Electrical Engineering

Computer Fundamentals

Comprehensive Theory

with Solved Examples and Practice Questions

MADE EASY Publications

MADE EASY Publications

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016
E-mail: infomep@madeeasy.in
Contact: 011-45124660, 8860378007
Visit us at: www.madeeasypublications.org

Computer Fundamentals

© Copyright by MADE EASY Publications.
All rights are reserved. No part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photo-copying, recording or otherwise), without the prior written permission of the above mentioned publisher of this book.

First Edition : 2015
Second Edition : 2016
Third Edition : 2017
Fourth Edition : 2018
Fifth Edition : 2019

Sixth Edition : 2020

© All rights reserved by MADE EASY PUBLICATIONS. No part of this book may be reproduced or utilized in any form without the written permission from the publisher.

Contents

Computer Fundamentals

Chapter 6

Basic Concepts of Operating Systems ..125

Chapter 7

File System 145

Chapter 8

Networking Fundamentals 159

Chapter 9

Programming Methodology 168

■■■■

Computer Fundamentals

Goal of the Subject

Basic understanding of Computer Fundamentals includes, the study about the basic architecture of computer, its Central Processing Unit (CPU), how the memory organization is done, I/O organization, data representation, basics of operating systems, file systems, basics of networking and elements of programming languages.

CHAPTER 5

Data Representation

5.1 Fixed-Point Representation

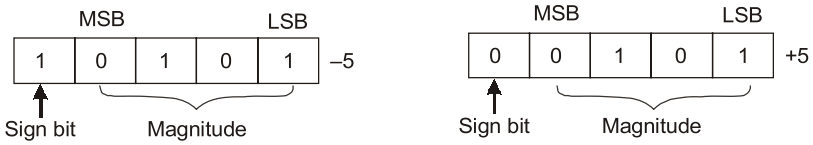
Positive integers, including zero, can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values. In ordinary arithmetic, a negative number is indicated by a minus sign and a positive number by a plus sign. Because of hardware limitations, computers must represent everything with 1's and 0's, including the sign of a number. As a consequence, it is customary to represent the sign with a bit placed in the leftmost position of the number. The convention is to make the sign bit equal to 0 for positive and to 1 for negative.

5.1.1 Integer Representation

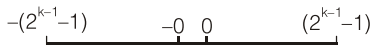
When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number. When the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of three possible ways:

1. **Signed Magnitude Method:** In this method number is divided into two parts: Sign bit and magnitude. If number is positive then sign bit will be 0 and if number is negative then sign bit will be 1. Magnitude is represented with the binary form of the number to be represented.

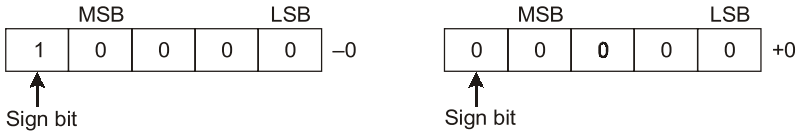
   Example: Let we are using five bits register. The representation of –5 and +5 will be as follows:



   Range of Numbers: For k bits register, MSB will be sign bit and (k – 1) bits will be magnitude. Positive largest number that can be stored is $(2^{k-1} - 1)$ and negative lowest number that can be stored is $-(2^{k-1} - 1)$.

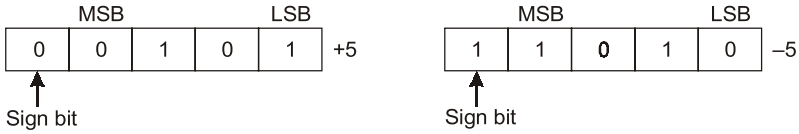


NOTE: Drawback of this system is that 0 has two different representation one is –0 and second is +0.



2. **1's Complement Method:** Positive numbers are represented in the same way as they are represented in sign magnitude method. If number is negative then it is represented using 1's complement. First represent the number with positive sign and then take 1's complement of that number.

Example: Let we are using five bits register. The representation of –5 and +5 will be as follows:



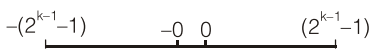
+5 is represented as it is represented in sign magnitude method. –5 is represented using the following steps:

(*i*) +5 = 0 0 1 0 1

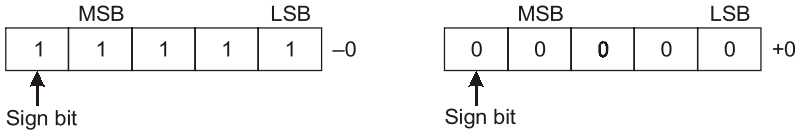
(*ii*) Take 1's complement of 0 0 1 0 1 and that is 1 1 0 1 0. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

Range of Numbers: For k bits register, positive largest number that can be stored is $(2^{k-1} - 1)$ and negative lowest number that can be stored is $-(2^{k-1} - 1)$.

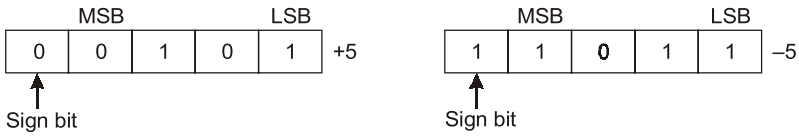


NOTE: Drawback of this system is that 0 has two different representation one is –0 and second is +0 same as in the case of sign magnitude method.



3. **2's Complement Method:** Positive numbers are represented in the same way as they are represented in sign magnitude method. If number is negative then it is represented using 2's complement. First represent the number with positive sign and then take 2's complement of that number.

Example: Let we are using five bits register. The representation of –5 and +5 will be as follows:



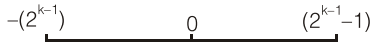
+5 is represented as it is represented in sign magnitude method. –5 is represented using the following steps:

(*i*) +5 = 0 0 1 0 1

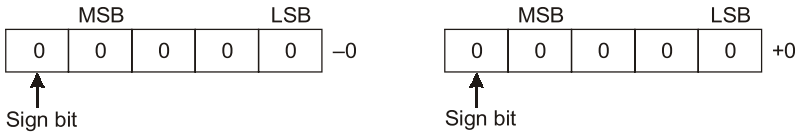
(*ii*) take 2's complement of 0 0 1 0 1 and that is 1 1 0 1 1. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

Range of Numbers: For K bits register, positive largest number that can be stored is $(2^{k-1} - 1)$ and negative lowest number that can be stored is $-(2^{k-1})$.



Advantage of this system is that 0 has only one representation for –0 and +0.



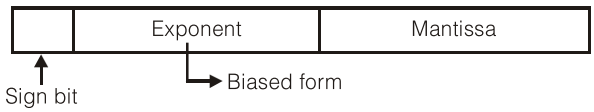
5.1.2 Floating-Point Representation

The floating-point representation of a number has two parts. The first part represents a signed, fixed-point number called the mantissa. The second part designates the position of the decimal (or binary) point and is called the exponent. The fixed-point mantissa may be a fraction or an integer. Floating-point is always interpreted to represent a number in the following form:

$$M \times r^e$$

Only the mantissa m and the exponent e are physically represented in the register (including their signs). A floating-point binary number is represented in a similar manner except that it uses base 2 for the exponent.

A floating-point number is said to be normalized if the most significant digit of the mantissa is one. For example, the 8 bit binary number 00011010 is not normalized because of the three leading 0's. The number can be normalized by shifting it three positions to the left and discarding the leading 0's to obtain 11010000. The three shifts multiply the number by $2^3 = 8$. To keep the same value for the floating-point number, the exponent must be subtracted by 3. Normalized numbers provide the maximum possible precision for the floating-point number. A zero cannot be normalized because it does not have a nonzero digit. It is usually represented in floating-point by all 0's in the mantissa and exponent.



Sign bit 0 means number is positive and sign bit 1 means number is negative. Exponent is always stored in biased form. Mantissa part stores the fractional part. For storing the exponent in biased form, bias number is calculated.

If k bits are used to represent exponent then

Bias Number = $(2^{k-1} - 1)$ and

Range of exponent is = $-(2^{k-1} - 1)$ to 2^{k-1}

Thus, if 7 bits are used for storing the exponent then bias number will be $(2^{7-1} - 1) = 63$ and exponent will ranges from –63 to 64.

Note that, here we always store exponent in positive. Biased number is also called excess number. Since exponent is stored in biased form so bias number is added to the actual exponent of the given number. Actual number can be calculated from the contents of the registers by using following formula:

$$\text{Actual Number} = (-1)^s (1 + m) \times 2^{e-\text{Bias}}$$

where s = sign bit

m = mantissa value of register

e = exponent value of register

Bias = bias number

Example: Represent +0.125 if 5 bits are used to represent exponent and 6 bits for mantissa.

Step 1: Calculate bias number

Bias Number = $(2^{k-1} - 1)$ and here k = 5

$= (2^{5-1} - 1)$

$= 15$

Step 2: Calculate binary number of the given decimal number

$0.125 \times 2 = 0.25$

$0.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$

$(0.125)_{10} = (0.001)_2$

The above binary is not normalized.

Step 3: Normalized the binary number

$0.001 = 1.0 \times 2^{-3}$

Step 4: Calculate exponent

Since bias number is 15 so it will be added to the exponent part

i.e., $-3 + 15 = 12$

2	12	0
2	6	0
2	3	1
	1	

$= 1100$

Step 5: Calculate mantissa

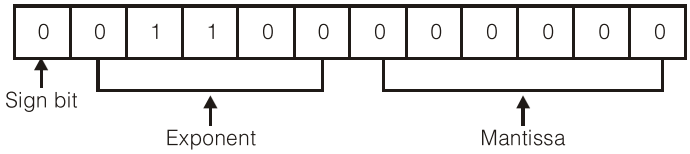
Number right to the binary point is 0 so mantissa will be all bits 0.

Step 6: Represent the number

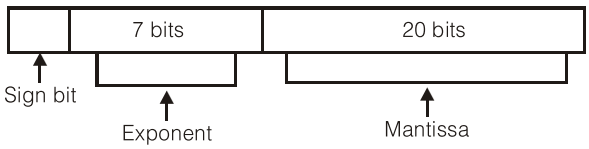
Since number is positive so sign bit will be 0.

exponent = 01100

mantissa = 000000



Range of Numbers: Let 7 bits are used to represent exponent and 20 bits are used to represent mantissa.



Bias number = $(2^{7-1} - 1) = 63$

Minimum positive number will be the number with lowest mantissa and exponent i.e.

$1.000000 \ldots\ldots 0000 \times 2^{-63}$

20 times

i.e., 1×2^{-63} where -63 is the lowest exponent.

Maximum positive number will be the number with largest mantissa and exponent i.e.

$$1.\underbrace{111111\ .....\ 1111}_{20\text{ times}} \times 2^{64}$$

i.e., $(2 - 2^{-20}) \times 2^{64}$ where 64 is the largest exponent.
Similarly, minimum negative number is $-(2 - 2^{-20}) \times 2^{64}$
and maximum negative number is -1×2^{-63}

$$\underbrace{-(2-2^{-20})\times 2^{64} \quad -1\times 2^{-63} \quad 1\times 2^{-63} \quad (2-2^{-20})\times 2^{64}}$$

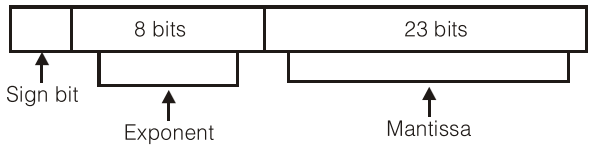
Floating-Point Number Density: As the floating-point number increases, the density of number decreases. So the maximum difference will be between the largest positive number and second largest positive number.

5.2 IEEE Floating-Point Number Representation

According to IEEE standard, floating-point number is represented in two ways:

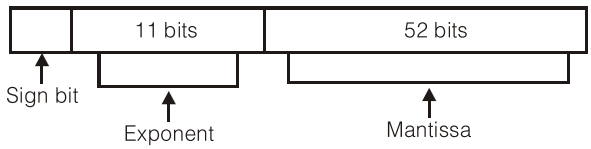
5.2.1 Single Precision

In single precision, 32 bits are used to represent floating-point number. Out of these 32 bits, MSB is used for sign bit, 8 bits for exponent and remaining 32 bits for mantissa. Hence bias number is 127.



5.2.2 Double Precision

In double precision, 64 bits are used to represent floating-point number. Out of these 64 bits, MSB is used for sign bit, 11 bits for exponent and remaining 52 bits for mantissa. Hence bias number is 1023.



In both single and double precision representations, all exponent bits are 0 or all exponent bits are 1 are not used for normal number representation.

Range of exponent will be $-(2^{k-1} - 2)$ to $(2^{k-1} - 1)$. Hence range of numbers in single precision is 2^{-126} to 2^{+127} and range of numbers in double precision is 2^{-1022} to 2^{+1023}.

5.2.3 Special Representation

1. All the exponent bits 0 with all mantissa bits 0 represents zero.
   (a) If sign bit is 0 then it represents + 0.
   (b) If sign bit is 1 then it represents – 0.
2. All the exponent bits 1 with all mantissa bits 0 represents infinity.
   (a) If sign bit is 0 then it represents $+\infty$.
   (b) If sign bit is 1 then it represents $-\infty$.
3. All the exponent bits 0 and mantissa bits non-zero represents denormalized number.
4. All the exponent bits 1 and mantissa bits non-zero represents error.

Here are the steps:

1. In this case, X is 1.01×2^2 and Y is 1.11×2^0.
2. The resulting exponent is 2 + 0 = 2
3. Multiplying 1.01 by 1.11 results in 10.0011.
4. Now, we have to renormalize 10.0011 to 1.00011 and increase the exponent by 1 to 3 .
5. The sign bit is 0 + 0 = 0.
6. We need to truncate 1.00011×2^3 to 1.000×2^3 and convert.

Product	Sign	Exponent	Fraction
X×Y	0	1010	000

Negative Values

Unlike floating-point addition, negative values are simple to take care of in floating-point multiplication. Treat the sign bit as 1 bit unsigned binary, and add modulo 2. This is the same as XORing the sign bit.

Summary

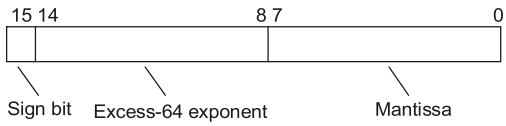
- Number representation $(N)_b = d_{n-1}\, d_{n-2} \ldots d_i \ldots d_1\, d_0 \cdot d_{-1}\, d_{-2} \ldots d_{-f} \ldots d_{-m}$.
- Each digit of a decimal number is represented by binary equivalent.
- In 4-bit binary formats: Total number of possible representation = 2^4 = 16, Valid BCD codes = 10 and Invalid BCD codes = 6.
- In 8-bit binary formats: Valid BCD codes = 100, Invalid BCD codes = 256 – 100=156.
- BCD is also called 8421 code.
- During arithmetic operation if invalid BCD is present then add (0110) to get correct result.
- It is a not a self complementing code.
- Excess-3 code is a 4-bit code. It can be derived from BCD code by adding "3" to each coded number. It is a "self-complementing code". It is the only code which is unweighted and self complementing.
- 2421,3321,4311 and 5211 all are self complementing codes where sum of weight is 9.
- Gray code also called "minimum change codes" in which only one bit in the code group changes when going from one step to the next. It is also called cyclic code or reflective code. Since error is minimum so also called "minimum error code".
- If a number system has base b then we can find its b's complement and $(b-1)$'s complement. To determine $(b-1)$'s complement subtract given number from maximum number possible to the given base i.e. $(r^n - 1)$. To determine b's complement, first determine $(b-1)$'s complement then add 1 to get b's complement of number.
- To convert decimal number into any other base b divide integer part with b and multiply fractional part with b. If $(X_3 X_2 X_1 X_0)_b = (A)_{10}$ then $A = X_0 + X_1 b^1 + X_2 b^2 + X_3 b^3$.
- In unsigned magnitude representation with 'n' bits, the possible integer values are [0 to $(2^n - 1)$].
- In a sign magnitude representation the range of number $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$.
- In 1's complement representation, the positive number are represented similar to positive number in sign magnitude, but for representing negative number, first consider positive number and then take 1's complement of that.

The exponent is in 2's complement representation and mantissa is in the sign magnitude representation. The range of the magnitude of the normalized numbers in this representation is

(a) 0 to 1 (b) 0.5 to 1
(c) 2^{-23} to 0.5 (d) 1 to $(2 - 2^{-23})$

Common Data Questions (29 and 30):

Consider the following floating point format



Mantissa is a pure fraction is sign-magnitude form.

Q.29 The decimal number 0.239×2^{13} has the following hexadecimal representation without normalization and rounding off

(a) 0D 24 (b) 0D 4D
(c) 4D 0D (d) 4D 3D

Q.30 The normalized representation for the above format is specified as follows. The mantissa has an implicit 1 preceding the binary (radix) point. Assume that only 0's are padded in while shifting a field. The normalized representation of the above number (0.239×2^{13}) is

(a) 0A 20 (b) 11 34
(c) 4D D0 (d) 4A E8

Q.31 The following bit pattern represents a floating point number in IEEE 754 single precision format:

1 10000011 10100000000000000000000

The value of the number in decimal form is

(a) –10 (b) –13
(c) –26 (d) None of these

Q.32 In the IEEE floating point representation the hexadecimal value 0x00000000 corresponds to

(a) the normalized value 2^{-127}
(b) the normalized value 2^{-126}
(c) the normalized value +0
(d) the special value +0

Student's Assignments Answer Key

1. (a)	**2.** (a)	**3.** (b)	**4.** (a)	**5.** (d)
6. (a)	**7.** (a)	**8.** (d)	**9.** (c)	**10.** (d)
11. (c)	**12.** (c)	**13.** (a)	**14.** (d)	**15.** (c)
16. (c)	**17.** (c)	**18.** (b)	**19.** (a)	**20.** (b)
21. (c)	**22.** (b)	**23.** (d)	**24.** (a)	**25.** (a)
26. (d)	**27.** (c)	**28.** (d)	**29.** (d)	**30.** (d)
31. (c)	**32.** (d)			

■■■■