

Representation of negative numbers

unsigned
binary $(0101)_2 = (5)_{10}$
Signed

Representation of Negative Numbers

In our traditional arithmetic we use the "+" sign before a number to indicate it as a positive number and a "-" sign to indicate it as a negative number. We usually omit the sign before the number if it is positive. This method of representation of numbers is called "sign-magnitude" representation. But using "+" and "-" signs on a computer is not convenient, and it becomes necessary to have some other convention to represent the signed numbers. We replace "+" sign with "0" and "-" sign with "1". These two symbols already exist in the binary system. Consider the following examples:

$(+1100101)_2 \rightarrow (01100101)_2$
 $(+101.001)_2 \rightarrow (0101.001)_2$
 $(-10010)_2 \rightarrow (110010)_2$
 $(-110.101)_2 \rightarrow (1110.101)_2$

$0101 = (+5)_{10}$
signe
+ve
magnitude
 $1101 = (-5)_{10}$
-ve
magnitude

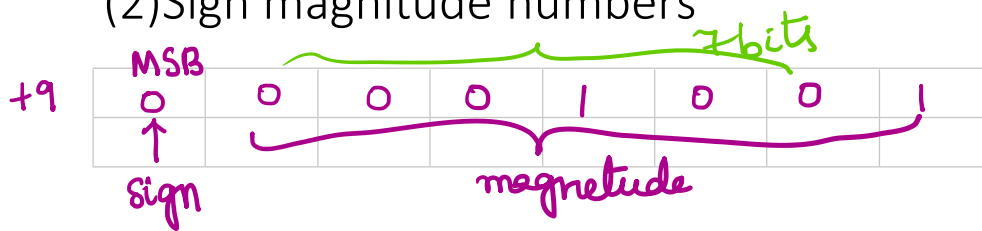
Sign magnitude Numbers

In the sign-magnitude representation of binary numbers the first digit is always treated separately. Therefore, in working with the signed binary numbers in sign-magnitude form the leading zeros should not be ignored. However, the leading zeros can be ignored after the sign bit is separated. For example,

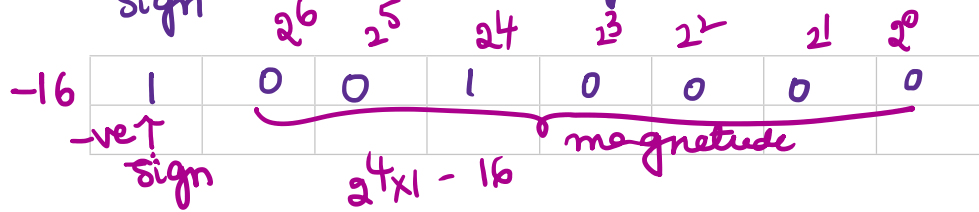
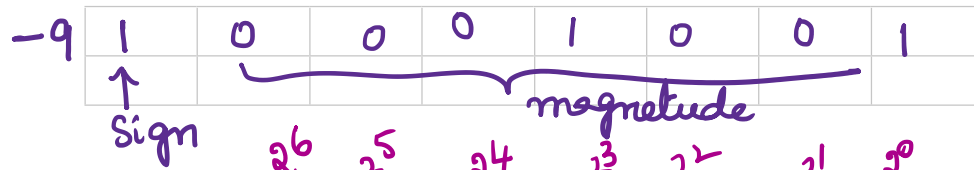
$$1000101.11 = -101.11$$

While the sign-magnitude representation of signed numbers appears to be natural extension of the traditional arithmetic, the arithmetic operations with signed numbers in this form are not that very convenient, either for implementation on the computer or for hardware implementation. There are two other methods of representing signed numbers.

(2) Sign magnitude numbers



9



10010101 = 8bit binary

+1 \Rightarrow 0001 } 4bit binary

Signed representation -1 \Rightarrow 1001 }
 ↑ sign magnitude

$-(\frac{n-1}{2}-1)$ to $+(\frac{n-1}{2}-1)$
 n = no. of bits

n = 4

$-\frac{4-1}{2}-1$ to $+\frac{4-1}{2}-1$

-7 to +7 ✓

0 to +7
 000
 001
 010
 011
 100
 101
 110
 111

+8 = (01000)
 ↑ sign magnitude

8 bit

8 bit representation

+8 = 0000 1000
 ↑ sign magnitude

-8 = 1000 1000
 ↑ sign

2 | 21
 2 | 10 1
 2 | 5 0
 2 | 2 1
 1 | 0

$(+21)_{10} = (10101)_2$

$(10101)_2$

010101 \Rightarrow 6 bit representation
 ↑ 5 bits will represent magnitude
 sign

$(+21)_{10} = 00010101$ 8 bit representation
 ↑ sign magnitude

$$-21 = \begin{array}{c} 10010101 \\ \hline \uparrow \quad \text{mag} \\ \text{sign} \end{array}$$

1. unsigned number range as

Tuesday, August 18, 2020 10:53 AM

$$0 \text{ to } 2^n - 1 \Rightarrow n = \text{no. of digits}$$

$$\text{4 bit } 0 \text{ to } 2^4 - 1 \Rightarrow 0 \text{ to } 15$$

$$\text{8 bit } 0 \text{ to } 2^8 - 1 \Rightarrow 0 \text{ to } 255$$

$$\text{3 bit } 0 \text{ to } 2^3 - 1 \Rightarrow 0 \text{ to } 7$$

signed binary numbers

$$\text{Range } -(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)$$

$$\text{8 bit} \rightarrow -(2^{8-1} - 1) \text{ to } (2^{8-1} - 1) \Rightarrow \underline{\underline{-127 \text{ to } +127}}$$

\downarrow
 $\underline{\underline{128-1}}$

more hardware

Two ways of representing signed numbers

1. sign magnitude form ✓

2. Complement form ✓

① radix (r)
(δ)
 r 's Complement

② Diminished radix represent
(δ)
($r-1$)'s Complement

$r = 10$ (Decimal)
① 10's Complement
 $r = 10$
($r-1$)'s Complement
② 9's Complement

$r = 16$ (Hexadecimal)
 r 's
16's Complement
($r-1$)'s
15's Complement

$r = 2$ (Binary)

2's Complement

($2-1$)'s Complement
1's Complement

$r = 8$ (Octal)

r 's = 8's Complement

($r-1$)'s = 7's Complement

Complements:

In digital computers to simplify the subtraction operation & for logical manipulation complements are used. There are two types of complements used in each radix system.

- i) The radix complement or r 's complement
- ii) The diminished radix complement or $(r-1)$'s complement