2-Arithmatic Operations

Binary Arithmetic

We are very familiar with different arithmetic operations, *viz*. addition, subtraction, multiplication, and division in a decimal system. Now we want to find out how those same operations may be performed in a binary system, where only two digits, *viz.* 0 and 1 exist.

-Binary Addition

it is a key for binary subtraction, multiplication, division. There four rules of

the binary addition.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1+1=10) i.e. 0 is write in

the given column and a carry of 1 over to the next column.

Exp1:

0011010 + 001100 = 00100110

	carry
1 1	
0 0 1 1 0 1 0	$= 26_{10}$
+ 0 0 0 1 1 0 0	$= 12_{10}$
0 1 0 0 1 1 0	$= 38_{10}$

- Binary Subtraction

Subtraction and Borrow, these two words will be used very frequently for the binary subtraction. There four rules of the binary subtraction. There four rules of the binary subtraction.

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

Exp:

0011010 - 001100 = 00001110

	1 1	borrow
	0 0 ~~1 1~~ 0 1 0	$= 26_{10}$
	- 0 0 0 1 1 0 0	$= 12_{10}$
	0 0 0 1 1 1 0	$= 14_{10}$

- **Binary Multiplication**

Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. There four rules of the binary multiplication.

Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Exp:

0011010 x 001100 = 100111000

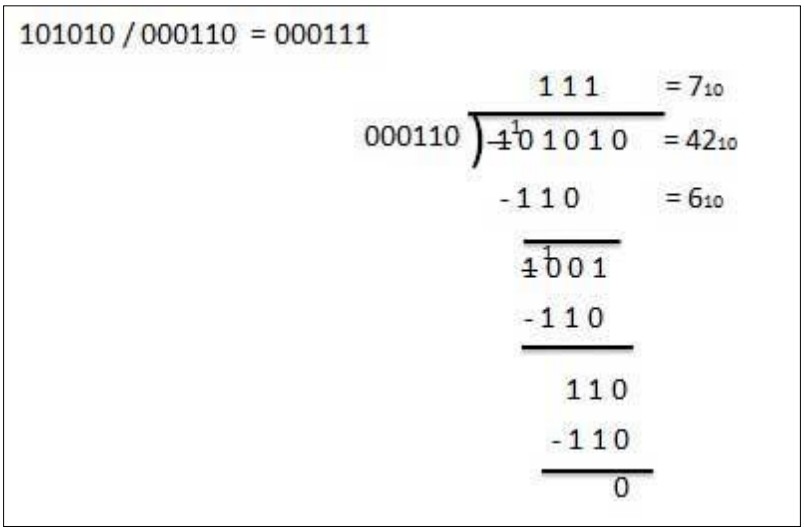
0 0 1 1 0 1 0	$= 26_{10}$
x 0 0 0 1 1 0 0	$= 12_{10}$
0 0 0 0 0 0 0	
0 0 0 0 0 0 0	
0 0 1 1 0 1 0	
0 0 1 1 0 1 0	
0 1 0 0 1 1 1 0 0 0	$= 312_{10}$

- **Binary Division**

Binary division is similar to decimal division.where, the division of two digits is as follows:

case	A ÷ B	Division
1	1 ÷ 1	1
2	0 ÷ 1	0

It is called as the long division procedure.



Home work :

1- $(11110)_2$ +$(1100)_2$ = $(?)_2$, 2-$(1011101)_2$ +$(1001011)_2$=$(?)_2$, 3-$(110)_2$ +$(101)_2$+$(10)_2$=$(?)_2$

4-$(10011)_2$ -$(1001)_2$ = $(?)_2$, 5- $(110.1)_2$ -$(100.10)_2$ = $(?)_2$, 6- $(111)_2$ - $(100)_2$ - $(01)_2$=$(?)_2$

7- $(101101)_2$ * $(10101)_2$ = $(?)_2$, 8- $(101.01)_2$ * $(11.01)_2$ = $(?)_2$, 9- $(110)_2$ * $(11)_2$ * $(01)_2$ = $(?)_2$

10- $(1100)_2$ ÷ $(11)_2$ = $(?)_2$, 11- $(1011010)_2$ ÷ $(11)_2$ = $(?)_2$, 12- $(1100101)_2$ ÷ $(100)_2$ = $(?)_2$

#Octal Arithmetic

This section describes octal arithmetic operations addition and subtraction.

Quick Preview **:**the following are the characteristics of an octal number system.

- Uses eight digits, 0,1,2,3,4,5,6,7.
- Also called base 8 number system
- Each position in a octal number represents a 0 power of the base (8). Example 80
- Last position in a octal number represents a x power of the base (8). Example 8x where
  x represents the last position - 1.

Example: Octal Number: 12570_8 , Calculating Decimal Equivalent:

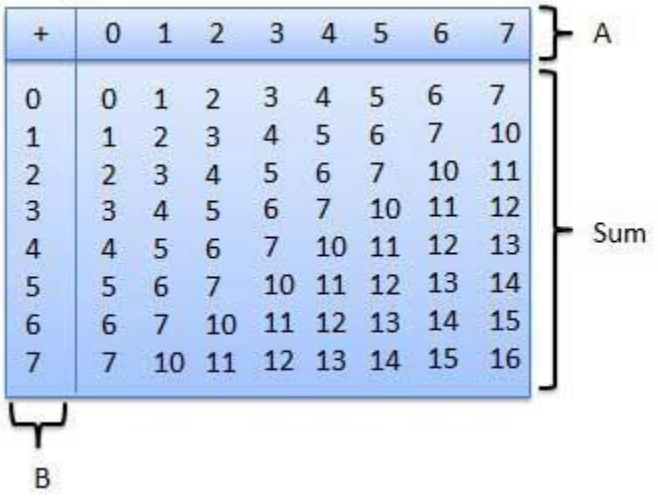
Step1: $12570_8 = ((1 \times 8_4) + (2 \times 8_3) + (5 \times 8_2) + (7 \times 8_1) + (0 \times 8_0))_{10}$

Step2: $12570_8 = (4096 + 1024 + 320 + 56 + 0)_{10}$
Step3: $12570_8 = 5496_{10}$

Note: 12570_8 is normally written as 12570.

- Octal Addition

Following octal addition table will help you greatly to handle Octal addition.



To use this table, simply follow the directions used in this example: Add: 6_8 and 5_8.Locate 6 in the A column then locate the 5 in the B column. The point in sum area where these two columns intersect is the sum of two numbers. $6_8 + 5_8 = 13_8$.

Example:

$456_8 + 123_8 = 601_8$

	1 1	carry
	4 5 6	$= 302_{10}$
+	1 2 3	$= 83_{10}$
	6 0 1	$= 385_{10}$

- Octal Subtraction

The subtraction of octal numbers follows the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}. In the binary system, you borrow a group of 2_{10}. In the octal system you borrow a group of 8_{10}.

Example:

$456_8 - 173_8 = 333_8$

	8	borrow
	$^{3}456$	$= 302_{10}$
	-173	$= 123_{10}$
	263	$= 179_{10}$

Home work :

1- $(123)_8 + (65)_8 = (?)_8$, 2-$(1740)_8 + (1234)_8 = (?)_8$, 3-$(16)_8 + (23)_8 + (10)_8 = (?)_8$

4-$(356)_8 - (43)_8 = (?)_8$, 5- $(4457)_8 - (3210)_8 = (?)_8$, 6-$(123)_8 - (65)_8 - (12)_8 = (?)_8$

Hexadecimal Arithmetic

This section describes hexadecimal arithmetic operations addition and subtraction.

Quick Preview :the following are the characteristics of a hexadecimal number system.

- Uses 10 digits and 6 letters, 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
- Letters represents numbers starting from 10. A = 10. B = 11, C = 12, D = 13, E = 14, F = 15.
- Also called base 16 number system
- Each position in a hexadecimal number represents a 0 power of the base (16). Example 16_0
- Last position in a hexadecimal number represents a x power of the base (16). Example $16_x$where x represents the last position - 1.

Example:

Hexadecimal Number: $19FDE_{16}$ Calculating Decimal Equivalent:

Step1: $19FDE_{16} = ((1 \times 16_4) + (9 \times 16_3) + (F \times 16_2) + (D \times 16_1) + (E \times 16_0))_{10}$
Step2: $19FDE_{16} = ((1 \times 16_4) + (9 \times 16_3) + (15 \times 16_2) + (13 \times 16_1) + (14 \times 16_0))_{10}$
Step3: $19FDE_{16} = (65536 + 36864 + 3840 + 208 + 14)_{10}$
Step4: $19FDE_{16} = 106462_{10}$

Note: $19FDE_{16}$ is normally written as 19FDE.

-Hexadecimal Addition

Following hexadecimal addition table will help you greatly to handle Hexadecimal addition.

To use this table, simply follow the directions used in this example: Add: A16 and 516.Locate A in the X column then locate the 5 in the Y column. The point in sum area where these two columns intersect is the sum of two numbers.

$A_{16} + 5_{16} = F_{16}$.

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

X

Sum

Y

Example:

$4A6_{16} + 1B3_{16} = 659_{16}$

1	carry
4 A 6	$= 1190_{10}$
+1 B 3	$= 435_{10}$
6 5 9	$= 1625_{10}$

-Hexadecimal Subtraction

The subtraction of hexadecimal numbers follows the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}. In the binary system, you borrow a group of 2_{10}. In the hexadecimal system you borrow a group of 16_{10}.

Example:

$4A6_{16} - 1B3_{16} = 2F3_{16}$

16	borrow
$^{3}4$ A 6	$= 1190_{10}$
-1 B 3	$= 435_{10}$
2 F 3	$= 755_{10}$

Home work :

1- $(5AE)_{16} + (94B)_{16} = (?)_{16}$, 2-$(52EB)_{16} + (1234)_{16} = (?)_{16}$, 3-$(10)_{16} + (20)_{16} + (11)_{16} = (?)_{16}$

4-$(FE)_{16} - (37)_{16} = (?)_{16}$, 5- $(BC66)_{16} - (2F3)_{16} = (?)_{16}$, 6-$(123)_{16} - (65)_{16} - (12)_{16} = (?)_{16}$

3-Representing Negative Numbers

In the "real world" of mathematics, computers must represent both positive and negative binary numbers. For example, even when dealing with positive arguments, mathematical operations may produce a negative result:

– Example: 124 – 237 = –113.

• Thus needs to be a consistent method of representing negative numbers in binary computer arithmetic operations.

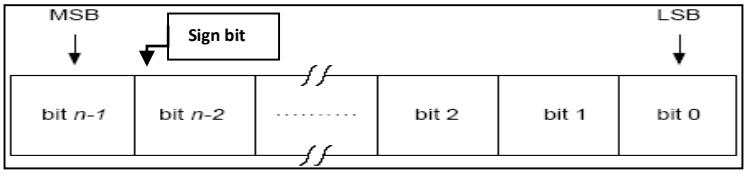
• Basically there are three types of representations of signed binary numbers— sign-magnitude representation, 1's complement representation, and 2'scomplement representations, which are discussed below.

1- Sign-magnitude representation.

In decimal system, generally a plus (+) sign denotes a positive number whereas a minus (–) sign denotes a negative number. But, the plus sign is usually dropped, and nosign means the number is positive. This type of representation of numbers is known as ***signed numbers.***

But in digital circuits, there is no provision to put a plus or minus sign, since everything in digital circuits have to be represented in terms of 0 and 1. Normally an additional bit is used as the *sign bit*. This sign bit is usually placed as the MSB. Generally a 0 is reserved for a positive number and a 1 is reserved for a negative number.

For example, an 8-bit signed binary number 01101001 represents a positive number whose magnitude is (1101001)2 = (105)10. The MSB is 0, which indicates that the number is positive. On the other hand, in the signed binary form, 11101001 represents a negative number whose magnitude is (1101001)2 = (105)10. The 1 in the MSB position indicates that the number is negative and the other seven bits give its magnitude. This kind of representation of binary numbers is called ***sign-magnitude representation.***



Example :

Signed Integer	Sign Magnitude
+2	0000 0010
+1	0000 0001
0	0000 0000
-1	1000 0001
-2	1000 0010

The drawbacks :to using this method for arithmetic computation are that a different set of rules are required and that zero can have two representations (+0, 0000 0000 and -0, 1000 0000).

Example 1.36. Find the decimal equivalent of the following binary numbers assuming the binary numbers have been represented in sign-magnitude form.

(a) 0101100 **(b)** 101000 **(c)** 1111 **(d)** 011011

Solution.

(a) Sign bit is 0, which indicates the number is positive.
Magnitude 101100 = $(44)_{10}$, **Therefore** $(0101100)_2 = (+44)_{10}$.

(*b*) Sign bit is 1, which indicates the number is negative.
Magnitude 01000 = $(8)_{10}$, **Therefore** $(101000)_2 = (-8)_{10}$.

(*c*) Sign bit is 1, which indicates the number is negative.
Magnitude 111 = $(7)_{10}$, **Therefore** $(1111)_2 = (-7)_{10}$.

(*d*) Sign bit is 0, which indicates the number is positive.
Magnitude 11011 = $(27)_{10}$, **Therefore** $(011011)_2 = (+27)_{10}$.

??? What Complements mean ?

complements are used in the digital computers in order to simplify the subtraction operation and for the logical manipulations. For each radix-r system (radix r represent base of number system) there are two types of complements

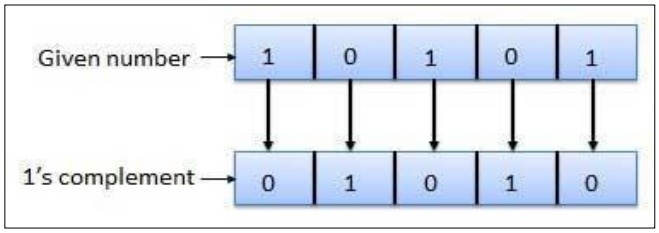
s.n	Complement	Description
1	Radix Complement	The radix complement is referred to as the r's complemen
1	Diminished Radix Complement	The diminished radix complement is referred to as the (r-1)'s complement

There for in Binary system complements has base r = 2. So the two types of complements for the binary system are following:

1' complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement.

Example:



ones' complement can be used to represent negative numbers. The ones' complement form of a negative binary number is the complement of its positive counterpart, which can be obtained by applying the NOT to the positive counterpart. Like sign-magnitude representation, ones' complement has two representations of 0: 00000000 (+0) and 11111111 (−0).

As an example, the ones' complement of 00101011 (43) is 11010100 (−43).

Note: The range of signed numbers using ones' complement in a conventional 8-bit byte is −127 to +127.

Signed integer	Unsigned integer	8 bit ones' complement
0	0	00000000
1	1	00000001
…….	…….	……
125	125	01111101
126	126	01111111
-127	128	10000000
-126	129	10000001
-125	130	10000010
…….	…….	……
-1	245	11111110
-0	255	11111111

+ 1's Complement Addition

To add two numbers represented in this system, we use the conventional binary addition, **but it is then necessary to add any resulting carry back into the resulting sum.** To see why this is necessary, consider the following example showing the case of the addition of −1 (11111110) to +2 (00000010).

Example: perform (7-3) using 1's complements method ?

```
Decimal      binary
  7    --->  111   ---->  111
- 3    --->  011        + 100
_____________________________
                        ¹ 011
                          +  1
                        _____
                          100  the result
```

Example: perform (8-12) using 1's complements method ?

```
 Decimal      binary
   8    ->    1000  -->    1000
 - 12   ->    1100  --> +  0011
 _________              _______
  - 4                    1011
```

2' complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

Note: 2's complement = 1's complement + 1
Example of 2's Complement is as follows.

The Two's complement representation allows the use of binary arithmetic operations on signed integers, yielding the correct 2's complement results.

Positive Numbers

Positive 2's complement numbers are represented as the simple binary.

Negative Numbers

Negative 2's complement numbers are represented as the binary number that when added to a positive number of the same magnitude equals zero.

Integer		2's Complement
Signed	Signed	

5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
``2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

Note: The most significant (leftmost) bit indicates the sign of the integer; therefore it is sometimes called the sign bit.

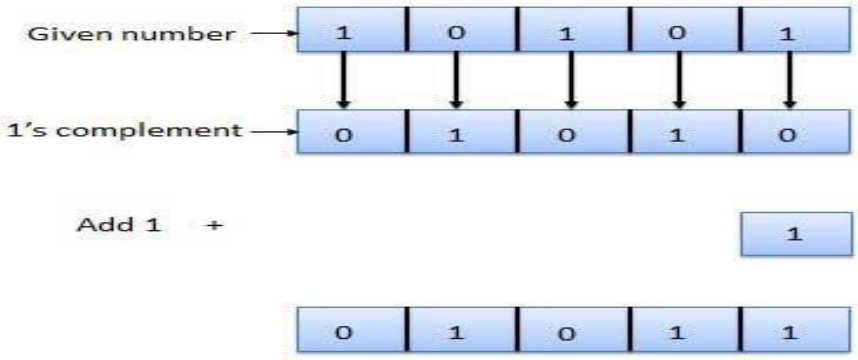
If the sign bit is zero,
then the number is greater than or equal to zero, or positive.

If the sign bit is one,
then the number is less than zero, or negative.

+ Calculation of 2's Complement

To calculate the 2's complement of an integer, invert the binary equivalent of the number by changing all of the ones to zeroes and all of the zeroes to ones (also called **1's complement**), and then add one.

Notes: The addition of ***n*-bit** signed binary numbers is straightforward using the 2's complement system. **The addition is carried out just as if all the numbers were positive, and any carry from the sign position is ignored**. This will always yield the correct result except when an overflow occurs. When the word length is *n* bits, we say that an *overflow* has occurred if the correct representation of the sum (including sign) requires more than *n* bits.



For example:

17 → **-17**

0001 0001(binary 17) → 1110 1111(two's complement -17)

NOT(0001 0001) = 1110 1110 (**Invert bits**)

1110 1110 + 0000 0001 = **1110 1111** (Add 1)

Home work :Now you try some :Find the two's complement for

a. - 11

b. - 43

c. - 123

+ 2's Complement Addition

Two's complement addition follows the same rules as binary addition.

For example:

5 + (-3) = 2 0000 0101 = +5

\+ 1111 1101 = -3

0000 0010 = +2

+ 2's Complement Subtraction

Two's complement subtraction is the binary addition of the minuend to the 2's complement of the subtrahend (adding a negative number is the same as subtracting a positive one).

For example:

7 - 12 = (-5) 0000 0111 = +7

\+ 1111 0100 = -12

1111 1011 = -5

Note: this suggests a new way to subtract in binary due to the fact that subtraction is defined in the following manner :

x-y= x+ (-y)

Note that a register of size eight can only represent decimal integers between $-2^{(8-1)}$ and $+2^{(8-1)}$ and, in general, a register of size *n* will be able to represent decimal integers between $-2^{(n-1)}$ and $+2^{(n-1)}$

EXAMPLE 2: *Subtract 29 from 23, as a computer would, using binary code.*

Again we use a register of size 8, so that 23 – 29 = 23 + (-29) becomes

0001 0111 + 1110 0011 = 1111 1010. (Verify both the binary form of –29 and the addition.) Note that no truncation of the leftmost bit is necessary here. The result is the *negative* (it starts with a 1) integer 1111 1010. This needs to be "translated" to change it back to a decimal (see the steps on how to do this in the box above). Hence, going backwards, 1111 1010 – 1 = 1111 1001. The complement of which is 0000 0110 which is 6 in decimal. Negating this we get -6 as expected.

Home work :Now you try some : subtract each , as a computer out , using a binary code using registers of size 8 :

a. 26 -15
b. – 31 - 6
c. 144 – 156
d. make up your own exercises as needed