4 bit number system

- * 3 → 0011
- * -3 → 1100
- * 5 → 0101
- * -5 → 1010

$$\begin{array}{r} 0011 \\ -1111 \\ \hline -1100 \end{array}$$

$$\begin{array}{l} 0/1 \quad 1/0 \\ \times \quad \times \\ 1-x \quad (A/O) \end{array}$$

McGraw Hill Education

21

So, before we move forward let me explain this formula that I have skipped. So, here what are we trying to do. We are considering a number u which can either be positive or negative. We are trying to map it this is the set S, we are trying to map it to a sequence of binary numbers sequence of binary digits, where N is it is unsigned or positive form. So, for positive numbers we do nothing, it is u itself, u is greater than or equal to 0, and for negative numbers, what we do is we take the value of u and we compliment every single bit of it, whether tilde sign is also the sign for a bitwise compliment. Alternatively what we can do is also you consider minus 3, how did we get here? We took 0 0 1 1 and we essentially subtracted this number from this number.

So, if you subtract anything from 1, you actually get its compliment, we can clearly see why 1 minus 1 is 0 1 minus 1 is 0 1 minus 0 is 1 1 minus 0 is 1, which is low and behold

the representation for minus 3. These methods are algebraically equivalent. So, let us give a Boolean variable x , if I consider x complement that is also the same as 1 minus x , why is this, the case assume that x is 0 , x complement is 1 and 1 minus x is also 1 ; assume x is 1 , x complement is 0 and 1 minus 1 is also 0 .

So, what we are doing over here is that. So, 2 to the power n minus 1 in a 4 bit number system would be 15 , which is represented as all 1 's. So, take a complement of a number, we can also subtract its absolute value from 2 to the power n minus 1 . So, we will get the complement of the number. So, there are two ways of thinking about how to take a 1 s complement. One is we take a complement every single bit or we subtract the number from 2 to the power n minus 1 , in this case n is 4 , because we have 4 bits. So, 2 to the power n minus 1 is 15 . So, we can subtract it from 15 in a sense it is equivalent to complementing every single bit.

So, already discussed, this problem has this approach has the set of problems, the biggest problem is there are two representations was 0 , and adding a positive 2 and negative number, or a negative 2 a negative number is hard, and this is not something that we would want to do.


(Refer Slide Time: 15:13)

Bias Based Approach

$$F(u) = u + \text{bias}$$

- * Consider a 4 bit number system with bias equal to 7
- * $-3 \rightarrow 0100$ $-3 + 7 = 4$ $3 + (-3)$
- * $3 \rightarrow 1010$ (10) $3 + 7 = 10$ $\begin{array}{r} 0100 \\ +1010 \\ \hline 1110 \end{array}$ (14)
- * $F(u+v) = F(u) + F(v) - \text{bias}$ (7) $\begin{array}{r} 1110 \\ -7 \\ \hline 1000 \end{array}$ (8)
- * Multiplication is difficult

$F(3-3) = F(3) - F(3) + \text{bias}$
 $= 7 - 7 + 7 = 7$ $F(3) = 0 + \text{bias} = 0 + 7 = 7$
 $F(3 + (-3)) = F(0)$


29

So, then let us consider one more approach called a bias based approach, which is used in a different number system, we will have a chance to talk about it. So, you consider 4 bit number system with the bias equal to 7 . So, what are we trying to do? We are saying

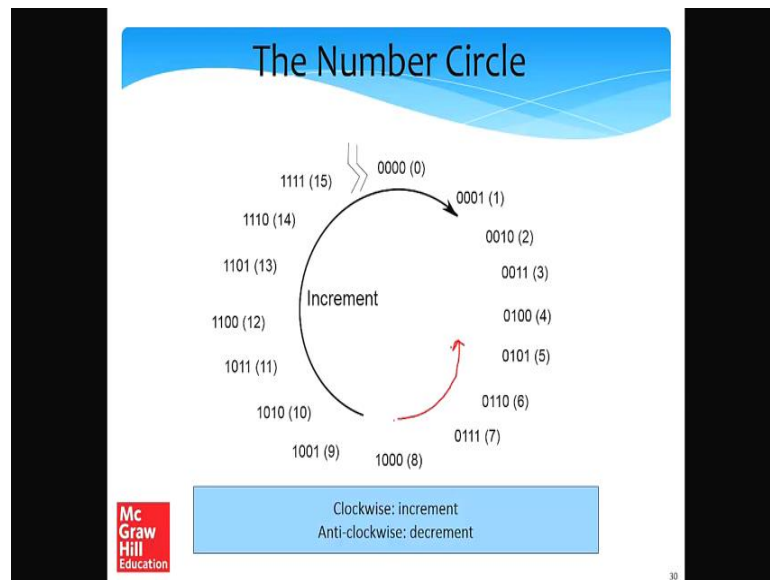
that it take any number is representation is a number plus the bias fine. So, let us see minus 3 it is representation would be minus 3 plus 7. So, minus 3 plus 7 is equal to 4 and it is binary representation is 0 1 0 0. Similarly if you take plus 3 will be 3 plus 7 which is equal to 10, and this is exactly what we have 8 plus 0 plus 2 plus 0 which is 10 in decimal.

If we want to add 2 numbers it is easy. So, in this case addition of number is easy, all that we need to do is the representation of $u + v$ F of $u + v$, is $F u + F v$ minus bias. So, we can clearly see why that is the case; let us say I want to do 3 plus minus 3. So, this would be equivalent to adding 4 sorry. So, basically we are adding 0 1 0 0 plus 1 0 1 0. So, this is 0 1 1 1. So, this in decimal is 14 then we need to subtract the bias which is equivalent to 7. So, if you subtract this in decimal we get 7, and what is 7? 7 well is the representation for 0, because in our system F of 0 the representation of 0 0 plus the bias, which is 0 plus 7 equal to 7.

So, what do we have we have F of 3 plus minus 3 is equal to F of 0, which is exactly the answer that we wanted. So, the advantage of this system is, there is only one representation for 0, which is the bias and 0 for 0 2 representations are not there. The other is at doing addition and subtraction is fairly easy there is absolutely no problem in doing an addition; subtraction can be slightly difficult, not very difficult though. So, if you consider subtraction, so let us try to subtract F of 3 minus 3. So, this is essentially equal to.

So, in this case instead of subtracting the bias, you add the bias. So, this is equal to 7, which is the same as. So, subtraction is also easy, but multiplication is difficult. So, now, we have seen 3 kinds of systems; one was the 1's compliment system, one is this particular approach, which is the bias based approach, and other was the sign magnitude system.

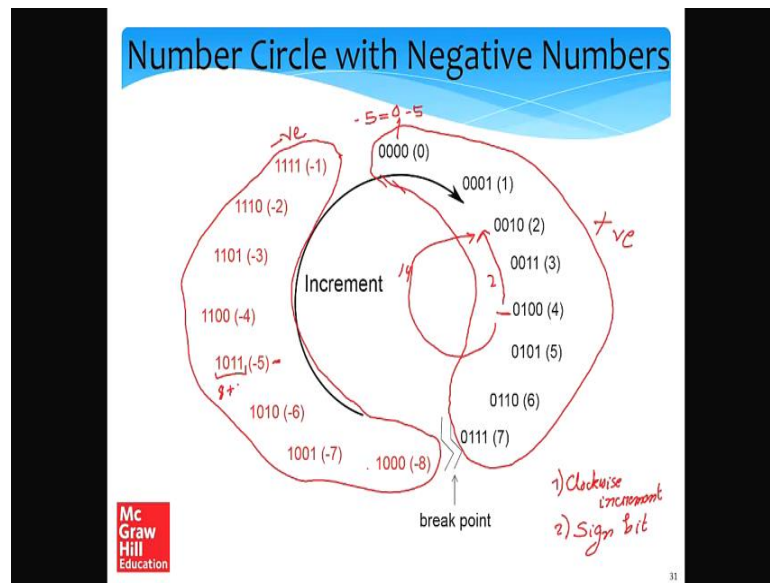
(Refer Slide Time: 18:39)



So, let us now discuss one approach which is used, and it is used for a very good reason, there it has very good. So, what was the reasons once again? The reasons are we need one representation for 0 that is point number 1. We should have some nice mathematical properties in the sense that we should be able to nicely add and subtract and multiply the number may be not divide. So, our all previous representations were simple, but they fell short.

So, let us do one thing, let us consider the set of all 4 bit numbers, and try to represent them as a circle. So, this is called the number circle. So, let us just consider 0 1 2 3 4 5 6 till 15, and from 0 till 15. So, each one of them I have it is binary unsigned representation and so one thing that you see is, at 15 and 0 I have placed a line like this as a discontinuity, and as I travel clockwise, the numbers are getting incremented. So, clockwise a increment, and if I travel anti clockwise which is the other direction this way, I am decrementing the numbers, did you seen the numbers by 1.

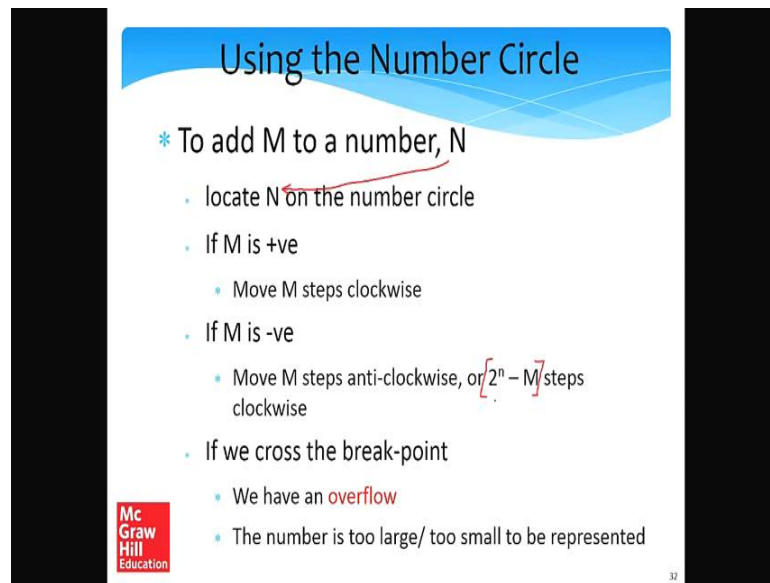
(Refer Slide Time: 20:01)



So, what I can do, if I can do something really interesting, I can take the same numbers circle and use the concept as a signed bit. So, I can treat all of these numbers from 0 till 7 as positive numbers, and after 7 I can treat them as negative numbers, but I will treat them in a negative number in a special way. So, after 0 1 1 1 the next number is 1 0 0. So, let me call this minus 8, and then minus 7, minus 6 minus 1 and so on. So, the red one over here are my set of negative numbers. So, a note that in this case the pointer, there is no discontinuity of this point; say minus 1 here and 0 here, but there is a point of discontinuity over here that we call the breakpoint.

So, let us now take a look at some of the salient features of making a number circle like this. As I moved clockwise by essentially increment the number other than are the breakpoints. So, as you see as I moved clockwise I am incrementing the number. So, this is feature number 1, that this clockwise increment is there in this particular number circle. Second, the notation of the sign bits there in the sense that if you take a look at all the positive numbers, their M S B of the most significant bits 0; and if you take a look at all the negative numbers their M S B of the most significant bits equal to 1. So, in that sense the notation of the sign bits there, and it has been preserved. So, this will come useful.

(Refer Slide Time: 22:12)



Using the Number Circle

- * To add M to a number, N
 - locate N on the number circle
 - If M is +ve
 - Move M steps clockwise
 - If M is -ve
 - Move M steps anti-clockwise, or $[2^n - M]$ steps clockwise
 - If we cross the break-point
 - We have an **overflow**
 - The number is too large/ too small to be represented

Mc Graw Hill Education

32

So, let us now try to use the number circle. So, let us assume that you want to add M to the number N. So, the first thing that we do is we locate N on the number circle, if M is positive we move M steps clockwise because every single step that we move we increment the number by 1. Similarly if M is negative we move M steps anticlockwise or. So, moving M steps anticlockwise will essentially decrement the number, and this is equivalent. So, so here is the very important point.

So, moving so in this case let me consider in this point 4, I want to subtract 2. So, I can move 2 steps anticlockwise one step second step. So, moving two steps anticlockwise is exactly the same as moving 14 steps clockwise right. So, this is a very important point that all of us need to remember this the cracks of this approach, that moving two steps anticlockwise, is exactly the same as moving 14 steps clockwise, which is exactly what we are trying to say, that in an N bit number system moving M steps anticlockwise, is the same as moving $2^n - M$ steps clockwise clearly. If we cross the breakpoint we can have an overflow which means a number is too large or too small to be represented, but we will discuss such special cases later.

(Refer Slide Time: 23:50)

2's Complement Notation

$$F(u) = \begin{cases} u, & 0 \leq u \leq 2^{n-1} - 1 \\ 2^n - |u|, & -2^{n-1} \leq u < 0 \end{cases}$$

* $F(u)$ is the index of a point on the **number circle**. It varies from 0 to $2^n - 1$

* Examples

- * $4 \rightarrow 0100 \leftarrow$
- * $-4 \rightarrow 1100 \leftarrow \quad 16 - 4 = 12$
- * $5 \rightarrow 0101 \rightarrow$
- * $-3 \rightarrow 1101 \quad = 16 - |3| = 13$

33

So, keeping these things in mind, let us define the 2's complement notation. So, the 2's complement notation is something like this. So, we will represent positive numbers, as we were doing, as long as a number is between 0 and to the power n minus 1 minus 1 in a 4 bit number system 2 to the power n minus 1 is 2 to the power 4 minus 1 and minus 1. So, this comes out to be 2 to the power 3, 8 minus 1 which is 7. So, as long as the number is between 0 and 7 all inclusive, we will represent it as u, which is exactly what we are doing in the number circle that a number between 0 and 7 less represented as the unsigned binary representation of the number itself. However if a number the moment it becomes negative, then what we will do, and the number is between minus 2 to the power n minus 1, which is like in a 4 bit number system minus 8 and 0, we will represent this as 2 to the power n minus the absolute value of u. So, this is very interesting why this is done.

So, let us start from point 0, let us assume that I want to represent minus 5. So, minus 5 is same as 0 minus 5. So, I should walk 5 steps anticlockwise. So, 1 2 3 4 and 5 I reach over here. So, the other is at the crux of the idea is walking 5 steps anticlockwise, also walking 11 steps clockwise. So, are the representation of minus 5 should be the same as walking 11 steps clockwise, which is plus 11 and you will see this is exactly the case that we have 8 plus 2 plus 1 which is plus 11. So, this is exactly what is being shown in this formula over here, that in the number circle representing a number of the form, u will, u is negative, is the same as 2 to the power n minus the absolute value of u. So, this is like

walking u steps absolute value of u steps anticlockwise from 0 or moving these many steps clockwise.

So, $F(u)$ in this case is index of a point on the number circle, and this will vary from 0 to $2^n - 1$. So, simple examples for plus 4 what we do? For plus 4, what we do is we represent it within the range of the number system 0 and 7. So, we represent this as 0 1 0 0. For minus 4 the way we represent, this is very simple, we just to 16 minus 4 which is 12, and this is exactly the representation of 12 in the binary system which is 8 plus 4 is 12. similarly for plus 5 what we do is that we represent this as a regular unsigned binary integer which is 0 1 0 1, and for minus 3 well is very simple we just to take a look at the formula over here, and it is representation will be 16 minus absolute value of 3 which is plus 13, and this is plus 13 is 8 plus 4 plus 1 which is 12 plus 1 equals 13.

(Refer Slide Time: 27:49)

Properties of the 2's Complement Notation

- * Range of the number system :
 - * $-2^{(n-1)}$ to $2^{n-1} - 1$ *4-bit (-8 to +7)*
- * There is a unique representation for 0
→ 000000 ✓
- * msb of $F(u)$ is equal to $\text{SgnBit}(u)$
 - * Refer to the number circle
 - * For a +ve number, $F(u) < 2^{(n-1)}$. MSB = 0
 - * For a -ve number, $F(u) \geq 2^{(n-1)}$. MSB = 1

McGraw Hill Education

Now, let us considered properties of the 2's complement notation; sotypically in any N notation the first property that we look at is the range of the number system. So, what we yet seen is that for a 4 bit number system, the range was between minus 8 to plus 7, this can be extended to an bit number system, where the ranges between minus 2 to the power n minus 1. So, you can substitute the value of 4 over here, and I will find it is minus 8 to plus 7.

So, this is the first property. So, one thing that we can infer from here is that we have one more negative integer than the number of positive integers that is because we have to accommodate for 0 also, there is a single unique representation for 0 which is all 0s, which is a very good thing. This is something that we wanted the most significant. So, third property are the most significant bit of $F(u)$ which is the representation of a sign number in this notation, is equal to the sign bit of u .

So, why is this case? Let us go back to the number circle and the answer is there. So, if we consider all the positive numbers, their most significant bits 0, which is the sign bit of the number. If you consider all the negative numbers written in red, their most significant bit or sign bits plus 1 which is what we get to see for all the number over here. So, it is very easy to find if a number in the 2's complement notation is positive or negative, we just take a look at it is M S B, the most significant bit.

So, alternatively this is more of a generic proof; that is $F(u)$ is less than 2^{n-1} and minus 1. The M S B is 0 you can answer it is positive and for a negative number it needs to be greater than equal to 2^{n-1} . It needs to cross the breakpoint in the number circle, this is a breakpoint and this find the M S B becomes equal to 1, thus the number is negative.

(Refer Slide Time: 30:15)

Properties - II

- * Every number in the range $[-2^{(n-1)}, 2^{(n-1)} - 1]$
 - * Has a unique mapping
 - * Unique point in the number circle
- * $a \equiv b \rightarrow (a = b \text{ mod } 2^n)$
 - * \equiv means same point on the number circle
- * $F(-u) \equiv [2^n - F(u)]$
 - * Moving $F(u)$ steps counter clock wise is the same as moving $2^n - F(u)$ steps clockwise from 0

Handwritten notes:

$F(-u) = 2^n - u$

$4 - \text{bit}$

$3 \equiv 19 \text{ mod } 16$

$F(-4) = 16 - F(4) = 16 - 4 = 12 \text{ (1100)}$

Now let us look at some more properties; say every number in the range in the range of the number system has a unique mapping, because it has a unique point in the number

circle. So, why is this called the 2's complement system? We will take a look at it in the next two properties, but first let us define something simple. We define this point, where this operator called equivalence with 3 arrows. I am sorry 3 horizontal lines, this means that they refer to the same point on the number circle or alternatively a is equal to $b \bmod 2$ to the power n and so we can see if why. So, let us consider a 4 bit number system and so let us consider 0.3, if we add 16 to 3 then we just need to walk 16 steps in the clockwise direction, will again come back to 3. So, essentially 3 is equivalent to 19 in the system, the reason being that 3 is equal to $19 \bmod 16$, for 16 is 2 to the power 4.

So, the advantage of having this particular notation is that we can say that a lot of numbers on the number line are equivalent, because there is only one representation for them on the number circle, and out of them only 1 is in the range of the number system and the others are not. So, now, let us come to the first serious property of the 2's complement notation.

So, let us take a number u . So, we want to find the notation of the number minus u . So, this is equivalent to where this is the same operator that we have defined 2 to the power n minus F of u . So, let me first give an example and then we will look at the proof of this. So, consider the representation of minus 4, this is equal to 16 minus the representation of plus 4; the representation of plus 4 is just 4. So, this is essentially equal to 12 binary representation which is 1 1 0 0.

So, it is very easy to find the representation of the negative of any number simply by subtracting it from 2 to the power n , and this is also why this is called a 2's complement number system, because we are subtracting it from 2 to the power n that is the reason.

Now, if you want to take a look at a proof of this, this can easily be proven again with the number circle method. So, this is how we will define a number circle with a breakpoint over here. So, what we want to do is that if you want to find F of minus u , this is same as moving $F u$ steps counter clockwise is this. So, basically moving $F u$ steps in a counter clockwise direction is the same as moving 2 to the power n minus $F u$ steps in a clockwise direction starting from 0.

So, what we are seeing is that assume. So, let us consider if u is positive, then we will have F of minus u is 2 to the power n minus u . So, what we need to do is that the location for u and the number circle let us at this point and the location of minus u is let see this

point. So, F of minus u we can get by essentially starting at 0 and walking 2 to the power n minus u steps in the clockwise direction, which will bring us exactly over here, because if you walk 2 to the power n steps will again come back at the origin, and at this point will exactly reach 2 to the power n minus u , which is exactly the same as walking minus u steps in the counter clockwise direction.

So, what we can see, that if u is positive this relationship holds, similarly if u is negative it is also easy to prove the same relationship, this I leave for the reader. The important point to note in this number circle based logic, is that walking u steps in a certain direction is same as walking 2 to the power n minus u steps in a opposite direction. This is a very important point to understand for all of us that in a number circle. So, when you are walking in one direction you are incrementing, walking in the other direction you are decrementing.

, when we, let us see want to compute the notation of minus u F of minus u . So, minus u is basically means that we start at the origin and we move u steps in the counter clockwise direction, which is exactly the same as we move F u steps in the clockwise. I am sorry which essentially means that we move 2 to the power n minus F u steps in the clockwise direction to reach the same point.

(Refer Slide Time: 36:04)

Prove: $F(u+v) \equiv F(u) + F(v)$

- * Start at point u
 - Its index is $F(u)$
 - If v is +ve,
 - * move v points clockwise. We arrive at $F(u+v)$.
 - * Its index is equal to $(F(u) + v) \bmod 2^n$.
 - * Since $v = F(v)$, we have $(F(u+v) = (F(u) + F(v)) \bmod 2^n)$

$$F(u+v) \equiv F(u) + F(v)$$

$$F(5) = F(2) + F(3)$$

$$= 0010 + 0011$$

$$= 0101$$

Mc Graw Hill Education

So, now let us take a look at some of the basic properties of a 2's complement number system, and see how you know what is the real advantage of this. So, actually the 2's


complement system has some wonderful properties; mathematical properties and that is very interesting. So, let us do one thing, let us start at a point u . So, what we want to prove is a basic rule for addition that F of u plus v , where we adding 2 numbers their representation on the number circle the point is the same as F u plus F v . So, this basically means that we can add the representations of u and v . We just do a simple binary addition and that will be the representation of u plus v .

So, what we do? We start at point u it is index is F u . So, let us first assume that v is positive. So, then we move v points clockwise, and will arrive at F u plus v . So, it is index will be equal to F u plus v mod 2 to the power n and since v is equal to F v . We will have this relationship which is exactly equal to this relationship, is just equal to represented by, sorry substituted by the equivalent sign. So, we will just have for positive v , F of u plus v is exactly the same as F u plus F of v .

So, let me give an example to explain this point let us assume you want to add 2 plus 3 in binary. So, the representation of this would be 0 0 1 0 plus 0 0 1 1 right. So, basically we can say that F of 5 which is 2 plus 3 is equal to F 2 plus F 3, which is equal to 0 0 1 0 in binary plus 0 0 1 1 which is 0 1 0 1.

(Refer Slide Time: 38:24)

Prove : $F(u+v) \equiv F(u) + F(v)$

- If v is -ve, 
- move $|v|$ points anti-clockwise.
- Same as moving $2^n - |v|$ points clockwise.
- We arrive at $F(u+v)$.
- $F(v) = 2^n - |v|$
- The index $-F(u+v)$ is equal to:
 - $(F(u) + 2^n - |v|) \bmod 2^n = (F(u) + F(v)) \bmod 2^n$

Handwritten binary example:
 $2 + (-4) = F(2) + F(-4)$
 $= 0010 + 1100$
 $= 0110 = F(-2)$

Mc Graw Hill Education

Now, let us consider v to be negative. So, we consider v to be negative what do we need to do? We need to start at the point u on the number circle with representation F u , and we need to move modulus of v points anticlockwise. So, this is the same as moving 2 to

the power n minus $\text{mod } v$ points clockwise. So, we will then arrive at $F u$ plus v since $F v$ is equal to 2 to the power n minus $\text{mod } v$, the index $F u$ plus v this number will be equal to, $F u$ plus 2 to the power n minus $\text{mod } v$ mod to the power N of course, which is equivalent to $F u$ plus $F v$ mod 2 to the power n , so what we see that this relationship holds for both positive v as well as negative v .


So, let us consider again an example, let us assume that we want to compute 2 plus minus 4 in binary. So, the representation of 2 plus minus 4 what we are seeing is F of 2 plus F of minus 4. So, F of 2 if we consider a binary 0 0 1 0; if I consider F of minus 4 that is essential essentially equal to F of 12, 16 minus 4 which is 0 0 1 0 plus 1 1 0 0. So, let us proceed an, add no problem. So, what we get is we get 1 1 1 0, which is actually the representation of. So, this is plus 14, because this is 4, I am sorry 2 plus 4 plus 8 which is plus 14, which is actually F of minus 2, which is exactly the answer that we needed to get 2 plus minus 4 is minus 2. So, what we can see is that both for positive v as well as negative v , if we need to add two numbers for the represented in the 2's complement binary notation, we just simply need to add them and the final result will be the representation of, what the final result should be. Write the result of this addition over here will be the representation of what the final result which is u plus v should be.

(Refer Slide Time: 41:09)

Subtraction

- * $F(u-v) \equiv F(u) + F(-v)$
 $\equiv F(u) + 2^n - F(v)$

$F(3-5) = F(3) + 16 - F(5)$
 $= 3 + 16 - 5 = 14 = F(-2)$
- * Subtraction is the same as addition
- * Compute the 2's complement of $F(v)$


38


Now, let us consider subtraction let us assume we want to subtract u minus v . So, this from the previous theorem is $F(u) + F(v)$, which is $F(u + 2^n - v)$.

So, subtraction in a sense is the same as addition, you just compute the 2's complement of v . Say let us consider an example let us see that I want to subtract in a 4 bit number system, 3 minus 5. So, this I can write $F(3) + 2^4 - 5$ which is $16 - 5 + F(3)$. So, we will be writing in both binary and decimal notations the notation needs to be sort of inferred from the context. So, in this case $F(3)$ is equal to 3. So, I am not writing it in binary to just keep the representation simple and $F(5)$ is equal to 5.

So, this is equal to 14 which is equal to $F(14)$; this is exactly what we wanted to prove. So, one thing that we see is, that adding or multiplying, I am sorry adding or subtracting 2's complement numbers is actually very easy, for addition we just go ahead and add, and if we want to subtract, we essentially added with its 2's complement.

(Refer Slide Time: 42:53)

Prove that :



$$F(-6) = 16 - 6 = 10$$

$$F(2 \times (-3))$$

$$\equiv F(2) \times F(-3)$$

$$\equiv 2 \times 13$$

$$\equiv 26 \equiv 26 \pmod{16}$$

$$\equiv 10 = F(-6)$$

* Prove that :

$$\rightarrow F(u * v) \equiv F(u) * F(v)$$

$$u * v \equiv u * v$$

$$|u| * |v| \equiv (2^m - |u|) * (2^n - |v|)$$

$$\equiv 2^{m+n} - 2^m |v| - 2^n |u| + |u| * |v|$$

$$\equiv |u| * |v|$$

Mc
Graw
Hill
Education

So, here is a trick question food for thought; does a similar relationship hold for multiplication as well right? This is something that needs to be proven and this actually does hold. So, this actually does hold and this is a very nice property of the 2's complement notation does not hold for division, but at least for multiplication this property definitely does hold, and this is very useful property. So, let us may be consider an example.

So, let us consider you want to multiply 2 times minus 3, we claim that this is equivalent to F of 2 times F of minus 3; say F of 2 is equal to 2, because a representation of 2 is 2 on a number circle, and similarly F of minus 3 is actually 13, because it is 16 minus 3. So, this is equivalent to 26 which is equivalent to 10, because we take always mod 16. So, mod 16 means consider the 4 lower bits and so this is equivalent to 10 or I can write $26 \bmod 16$, which is essentially equal to 10, which is equal to F of minus 6 why minus 6? Because F of minus 6 is equal to 16 minus 6 is equal to 10. So, this is exactly what we had set out to prove that F of 2 times minus 3 is F of minus 6 and a low and behold we have a proof over here.

So, now the question is that we want to prove this result for any arbitrary u and v . So, let me prove some of the sub cases, and some cases I will leave as an exercise to the reader can consider u and v both are positive u both are positive, then we have F of u times v is just u times v of course, when there are no overflows, we will discuss that later multiplied by u times v which is really true. Considered the fact that both u and v are negative; so then let us look at say if u and v both are negative, the product is positive. So, the RHS still u multiplied with v . Whereas the LHS is 2 to the power n . So, maybe I can just write it mod u times mod v just to make things straight forward.

So, since we take all of these numbers mod 2 to the power n . So, why do we do that, because the range the number system is n bits, more than that we cannot represent. So, since we take everything mod 2 to the power n , we can cancel out all of these terms. So, since this term gets cancelled and this term gets cancelled, what we are left with is mod u multiplied with mod v and thus LHS is equal to RHS. The only other case that we need to prove is when u is positive and v is negative, and when u is negative and v is positive is an exactly symmetric case, so we just need to prove 1. So, you need to. So, the reader needs to prove the case when u is positive and v is negative, the reader needs to prove this result and this I leave as an exercise.

(Refer Slide Time: 49:03)

Sign Extension

- * Convert a n bit number to a m bit 2's complement number ($m > n$)
- * **+ve** [0000 0100]
 - * Add $(m-n)$ 0s in the msb positions
 - * Example, convert 0100 to 8 bits \rightarrow 0000
- * **-ve**
 - * $F(u) = 2^n - |u|$ (n bit number) system
 - * Need to calculate $F'(u) = 2^m - |u|$

Handwritten annotations: 2^4 above 0100, 2^8 above 0000, 2^4 above 1100, $2^6 - 4$ above 1100, $2^8 - 4$ above 1100.

McGraw Hill Education logo in the bottom left corner.

Now, let us take a look at one more trick in the realm of 2's complement numbers it is called sign extension. So, suppose we convert n bit number to a m bit 2's complement number right. So, the idea here is that, assume that this is a 4 bit number and you want to convert it to an 8 bit number. So, what do you do? So, assume that the number is positive. So, in almost all proofs we sort or divide it into two sub cases in one sub case the number is positive, and in the other sub case the number is negative.

So, let us consider that I have a number of the form in a plus 4 represented in binary 0 1 0 0 and I want to convert it into an 8 bit positive number, all that I need to do is that in the 4 M S B positions, most significant positions at 4 0s. So, this is a valid representation in a 8 bit number system.

Now, let us assume that the number is negative. So, let us consider minus 4. So, minus 4's representation is 1 1 0 0 in the 2's complement system. So, we need to calculate what would be the representation of minus 4 in a 8 bit number system right. So, minus 4 how do we get this number? This is actually the decimal form is 12 and see 16 minus 4, in 8 bits we need to compute 2 to the power 8 minus 4. So, 2 to the power 8 is 256 minus 4.

(Refer Slide Time: 51:02)

Sign Extension - II

$$\begin{aligned}
 & * 2^m - u - (2^n - u) \\
 & = 2^m - 2^n \\
 & = 2^n + 2^{(n+1)} + \dots + 2^{(m-1)} \\
 & = \underbrace{11110000}_{m-n} \underbrace{0000}_n
 \end{aligned}$$

$F'(u) = F(u) + 2^m - 2^n$

**Mc
Graw
Hill
Education**

$2^4 - 2^2 = 12$
 $= 2^2 + 2^3$
 $= 4 + 8$
 $F_8(-u) = 2^8 - u$ $m=8$
 $F_4(-u) = 2^4 - u$ $m=4$
 $F_8(-u) = \underbrace{2^4 - u}_{m-n \text{ bit}} + \underbrace{11110000}_{m-n}$
 $= \underbrace{1111}_{m-n} \underbrace{0000}_n$
 $F_4(-4) = 1100$ $(2^4 - 4)$
 $F_8(-4) = 11110100$

So, let us do a little bit of algebra. So, let us assume. So, this is the representation of number in the m bit number system. This is the representation of the number in the n bit number systems. So, let us just subtract; if you subtract we get 2 to the power m minus 2 to the power n. So, if we would actually subtracted what we would find is the result is 2 to the power n plus 2 to the power n plus 1 till 2 to the power M minus 1.

So, maybe just giving an example; let us see you want to subtract 2 to the power 4 minus 2 to the power 2, which is 16 minus 4 which is 12 right. So, the claim here is that, this is equal to 2 to the power 2 plus 2 to the power 3 which is 4 plus 8. So, this is a very easy identity to prove and so I leave this to the reader, but the interesting part is that if I take this expression, and I convert it into binary what I would get is, that I would get n 0s and the remaining m minus n 1's all right. So, this is exactly what I would get that I would get n 0s and m minus N 1's. Now assume that the representation in a 4 bit number system of a number of the form minus u right for u is positive is 2 to the power n minus u. And let us see an 8 bit number system we have this equal to 2 to the power m minus u where n is 4 and m is equal to 8.

So, what we have; we have this relationship over here that F of 8 minus u, is actually equal to 2 to the power n minus u, which is again an N bit number right, which is an n bit number plus this expression over here, for the first n bits are all 0s, and the remaining 1s are all 1s. So, I would consider a relationship of a num, I am sorry a number of this type

which is m minus n 1s and the rest n are 0s right. So, if I add this number plus this number what I will get is that the first m minus n positions, you know will still be sorry the first m minus n positions will still be 1s. And then the remaining positions over here will be occupied by whatever is a representation of 2 to the power n minus u right, for the remaining n positions. Since this number is negative the M S B of this part will also be 1 and the remaining will be 1s all right.

So, the important point to note is that, whatever is a representation in the 4 bit number systems. So, let us again consider F of minus 4. F of minus 4 in a 4 bit number system was 1 1 0 0. Say F of minus 4 in an 8 bit number system will first p 1 1 0 0 and the rest all the positions will be equal to 1. So, the readers can manually verify this is exactly what comes why another calculation which is 256 minus 4. So, this is also the representation of 252 all right. And so what we say is that they sign bit over here we can think of it as getting replicated across the rest of the new positions. This is exactly what was happening for positive numbers that the sign bit for the 4 bit number was getting replicated across the new positions, that new positions to the left; and here also the sign bits being replicated towards the new positions on the left.

(Refer Slide Time: 55:28)

Sign Extension - III

- * To convert a negative number :
 - * Add $(m-n)$ 1s in the msb positions
- * In both cases, extend the sign bit by :
 - * $(m-n)$ positions

$4 \rightarrow 8$
 0000 0010
 1111 1011

Mc
 Graw
 Hill
 Education

So, this trick is called sign extension that to convert a negative number from n bit number system to an m bit number system, you add m minus n 1s in the most significant M S B positions. So, what we can do just summarizing and closing the discussion is that

in both cases extend the sign bit by m minus n positions. For example, (Refer Time: 55:57) if I want to convert from a 4 bit to an 8 bit system number system, and my number is 0 0 1 0 all that I do is, I add 4 0s to the left of it if I have one number of the form 1 0 1 1 I just had 4 1s to the left of it. So, essentially take a look at the sign bit, whatever it is I just extend the sign bit replicated across the positions to the left.

(Refer Slide Time: 56:26)

The Overflow Theorem

- * Add : $u + v$
- * If $uv < 0$, there will **never** be an **overflow** 4-bit
5+5=10
- * Let us go back to the number circle
 - * There is an overflow only when we cross the break-point
- * If $uv = 0$, one of the numbers is 0 (no overflow)
- * If $uv > 0$, an **overflow is possible**

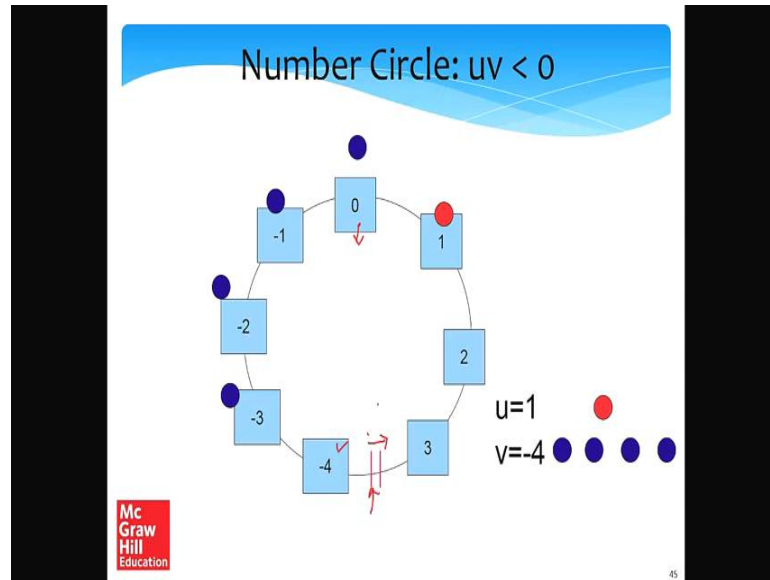
Mc
Graw
Hill
Education

Now, let us consider one more point so which is called an overflow theorem. So, the overflow theorem is something like this. So, let us assume that in a 4 bit number system, we want to add 5 plus 5. So, 5 plus 5 the answer is 10, but 10 cannot be represented in 4 bits, the reason being that the largest number that can be represented, is actually plus 7. So, in this case we say that we have an overflow, which means that the number cannot be represented in the particular number system. Similarly if I add minus 5 and minus 5 it will be minus 10, but the range of the 4 bit number system is only till minus 8. So, we cannot do the addition, as a result will also have an overflow in this case.

So, the overflow theorem is something like this, that if I am adding u plus v and if $u v$ is less than 0, which means one of them one of the numbers u or v is positive, and the other one is negative we will never have an overflow. So, to prove this let us go back to the number circle. So, there will be an overflow only when we cross the breakpoint, otherwise you are within the range of the number circle. So, $u v$ is equal to 0, one of the numbers is you know if $u v$ is equal to 0, means one of the numbers is 0. So, one of the

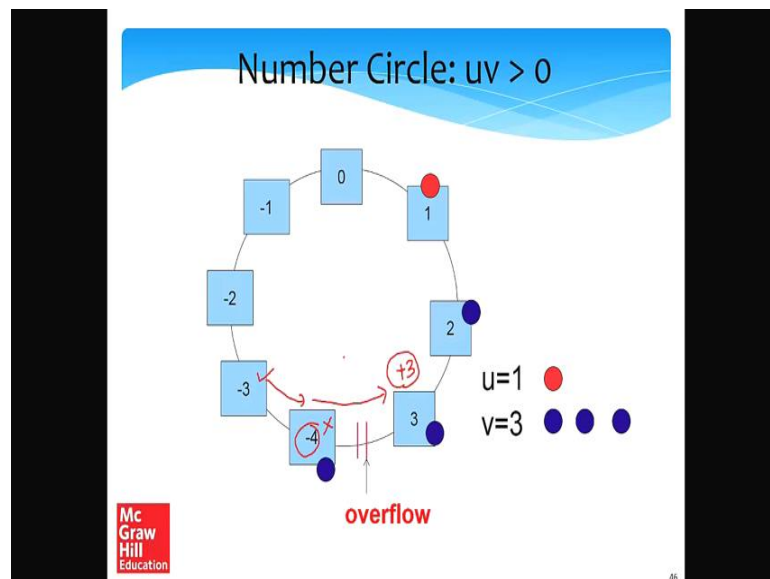
numbers is 0 then not be any overflow. If $u \cdot v$ is greater than 0 and overflow is possible, and we just saw an example of this, where we are adding 5 and 5 or adding minus 5 and minus 5.

(Refer Slide Time: 58:17)



Now, $u \cdot v$ is less than 0, let us see what will happen, let us say that you want to add 1 and minus 4; to we start at 1, and we count 4 positions in the anticlockwise direction. So, we can clearly see that we are not crossing the breakpoint. So, there is no overflow right this is the breakpoint.

(Refer Slide Time: 58:41)



But if we had u and v both as positive, and in this simple 2 bit number system that we have we can clearly see that we are crossing the breakpoint, and there will be an overflow. So, I am adding 1 and 3. So, to add 3 I need to move 3 steps clockwise, and I will cross the breakpoint and so 1 and 3 is not minus 4 it is actually plus 4 is the wrong answer. So, I am having an overflow. So, what are the conditions for an overflow; if u v is less than equal to 0 will never have an overflow.

(Refer Slide Time: 59:22)

Conditions for an Overflow

- * $uv \leq 0$
 - * Never
- * $uv > 0$ (u and v have the same sign)
 - * The sign of the result is different from the sign of u

Mc
Graw
Hill
Education

47

If u v is greater than 0 means say the same sign we can have an overflow, but when will we have an overflow when the sign of the result is greater than, is different from the sign of both u and v right? In this case u and v both are positive and we add them. So, if the result is negative we know there is an overflow. Similarly if we let us say we are adding minus 3 and minus 2. So, I can start it minus 3 to add minus 2, I need to travel 2 steps in the anticlockwise direction. So, I come here and then I come here right and so the answer that I get is plus 3 which can never be the answer, because minus 3 and minus 2 are both negative numbers. So, the answer cannot be positive.

So, then also I will find that I am having an overflow. So, in the book there is a very detailed proof of this result and. So, in the slides you know this is a last slides we are not presenting a result, but I will just give you an overview of the proof, but the detailed proof is there in the book, and I would request all readers to go to the book and take a

look at the proof. So, the proof goes somewhat like this. So, let us go back to this example.

So, let us assume that you know with no loss of generality, u is positive and v is negative right $u + v$ is less than 0. So, if you start at a positive point to have an overflow I need to sort of cross this breakpoint right; so to cross this breakpoint. So, let me assume a contra positive that you can have an overflow. So, the best possible way that I can have one you know to maximize my chances. I would like to meet u as small as possible, and the absolute value of v as large as possible. So, the smallest value of u that is not negative is 0, and the largest absolute value of v is actually 4. So, the v will become minus 4.

So, if I add 0 plus minus 4, I will still come over here, which is in the range of the number system and I will not cross the breakpoint right. So, basically whatever I do, I will not be able to cross the breakpoint, and thus I will not have an overflow. So, this is a very high level hand wavy kind of proof, but better proof and a very rigorous mathematical proof is there in the book that is the reason I am asking you to go there, and similarly if you know if I consider any other examples, let us I start at 2, I need to travel more than 7 steps to actually cross the breakpoint, and sort of need to add minus 7, and since minus 7 is not in the range of the number system will not have an overflow.

So, this is why if one of the number is positive, and other is negative or the other is 0. We will not have an overflow; but if both the numbers has a same sign we can always cross the breakpoint as we have seen. Now reader can argue that is it possible for the two numbers to have the same sign and I will cross the breakpoint have an overflow with the result will also have the same sign, which means maybe I start from this point, I add a positive number and I walk all the way up, and then I again my result is in the positive half of the number circle this will never happen the reason this will never happen is. So, my best cases that I start at let us say 3 and add 3. So, one step 2 and 3 the result is still negative, on in the other case I can start at minus 4, and add minus 4; in this case one step 2 step 3, and 4 right, because I am walking anticlockwise now, the result is still non negative.

So, what we see is that respective or whatever you do if I am adding two numbers with the same sign, and the result is of a different sign of then opposite sign I can always infer an overflow, which is exactly what the overflow theorem is trying to tell you that if one

of the numbers is 0, will never have an overflow, a 1 is positive the other is negative will never have an overflow, only when both the numbers u and v have the same sign would we actually have an overflow in a condition is the sign of the result should be different from the sign of both the numbers.