IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 12, Issue 4 Ver. I (Jul. - Aug.2016), PP 92-98
www.iosrjournals.org

The Magic of Negative Numbers in Computers

U. Sridevi[1], L. Vishnu Priya[1], and Dr. D. Chinni Krishna[2*]
[1]*Dept. of Mathematics, Govt. City College, Nayapul, Hyderabad-500 001, Telangana State, India.*
[2]*Dept. of Physics, Bhavan's New Science College (PG), Narayanaguda,*
Hyderabad-500 029, Telangana State, India

Abstract: *The negative numbers played an important role in daily life as well as in performing various arithmetic operations in computers. In a given data, most significant bit (MSB) is used for indicating the negative number and if it is equal to 1, the data is a negative number and if it is equal to 0, the given data is a positive number. Detailed discussions on negative numbers were made by considering an 8 bit data. The magic circle shows all positive and all negative numbers by moving through anti clock wise or clock wise direction respectively through outer circle. In computers, all negative numbers are appeared in 2's compliment form and this format played a crucial role in all arithmetic operations with negative numbers. Adding two negative numbers carry condition is 1, which indicates the result is negative, while in adding one negative number with one positive number, the carry will be discarded or sometimes the generation of carry indicates no need of borrow and the result is positive. In computers, multiplication can be done by repeated addition process and repeated subtraction process is used in division operation and for all subtraction operations can be done by addition of 2's compliment values of negative numbers. Therefore a systematic analysis was carried out to explain the role of negative numbers in all arithmetic (Addition, Subtraction, Multiplication and Division) operations.*

Keywords: *Hexadecimal Numbers, Negative numbers, Data byte, Word, Most significant bit, 2's Compliment form, repeated addition and repeated Subtraction.*

I. Introduction

Negative numbers played a vital role in human life. They are used to represent borrows taken from the banks, neighbours, shops or finance companies etc.; In real life negative numbers used are -1, -2, -3, -4, ...-∞. The lower digit negative number has higher value than the higher digit negative number. For example, -3 have greater value than -5. In general, the order of the numbers in Mathematics is given below:

$$-\infty < -\infty + 1 < -\infty + 2 < -\infty + 3 .... < -3 < -2 < -1 < 0 < 1 < 2 < 3 < 4 .... < \infty-3 < \infty-2 < \infty-1 < \infty \qquad (1)$$

As far as computers are concerned, they does not know the negative (-) sign and negative numbers, as it is purely functioning on the basis of electronic signals represented in terms of 0 and 1, where 0 indicates ground connection (0Volts) and 1 indicates 5Volts power supply. Sometimes, the 5Volts power supply is also called as $+V_{CC}$. In electronics, the term $-V_{CC}$ is used to represent negative voltages, which has 180^o phase difference with $+V_{CC}$. If $-V_{CC}$ is used to represent negative values in computers, there is a chance of overlapping of data and damaging the actual information to be transferred from the source and also at the receiving end there will be a chance of getting errors.

With the above views, the computer scientists come to a conclusion that it is always convenient to keep one of the data bit, usually most significant bit (MSB) is used to indicate the negative sign. All modern computers are designed to use binary digits to represent numbers and other information as given in a text book in [1]. In computers any information (Audio, Video, Text etc.), can be converted in to its equivalent electronic digits 0s and 1s and split the digits in byte (group of 8 bits) format, which is the smallest data handled by the computers. There is no limit to the number of digits that may be needed to write down integer numbers. On the other hand, the resources in terms of storage capacity and available computing time are always finite. So, there will always be an upper limit on the magnitude of numbers that can be handled by a given computer. For efficiency, the electronic circuits have been designed to handle groups of binary digits. The smallest such group consists of 8 binary digits and is called a byte. Larger groups of bits are usually bytes. For manipulation of numbers, groups of 4 and 8 bytes are usually used.

The byte can be represented in 8 bit binary or 2 digit hexadecimal codes. Each hexadecimal digit consists of 4 bits Binary data called as a nibble. In computer, the Kilo stands for 2^{10} = 1024, Mega is 2^{20} = 1048576, and Giga is 2^{30} = 1073741824. Normally, the exponent is used to indicate the address bits. The first computer has just 4 address bits. It means it has 2^4 =16 address locations and in each location one byte of data will be stored. Therefore, a computer has n address bits can able to store 2^n bytes of data, which is the minimum size of memory of the computer. By adding an address bit will doubles the memory storage capacity.

The 8085 microprocessor and 8051 microcontrollers both has 16 bit address capacity, therefore, they can able to store 2^{16} bytes = 2^6 x 2^{10} bytes = 64 Kilo Bytes of data. In the present market, 32 bit (4GB) and 33 bit (8GB) capacity memory devices are more familiar. The memory is usually organized in to strings of bits called words. Each such string has the same length in a particular computer, although different computers may use different word lengths. For example, IBM PC and AT Systems use a word length of 16 bits, while VAX 11 systems use a word length of 32 bits.

The largest decimal number that can be stored in computers is given by 2^{n-1}, where n is word length in bits. For an 8 bit data, the largest number is 2^8 -1 = 255_{10} = FFH. If sign is considered the numbers are from -2^7 to $2^7 - 1$, i.e. -128 to + 127. Similarly for 16 bit data, the largest number is $2^{16} - 1$= 65535_{10} = FFFFH and the signed numbers are from -2^{15} to 2^{15}-1, i.e. -32768 to + 32767. More details on magnitudes of numbers were given in a thesis given in [2]

If the given data (x say) is a real number, the floating point form representation is x = f x 10^E where f is mantissa and E is an exponent. Floating point numbers are stored differently. The entire memory location is divided in to 3 fields. The first field is reserved for sign, second field is for the exponent of the number and the third field is for mantissa of the number. In the present work is concentrated on negative numbers used in computers only.

The computer data byte can be represented in below format

D7	D6	D5	D4	D3	D2	D1	D0
MSB							LSB

The bit D7 (MSB) is used to represents the sign. If D7 = 0, the given data is a positive number, while D7 = 1, the data is a negative number. In an 8 bit data, the values 00 to 7FH (0 to +127) are positive values and FFH to 80H (-1 to -128) are the negative values. For unsigned numbers all bits are considered as positive and the values are from all bits low to all bits high i.e. 00 to FFH (0 to 255_{10}).

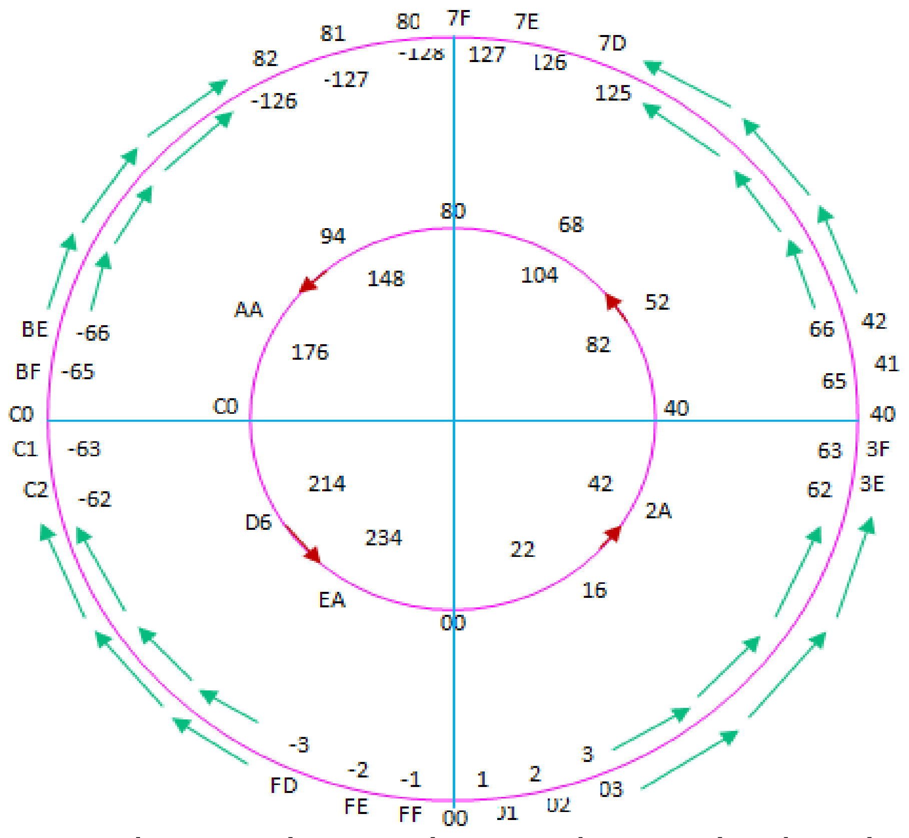

Fig. 1. The Magic Circle: Inner Circle represents the Un Signed Numbers and outer circle represents the Signed Numbers in Hex and Decimal form

II. The Magic Circle

The magic circle consists of inner and outer circles. The inner circle represents the unsigned n bit data and outer circle represents the signed numbers. The magic circle for an 8 bit data was shown in Fig. 1. It was observed from figure 1 that, while moving through inner circle in an anticlockwise direction from the origin 00, it gives all positive values from 00 to FFH as shown through outer part of the inner circle or its decimal equivalent values from 0 to 255_{10} as shown through inner part of the inner circle.

It was also observed from figure 1 that, while moving in an anticlockwise direction through outer circle from the origin 00 to 7FH are the positive values, but from 80H to FFH are negative values. It is always comfort to represents the signed numbers by moving from the origin 00 to half cycle clockwise direction for negative numbers and half cycle anti clock wise direction for positive values. The negative values -1, -2, -3, -4, -5, ..., -127, -128 are obtained by their 2’s compliment hexadecimal values. Similar type of circle for 3 bit signed numbers was given in an internet article in [3]. The 8 bit data in different forms were given in Table 1.

Table 1. Representation of 8 bit numbers in Binary, Hexadecimal, Positive and Negative Decimal forms

Binary Form								**Hex Form**		**Positive Decimal**			**Negative Decimal**		
2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**	**16^1**	**16^0**	**10^2**	**10^1**	**10^0**	**10^2**	**10^1**	**10^0**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
0	0	0	0	1	0	0	1	0	9	0	0	9	0	0	9
0	0	0	0	1	0	1	0	0	A	0	1	0	0	1	0
0	0	0	0	1	0	1	1	0	B	0	1	1	0	1	1
0	0	0	0	1	1	0	0	0	C	0	1	2	0	1	2
0	0	0	0	1	1	0	1	0	D	0	1	3	0	1	3
0	0	0	0	1	1	1	0	0	E	0	1	4	0	1	4
0	0	0	0	1	1	1	1	0	F	0	1	5	0	1	5
0	0	0	1	0	0	0	0	1	0	0	1	6	0	1	6
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
0	1	1	0	0	0	1	1	6	3	0	9	9	0	9	9
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
0	1	1	1	1	1	0	1	7	D	1	2	5	1	2	5
0	1	1	1	1	1	1	0	7	E	1	2	6	1	2	6
0	1	1	1	1	1	1	1	7	F	1	2	7	1	2	7
For below data, MSB bit is 1. If sign is considered all numbers from here are negative numbers															
1	0	0	0	0	0	0	0	8	0	1	2	8	-1	2	8
1	0	0	0	0	0	0	1	8	1	1	2	9	-1	2	7
1	0	0	0	0	0	1	1	8	2	1	3	0	-1	2	6
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
1	1	0	0	0	0	0	0	C	0	1	9	2	-0	6	4
1	1	0	0	0	0	0	1	C	1	1	9	3	-0	6	3
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
1	1	1	1	0	0	0	0	F	0	2	4	0	-0	1	0
1	1	1	1	0	0	0	1	F	1	2	4	1	-0	0	F
1	1	1	1	0	0	1	0	F	2	2	4	2	-0	0	E
“	“	“	“	“	“	“	“	“	“	“	“	“	“	“	“
1	1	1	1	1	1	0	0	F	C	2	5	2	-0	0	4
1	1	1	1	1	1	0	1	F	D	2	5	3	-0	0	3
1	1	1	1	1	1	1	0	F	E	2	5	4	-0	0	2
1	1	1	1	1	1	1	1	F	F	2	5	5	-0	0	1

The Table 1 gives all possible positive and negative values for an 8 bit data in binary, hexadecimal and decimal forms. As for the above discussions, the negative numbers would have ‘1’ as the left most significant bit. So, -1 would be the value 1000 0001. This causes a slight problem as discussed in the thesis in [2], as explained below:
since -1 + 1 in the computer would become

-1 + 1 = 0 = 1000 0001 + 0000 0001

0 = 2’s compliment of 1000 0001 + 0000 0001

= 0111 1111 + 0000 0001 = 1000 0000

Therefore, if -1 is equated with 1000 0001, then 1000 0000 should represent ‘0’. But, the binary pattern 0000 0000 is used to represent the value ‘0’. So, both 1000 0000 and 0000 0000 would represent zero. It violates the principle of Mathematics.

Suppose, -1 is represented with its 2’s compliment form 1111 1111, then

-1 + 1 = 0 = 1111 1111 + 0000 0001 = 0000 0000 by discarding carry condition and which gives the correct value for ‘0’. Therefore, always represents the negative numbers in 2’s compliment form only.

III. Conversion From Hexadecimal To Its Negative Equivalent And Vice Versa

The 2's compliment value of hexadecimal data was its negative hexadecimal equivalent. Converting from hexadecimal to decimal will give negative equivalent decimal value. The 2's compliment of 'a' can be obtained by determining the 1's compliment of 'a', and adding 1 to it. i.e. 2's compliment of 'a' = $\bar{a}$ + 1.

Ex: 1. The 2's compliment of FFH is determined by the following way:
The binary equivalent of FFH = 1111 1111B (Where B stands for Binary)
The 1's compliment of FFH = 0000 0000B = 00H
The 2's compliment of FFH = 1's compliment of FFH + 01H = 0000 0000B + 0000 0001B
= 0000 0001B = 01H
Therefore the negative equivalent of FFH = $(-01)_{10}$

Ex: 2. The 2's compliment of 80H is determined by the following way:
The binary equivalent of 80H = 1000 0000B (Where B stands for Binary)
The 1's compliment of 80H = 0111 1111B = 7FH
The 2's compliment of FFH = 1's compliment of FFH + 01H = 7FH + 01H
= 80H = $(128)_{10}$
Therefore the negative equivalent of 80H = $(-128)_{10}$
Note: The only signed hexadecimal data, which is equal to its 2's compliment hexadecimal value. The 2's compliment of 00 is also 00 by discarding carry condition, but it is treated as a positive number.

The negative decimal value equivalent of hexadecimal form can be obtained by converting its magnitude in hexadecimal form and finding its 2's compliment value.

Ex: 1. the negative value $(-20)_{10}$ equivalent of hexadecimal data can be obtained by the following way
$(20)_{10}$ equivalent hexadecimal value = 14H = 0001 0100B
1's compliment $(20)_{10}$ = 1110 1011B = EBH
2's compliment $(20)_{10}$ = EBH + 01H = ECH
Therefore, negative value $(-20)_{10}$ equivalent hexadecimal is ECH

Ex: 2. the negative value $(-105)_{10}$ equivalent of hexadecimal data can be obtained by the following way
$(105)_{10}$ equivalent hexadecimal value = 69H = 0110 1001B
1's compliment $(20)_{10}$ = 1001 0110B = 96H
2's compliment $(20)_{10}$ = 96H + 01H = 97H
Therefore, negative value $(-105)_{10}$ equivalent hexadecimal is 97H

IV. Arithmetic Operations With Negative Numbers

The widely used arithmetic operations are addition, subtraction, multiplication and division. All these arithmetic operations with suitable examples for negative numbers were given below

IV.1. Addition of Two Negative Numbers

Adding Two negative numbers is the same as decimal addition as mentioned in a Lecture [4]. If the number is already in 2's compliment form add them directly. For negative decimal numbers, first converting them in to its hexadecimal equivalent by finding 2's compliment value and then add the 2's complimented values.
Ex. 1. (-01) + (-01) = 2's compliment equivalent of 01 + 2's compliment equivalent of 01
= 1111 1111 + 1111 1111 = FFH + FFH
= 1111 1110 = FEH = (-02)
Here, carry is simply discarded as per the details given in a book in [5] or the generation of carry indicates the sum is a negative number.

Ex. 1. (-63) + (-64) = 2's compliment equivalent of 63_{10} + 2's compliment equivalent of 64_{10}
= 2's compliment equivalent of 3FH + 2's compliment equivalent of 40H
= 1100 0001 + 1100 0000 = C1H + C0H
= 1000 0001 = 81H = $(-127)_{10}$
Here, carry is discarded and the sum is a negative number.

Ex. 3. (-64) + (-65) = (-129) is the expected answer
2's compliment equivalent of 64_{10} + 2's compliment equivalent of 65_{10}
= 2's compliment equivalent of $40H_0$ + 2's compliment equivalent of 41H

= 1100 0000 + 1011 1111 = C0H + BFH = 7FH = $+127_{10}$?

Here, the obtained answer is not matching with the expected answer, because the sum is outside the range of 8 bits data limit. The highest negative number of an 8 bit data is 80H = $(-128)_{10}$

IV.2. Subtraction of Two Negative Numbers

Operating with two positive numbers also, there is a chance of producing negative numbers. For example, both a and b are positive numbers, in subtraction a-b = negative if b >a. Let the negative numbers A and B and their subtraction is performed as

-A – (-B) = -A + B

Therefore, the subtraction is nothing but addition of one positive number and one negative number. To subtract binary number b, from a, simply take the 2's compliment of b, and add to a. That is:

a - b = a + (2's Compliment of b) = a + ($\bar{b}$ + 1) = a + $\bar{b}$ + 1

Ex. 1. (-01) + 03 = 2's compliment of 01 + 03
= FFH + 03 = 1111 1111 + 0000 0011
= 0000 0010 = 02

Here, carry is discarded or the generation of carry indicates no need of borrow, therefore, the carry condition in the flag register is '0' and the answer is a positive value. This concept is completely opposite to the previous cases of addition of two negative numbers, where generation of carry indicates the results is a negative value and the carry condition in the flag register is 1. This property is experimentally proved for an 8 bit data in a lab manual in [6].

Ex. 2. 01 + (-03) = 01 + 2's compliment of 03
= 01 + FDH = 0000 0001 + 1111 1101
= 1111 1110 = FEH = (-02)

Here, no carry generated means the answer is a negative number. It needs borrow and the carry condition in the flag register is 1. This process can be easily understood by normal subtraction process as given in Table 2.

Table 2. Analysis of subtraction process to understand the concept of generation of carry in subtraction

Ex. 1. (-01) + 03 = 03 + (-01)		Ex. 2. 01 + (-03)	
Step 1:		Step 1:	
First number	03	First number	01
Second number	01	Second number	03
		Step 2:	
Step 2:		Second number is bigger one and take borrow by setting Cy =1 and add 16_{10} to MSB bit of first number and forward 16_{10} to LSB bit place	$Fx16_{10} + 16_{10}$
First number is bigger one than second one and no need of borrow to subtract. Therefore Cy = 0 and no need of any changes in the given numbers		First number was	01
		2's compliment of Second number	03
First number was	03	Step 3:	
Second number	01	First number was	$Fx16_{10} + 17_{10}$
		Second number	$0x16_{10} + 03$
Difference is	02	Difference was	$Fx16_{10} + 0E$
No need of borrow in subtraction indicates Cy = 0, it indicates the answer is positive.			= FEH = (-02)
		Borrow taken, therefore Cy =1 and answer is negative.	

IV.3. Multiplication Of Two Negative Numbers

In computers, multiplication can be done by repeated addition process. In which one multiplicand can be added by another multiplicand times. It is always comfort to add the higher multiplicand by the lower multiplicand times.

Ex. 1. $(-128)_{10}$ x $(-128)_{10}$ = $(16384)_{10}$ was the expected answer
(2's compliment of 128_{10}) x (2's compliment of 128_{10})
= (2's compliment of 80) x (2's compliment of 80)
= (80H) x (80H)
i.e. addition 80H for 128_{10} times
(80H + 80H) + (80H + 80H) + ...+ 64 times
= (100H) + (100H) +..................64 times
= (200H) + (200H) + ...32 times
= 400H + 400H + ...16 times
= 800H + 800H + ...08 times
= 1000H + 1000H + ...04 times
= 4000H = 4 x 16^3 = 4 x 4096 = 16384_{10}

The result is matched with the expected answer in magnitude.

Ex. 1. $(05)_{10}$ x $(-05)_{10}$ = $(-25)_{10}$ was the expected answer
$(05)_{10}$ x (2's compliment of 05_{10})
= (05) x (FBH)
(i.e. addition FBH for 05 times)
= (FBH + FBH) + FBH + FBH + FBH
= (F6H + FBH) + FBH + FBH
= (F1H + FBH) + FBH
= ECH + FBH
= E7H
= -25_{10}

The result is matched with the expected answer and carry was discarded in each addition.

IV.4. Division Of Two Negative Numbers

In computers, division can be done by repeated subtraction process. In which denominator is always compared with the numerator and subtract denominator from numerator when numerator is greater or equal to denominator. If denominator is larger stop the subtraction process. If A is at numerator place and B is at denominator place, then carry is set to 1 for A < B only in comparing of B with A. Therefore subtraction can be done until carry is set to '1' and it is essential to compare the remainder with the denominator after each subtraction.

Ex. 1. (-128_{10}) / 20_{10} = 06 with remainder -08 was the expected answer
i.e. 80H / 14H = 06 with remainder 08 was the expected answer

The process of division was explained in Table 3.

Table 3. The process of repeated subtraction by using decimal and Hexadecimal numbers

Division (repeated subtraction) in decimal form in a manual process			**Division (repeated subtraction) in hexadecimal form performed by the computer**		
	20) -128 (Quotient = 00			14H) 80H (Quotient = 00	
-128 > 20, so subtract	20	= 01	80H >14H, so subtract 14H	ECH	= 01
	-108		(Add 2's compliment of 14H)	6CH	
-108 > 20, so subtract	20	= 02	6CH > 14H, so subtract 14H	ECH	= 02
	-88			58H	
-88 > 20, so subtract	20	= 03	58H > 14H, so subtract 14H	ECH	= 03
	-68			44H	
-68 > 20, so subtract	20	= 03	44H > 14H, so subtract 14H	ECH	= 04
	-48			30H	
-48 > 20, so subtract	20	= 05	30H > 14H, so subtract 14H	ECH	= 05
	-28			1CH	
-28 > 20, so subtract	20	= 06	1CH > 14H, so subtract 14H	ECH	= 06
	-08			08H	
-08 < 20, Cy = 1, Stop the subtraction process			08H < 14H, Cy = 1, Stop the subtraction process		

It is observed from Table 3 that the magnitudes in results obtained in manual division and the division performed by the computer were perfectly matched and the quotient value is incrementing by 1 for each subtraction. The electronic circuit diagrams for addition, subtraction, multiplication and division were given in an article in [7, 8].

V. Conclusions

1. The Magic Circle gives the complete information about un signed numbers through inner circle and signed numbers through outer circle in computers.
2. The most significant bit (MSB) is used for indicating the sign in computers. If MSB = 0, the given data is a positive number and MSB = 1, the data is a negative number.
3. The negative integer negative numbers of an 8 bit data -1, -2, -3, -4, ...-127, -128_{10} can be represented in its 2's compliment form as FFH, FEH, FDH, FCH,...........81H, 80H.
4. For an 8 bit data, total 256 data's can be splitting into 128 positive data's from 00 to 7FH (0 to 127_{10}) and 128 negative data's from FFH to 80H (-1 to -128_{10}).
5. In an arithmetic operation, there is a chance of obtaining wrong results when answer exceeding the size of the data bits. Enhance the bits to get accurate results.

6. The generation of carry in subtraction process indicates no need of borrow (i.e. the CY flag bit is '0') and the result is positive. If carry not generated means borrow required (i.e. the CY flag bit is '1') and result is negative and it is appeared in 2's compliment form of a negative number.
7. In computers, multiplication can be done by repeated addition process and division can be performed by repeated subtraction operation.

Acknowledgements

The First and Second authors would like to express their thanks to the Principal, Staff and students of Govt. City College for their support. They also would like to expresses their thanks to their family members for their encouragement, kind cooperation and support.

The third and corresponding author would like to express his thanks to the Management of Bhavan's New Science College (PG) for their encouragement and support. He also wishes to thanks his Family members (Especially his kid Master Krishna Chaithanya, who makes him happy) colleagues and M.Sc. Applied Electronics students for their support.

References

[1] Numerical Methods by E. Balaguru Swamy, Tata Mc grawhill Publishing Company Limited, New Delhi.
[2] Chapter 4, Computers, Numbers and Text: a thesis appeared on Internet Google search engine.
[3] Representing Signed integer numbers inside a computer, an article available in Google search engine.
[4] Signed Binary numbers and Binary codes by N.B. Dodge on 09/2015 at Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas (UTD).
[5] Digital Principles and Applications, 7th edition 2011, Donald P Leach, Albert Paul Malvino and Goutam Saha, Tata Mc grawhill Publishing Company Limited, New Delhi.
[6] Dr. D. Chinni Krishna and Prof. G. Pushpa Chakrapani M. Sc. Physics, Laboratory Manual & Record on Memory Devices & Microprocessors of Dr.B.R. Ambedkar Open University, Hyderabad.
[7] Thesis Chapter 9, Numbers in Computers, Brand Fortner, Theodore E. Meyer in Number by colours, Springer Verlog, New York INC, 1997.
[8] Number Representation and Computer Arithmetic (B. Parthami / UCSB), Article to appear in Encyclopedia of Information Systems, Academic Press, 2001. Pp: 1-49.