Proceedings of the 25th SMART-iSTEAMS
Trans-Atlantic Multidisciplinary Conference
in Collaboration with
The Laboratoire Jean Kuntzmann, Universite Grenoble, Grenoble, France
Society for Multidisciplinary & Advanced Research Techniques (SMART)
The Institute of Electrical & Electronics Engineers Computer Society, Nigeria
www.isteams.net/france2020

Implementing Integer Number Representations Using Advanced Radix and Diminished-Radix Methods.

Ashioba, N. C
Department of Computer Science
Delta State Polytechnic
Ogwashi-Uku, Nigeria.
E-mail: ashinze2008@yahoo.com

ABSTRACT

In digital system, complement operations have been used to simplify the subtraction (unary minus) and arithmetic not operations on positive integer numbers. These operations have been solved using the signed-magnitude representation, signed 1's complement representation and the 2's complement representation. These methods did not consider the sign-bit of the given positive integer numbers. This paper proposes the advanced radix and diminished-radix methods to perform the two's complement and one's complement operations respectively. The proposed methods provide mathematical models for evaluating the complement operations.

Keywords: Number System, Complement, Radix method and Diminished-Radix Method

25th iSTEAMS Trans-Atlantic Multidisciplinary Conference Proceedings ReferenceFormat
Ashioba, N.C. (2020): Implementing Integer Number Representations Using Advanced Radix and Diminished-Radix Methods.. Proceedings of the 25th iSTEAMS Trans-Atlantic Multidisciplinary Virtual Conference, Laboratoire Jean Kuntzmann, Universite Laboratoire Jean Kuntzmann, Universite Grenoble, Alpes, France June, 2020. Pp 221-228. www.isteams.net/France2020.

1. INTRODUCTION

In a positional number system, each number is represented by a string of digits in which each digit position i has an associated weight r_i, where r is the radix or base of the number system. The general form of a number in a number system with radix-r is [1] –[4]:

$$x = (\dots\dots..x_{k-1}x_{k-2}x_{k-3}\dots x_1x_0\dots\dots.)_r$$

$$= \sum_{i=-\infty}^{\infty} x_i r^i$$

where the value of any digit x_i is an integer in the range $0 \leq x_i \leq r$. For known digits, the general form of the number is:

$$x = (x_{k-1}x_{k-2}x_{k-3}\dots x_1x_0)_r$$

$$= \sum_{i=0}^{k-1} x_i r^i$$

For a number with sign-magnitude digit, the general form of number in a system with radix-r is represented in Equation (i):

$$x = (x_k x_{k-1} x_{k-2} x_{k-3} \dots x_1 x_0)_r$$

$$= \sum_{i=0}^{k} x_i r^i \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (i)$$

Where x_k represents the sign bit which can be positive or negative.

A floating-point (FP) number can be represented in the form ±M x b^e, where m, called the mantissa, represents the fractional part of the number and is normally represented as a single binary fraction; e represents the exponent and b represents the base radix of the exponents [1]-[4]. The floating-point numbers are represented as:

$$x = (\pm 1) \times m \times 2^{\pm \text{exponent}}$$

$$x = \text{sign} \times \text{Mantissa} \times 2^{\pm \text{exponent}}$$

where $1.0 \le m < 2$ and $m = (b_0 . b_1 b_2 b_3 \dots\dots)_2$

To store the number in floating-point representation, a computer word is divided into three fields: sign bit, exponent and mantissa respectively. In a 32-bits single precision floating point representation, the most significant bit is used for the sign (s) bit with 0 for positive number and 1 for negative number. the next 8-bits stands for the exponent and the last 23-bits is used for the mantissa. The 32-bits single precision floating-point representation is shown in Figure 1.

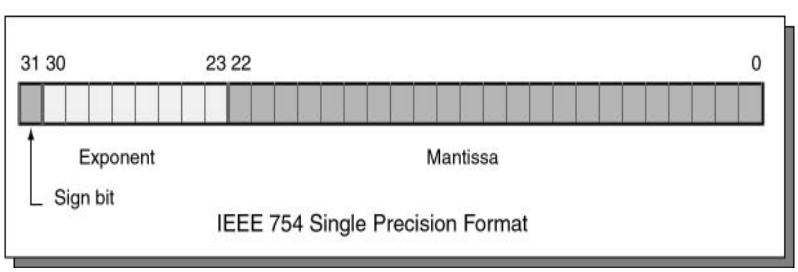

Figure 1: 32-bits single-precision floating-point representation (Source: [1]-[4])

In a 64-bits double precision floating point representation, the most significant bit is used for the sign (s) bit with 0 for positive number and 1 for negative number. the next 11-bits is used for the exponent and the last 52-bits for the mantissa. The 64-bits single precision floating-point representation is shown in Figure 2.

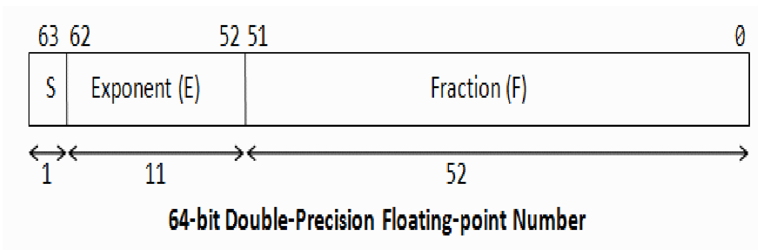

Figure 2: 64-bits double precision floating-point representation (Source: [2]-[3]).

The representation of the fields is shown in Table 1.

Table 1: Fields size representation of the floating-point numbers (Source: [1]).

Floating point type	Sign field size	Exponent field size	Mantissa field size	Bias
Single precision 32bits computers	1	8	23	127
Double precision 64bits computers	1	11	52	1023

Similarly, integer numbers can be represented as a word that is divided into two files: sign-field and the integer number field. An integer number representation of a word is shown in Figure 3.

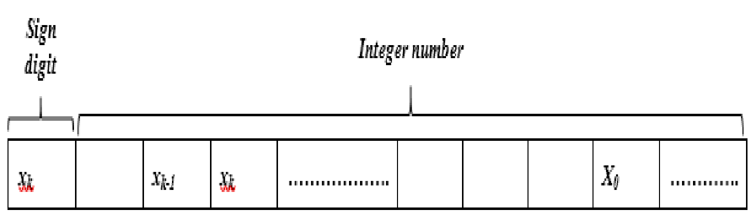

Figure 3: Integer number representation

Where s = x_k is the sign magnitude digit of the number, the sign digit takes two values: 0 and (r-1). The (r-1) value represents a negative sign whose value is always -1 in any radix -r.

A number system in base or radix-r is a system that uses r-distinct symbols or elements, ranging from 0 to (r-1). For example, the elements of a number system in base 10 are represented in a set. The 10 symbols or elements of the decimal number or radix-10 number system are:

$S_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Numbers are represented by strings of digit symbols. These number is derived by concatenating one or more digit-symbols of a particular base. Generally, any radix number is a sum of series of power of the base times a number from 0 to 1 less than the base. The syntax of number representation is shown in Equation (ii).

Number =............ $num[n] x base^{n} + num[n-1] x base^{n-1} + \ldots + num[0] x base^{0} + \ldots$ (ii)

For example, the string of digits 742.5_{10} is interpreted to represent a quantity expressed in decimal base as follows:

$$742.5 = 700 + 40 + 2 + \frac{5}{10} = 7x10^{2} + 4x10^{1} + 2x10^{0} + 5x10^{-1}$$

For example, the string of digits 101101 is interpreted to represent a quantity

$$101101 = 1x2^5+0x2^4+1x2^3+1x2^2+0x2^1+1x2^0$$
$$= 32 +0 +8 +4 +0 +1$$
$$= 45$$

Besides the decimal number system, the binary (radix-2) the octal (radix 8) and hexadecimal (radix 16) are important in digital computer system.

2. REVIEW OF RELATED WORKS

Complements are used in digital computers for simplifying subtraction (unary minus) operations and arithmetic not manipulations. The three methods used to implement the complement operations for a binary number system are:

1. The Sign-Magnitude operation
2. The Radix-1's complement operation
3. The Radix-1's complement operation

For instance, when dealing with binary number system where the value of radix is 2, the complements are 2's complement and 1's complement. Similarly, for decimal system where the value of radix is 10, the complements are 10's complement and 9's complement [mano].

2.1 Sign-Magnitude Method

According to this representation, the most significant bit (of the n-bits used to represent the number) is used to represent the sign of the number, such as a 1 in the most significant bit position indicates a negative number while a 0 in the most significant bit position indicates a positive number [1]. The remaining (n-1) bits are used to represent the magnitude of the number. for example, the negative number (-19) is represented binary number system as 110011.

2.2 The Radix Complement Method

[3]-[4] developed a mathematical model for the radix complement method. The radix complements also called the r's complement of a number N_r with a base of r is defined as another number $(Nr)_r$ obtained by subtracting each digit of N_r from r-1 and adding 1 [5].

That is:

$$\underline{\mathbf{N}}_{\mathbf{r}} = (\mathbf{Nr})_{\mathbf{r-1}} + \mathbf{1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\mathbf{iv})$$

Let Nr = ($n_1n_2n_3n_4n_5n_6\ldots n_i$)r where i= total number of digits that make up Nr. By definition, we have:

($\underline{\mathbf{Nr}}$)r-1 = (r-1-n1)(r-1-n2)….(r-1-ni) +1

For example, the 2's complement of +19 is calculated as follows:

$$\mathbf{19 = 10011 = (1-1)(1-0)(1-0)(1-1)(1-1)}\text{+1}$$
$$\mathbf{= 01100 + 1 = 01101}$$

3. Diminished-Radix Complement Method

[3]-[4] also developed a mathematical model for the diminished-radix complement method. The diminished-radix complements also called the (r-1)'s complement of a number Nr with a base of r is defined as another number ($\underline{\mathbf{Nr}}$)r-1 obtained by subtracting each digit of Nr from r-1 [5]. That is:

$$\underline{\mathbf{Nr}} = (\mathbf{Nr})_{\mathbf{r-1}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\mathbf{iv})$$

Let Nr = ($n_1n_2n_3n_4n_5n_6\ldots n_i$)r where i= total number of digits that make up Nr. By definition, we have:

($\underline{\mathbf{Nr}}$)r-1 = (r-1-n1)(r-1-n2)….(r-1-ni)

The r's and (r-1)'s complements are related by the equation:

$$\mathbf{r's\ complement = (r-1)'s\ complement + 1}$$

For example, the 1's complement of +19 is calculated as follows:

$$\mathbf{+19 = 10011 = (1-1)(1-0)(1-0)(1-1)(1-1)}$$
$$\mathbf{= 01100 + 1 = 0110}$$

Therefore, the 2's complement will be:

1's complement + 1 = 0110 + 1 = 01101

These methods do not consider the sign-bit of integer numbers. The methods add 1 to the most significant bit position to generate:

101101 = -19

3. THE PROPOSED METHODS

To introduce the sign-bit, the researcher has developed the advanced radix complement and diminished-radix complement methods.

3.1 Advanced Radix Complement Method

The advanced radix complement method is defined in Equation (v).

$$\mathbf{r'scomplement} = \{\mathbf{r^{n+1} - N \ for \ N \neq 0 \ 0 \ for \ N = 0} \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (\mathbf{v})$$

Where r represents the base or radix of the integer number of digit n and (n+1) digit represents the sign-magnitude of the number. The sign magnitude can take positive or negative values. A negative sign-magnitude of -1 in the (n+1) digit takes value (r-1) while a positive sign magnitude takes value 0 in the most significant bit position.

For example, the 2's complement of +19 is calculated as follows:

$$\mathbf{2's\ \ complement} = \mathbf{2^{5+1}} - \mathbf{10011} = (\mathbf{1000000} - \mathbf{10011}$$
$$= 111111 - 10011 = 101100$$

This number is translated as:
$$-1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$
$$= -32 + 0 + 8 + 4 + 0 + 1 = -32 + 13 = -19$$

This number is translated as:
$$-1 \times 10^5 + 7 \times 10^4 + 6 \times 10^3 + 5 \times 10^2 + 5 \times 10^1 + 0 \times 10^0$$
$$= -100000 + 70000 + 6000 + 500 + 50 + 0 = -23450$$

3.2 Advanced Diminished-Radix Complement Method

The advanced diminished-radix complement method is defined as:

If a positive number N is given in base r with an integer part of n-digits and a fractional part of m-digits, then the (r-1)'s complement of N is shown in Equation (vi).

$$(\mathrm{r}-1)'\mathrm{s\ complement} = \{(\mathrm{r}^{\mathrm{n+1}} - 1) - \mathrm{N\ for\ N} \neq 0\ 0\ \mathrm{for\ N} = 0\ \ \ldots\ldots\ldots\ldots. (\mathrm{vi})$$

The r's and (r-1)'s complements are related by the equation:
$$\mathrm{r's\ complement} = (\mathrm{r}-1)'\mathrm{s\ complement} + 1 \qquad (\mathrm{vii})$$

For example, the 1's complement of +19 is calculated as follows:

$$1'\mathrm{s\ complement} = (2^{5+1} - 1) - 10011 = (1000000 - 1) - 10011$$
$$= 111111 - 10011 = 101100$$

This number is translated as:
$-1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
$= -32 + 0 + 8 + 4 + 0 + 0 = -32 + 12 = -20$

Therefore, the r's complement is calculated as:

r's complement = -20 +1 = -19

4. DISCUSSION OF RESULTS

From the results obtained, it can be inferred that the advanced methods developed or modelled by the researcher in this paper has been able to handle the sign-bit and sign-magnitude of the positive integer numbers used in the number systems.

5. CONCLUSION

The researcher has modelled or developed mathematical models for simplifying the subtraction (unary minus) and Arithmetic not operations in a digital computer system.

REFERENCE

[1] Mostafa, A, B, and Hesham, E. Fundamentals of Computer Organization and Architecture. Wiley Interscience, Inc Publication Hoboken, New Jersey New York, 2005.

[2] Elahi A. (2018). Computer System Digital Design, Fundamentals of Computer Architecture and Assembly language. Springer International Publishing, USA.

[3] Mano, M. M. (1993). Computer System Architecture. 3rd ed. Prentice Hall.

[4] Mano, M. M. and Ciletti. M. D. (2013). Digital Design with an Introduction to Verlog HDL 5th ed. Prentice Hall, New Jersey, USA.

[5] Omotosho, O. J., Ogunlere, S. O. and Awodele, O. (2013) Negative Numbers & Complement Arithmetic in Computer Systems. International Journal of Advanced and Innovative Research (2278-7844) #138, Vol2, Issue 11.