Journal of Engineering and Applied Sciences 4 (2): 132-134, 2009
ISSN: 1816-949X
© Medwell Journals, 2009

Application of Decimal-To-Binary Conversion Technique to Finding the Exponent of Numbers

Benjamin A. Weyori
Department of Applied Mathematics and Computer Science,
University for Development Studies, Navrongo, Ghana

Abstract: Exponentiation is one of the important functions in computer arithmetic. In this study, we propose a new method for finding the exponents of numbers. The method is valid for both signed and unsigned real numbers with the exception of zero. In order to demonstrate the applicability of the proposed method, we wrote a simple C++ program.

Key words: Exponentiation, signed, unsigned, C++ program, computer arithmetic, decimal-to-binary conversion

INTRODUCTION

Number representation is arguably the most important topic in computer arithmetic. Hence, the choice of number representation affects the implementation cost and delay of all arithmetic operations (Parhami, 2000).

Numbers play an important role in computer systems. Numbers are the basis and object of computer operation. The main task of computer is computing, which deals with numbers all the times (Mi Lu, 2004).

The elementary method of finding the square of a number is to multiply the number by itself once, in the case of a cube the number is multiplied by itself twice. For the case that the exponent is greater than three the number is multiplied by itself the numbers of counts of the exponent (David, 1980). An example is shown as:

$$n^4 = n \times n \times n \times n$$
$$m^7 = m \times m \times m \times m \times m \times m \times m$$

Many researchers have worked on numbers and their squares, cubes and exponents. Since, the basis of mathematics comes from numbers and also exponent of numbers being the backbone of numbers is generated by a simple multiplication.

Andrews (1960) worked on magic squares and cubes of numbers, where a simple method is use to generate the square and cube of numbers. Apostol and Zuckerman (1951) worked on the magic squares constructed by the uniform step method.

MATERIALS AND METHODS

Arithmetic operations in digital systems are usually done in binary because design of logic circuits to perform binary arithmetic is much easier than for decimal. Binary arithmetic is carried out in much the same manner as decimal, except the addition and multiplication table are much simpler (Herman and O'Malley, 1988).

The main idea of this study into the development of a new method is derived from the conversion process of a binary number back into a decimal number for representation.

The binary number system is simply another way to count. It is less complicated than the decimal system because it is composed of only two digits (0 and 1). Addition and subtraction is simply carried out like in the case of a decimal number system, but it may seem more difficult at first because it is unfamiliar to us (Irv, 2003).

Say given a binary numbers 1111 to be converted to decimal for representation and use. The process below is looked at in details by Charles (2004) and Thomas (1986).

1111 to convert to binary

$$= (1\times2^3) + (1\times2^2) + (1\times2^1) + (1\times2^0)$$
$$= (1\times8) + (1\times4) + (1\times2) + (1\times1)$$
$$= 8+4+2+1$$
$$= 15$$

Considering the above example, since the digits in the binary number (say that is the number of ones in the number), are four so we assume that we take two to the power five and use the method above:

$$K = 2^3 + 2^2 + 2^1 + 2^0$$
$$k = 8 + 4 + 2+1 = 15$$
$$\text{but, } 2^4 = 16$$

From induction we say

$2^4 = k + 1$, but $k = 15$

$2^4 = (2^3 + 2^2 + 2^1 + 2^0) + 1 = 16$

and applying it to 3-5 and so on, will help us to generalize the formulae for signed and unsigned real numbers. Let find 3^3:

K in this situation will be

$$K = 3^2 + 3^1 + 3^0$$
$$K = 9 + 3 + 1 = 13$$
but, $3^3 = 27$

Hence, from mathematical induction, using the new method

$$3^3 = 2k + 1$$
$$3^3 = 2(3^2 + 3^1 + 3^0) + 1$$
$$= 2(13) + 1 = 27$$

Applying it to 4^2

$$4^2 = 3k + 1$$
$$K = 4^1 + 4^0 = 5$$
$$4^2 = 3(4^1 + 4^0) + 1$$
$$= 3(5) + 1 = 16$$

Since, k was multiplied by two when we use 3 and multiplied by 3 when we used 4. Then, we can say if T is the number then;

$$T^4 = (T - 1) \times k + 1$$
$$K = T^3 + T^2 + T^1 + T^0$$

Hence, K depends on the exponent of the number and the value multiplied by the k depends on the number, say T.

The above research, we can generalize the formulae to be; this formulae works for positive exponents $R^n = (R-1)(R^{n-1} + R^{n-2} + R^{n-3} + \ldots R + 1) + 1$ for positive exponents (n) and positive and negative numbers (R).

For a negative exponent (-n) and negative or positive numbers:

$$R^{-n} = \frac{1}{(R-1)(R^{n-1} + R^{n-2} + R^{n-3} + \ldots + R^{n-n}) + 1}$$

This study looks at finding the positive exponents as well as the negative exponents of real number both positive and negative numbers. Ahmed (2004) and Henrich (1991) have focused their study on the generation of square of numbers.

RESULTS AND DISCUSSION

The proposed architecture is divided into four stages. The first stage is when the real number and the exponent are all positive, the second stage when the real number is positive and the exponent is negative, the third stage when the real number is negative and the exponent is positive and finally both the real number and the exponent are negative. The implementation of the proposed method is shown as:

Positive base and positive exponent:

$$R^n = (R-1)(R^{n-1} + R^{n-2} + R^{n-3} + \cdots R + 1) + 1$$
$$3^5 = (3-1)(3^{5-1} + 3^{5-2} + 3^{5-3} + 3^{5-4} + 3^{5-5}) + 1$$
$$= 2(3^4 + 3^3 + 3^2 + 3^1 + 3^0) + 1$$
$$= 2(81 + 27 + 9 + 3 + 1) + 1$$
$$= 2(121) + 1$$
$$= 242 + 1 = 243$$

Negative base and positive exponent:

$$R^n = R-1(R^{n-1} + R^{n-2} + R^{n-3} + \cdots + R^{n-n}) + 1$$
$$-0.7^3 = -0.7 - 1(-0.7^{3-1} + (-0.7^{3-2}) + (-0.7^{3-3})) + 1$$
$$= -1.7(-0.7^2 + (-0.7^{1)} + (-0.7^0)) + 1$$
$$= -1.7(0.79) + 1$$
$$= -0.343$$

Positive base and negative exponent:

$$R^{-n} = \frac{1}{R-1(R^{n-1} + R^{n-2} + R^{n-3} + \ldots + R^{n-n}) + 1}$$
$$0.7^{-3} \frac{1}{-0.7 - 1((-0.7)^{3-1} + (-0.7)^{3-2} + (-0.7)^{3-3}) + 1}$$
$$= \frac{1}{-0.3(2.19) + 1}$$
$$= \frac{1}{0.343}$$

Negative base and negative exponent:

$$R^{-n} = \frac{1}{R-1(R^{n-1} + R^{n-2} + R^{n-3} + \ldots + R^{n-n}) + 1}$$
$$-6^{-n} = \frac{1}{(-6-1)(-6^{4-1} + (-6)^{4-2} + (-6)^{4-3} + (-6)^{4-4}) + 1}$$
$$= \frac{1}{-7(-185) + 1}$$
$$= -\frac{1}{1296}$$

Implementation of C++ program:

```
 # Include<iostream.h>
# include<math.h>
Main ( )
{
/* Read input Data*/
Cout<<"Enter the value of the exponent n"<<endl;
Cin>>n;
Cout<<"Enter the number R"<<endl;
Cin>>R;
M=0
K=1
If (n<0)
{
n=n*(-1)
Do {
n=n-1
Do {
r=n
K=K*R
r=r-1
While, (r>1)
 }
m=m+k
while (n>1)
}
R=R-1(m+1)+1
R=1/R
}
Else {
Do {
n=n-1
Do {
r=n
K=K*R
r=r-1
While, (r>1)
 }
m=m+k
while (n>1)
}
R=R-1(m+1)+1
}
Cout<<" the result of Rn "<<endl;
Cout<<R;
}
```

CONCLUSION

In this brief, a new technique of finding the exponents of real numbers is derived from the conversion of a binary number to a decimal number. This proposed technique works for both positive and negative real numbers bases and for positive and negative whole number exponents with the exception of zero. The method has been applied successfully in four different stages that prove the correctness and accuracy of the proposed method.

REFERENCES

Ahmed, M.M., 2004. How many squares are there, Mr. Franklin? Constructing and enumerating franklin squares. Am. Mathe. Monthly, 111: 394-410.

Andrews, W.S., 1960. Magic squares and cubes. Dover Publications, pp: 67-70. ISBN: 0-486-20658-0.

Apostol, T.M. and H.S. Zuckerman, 1951. On magic squares constructed by the uniform Step method, Proc. Am. Math. Soc., 2 (4): 557-565.

Charles, H.R. Jr., 2004. Fundamental of Logic Design. 5th Edn. Thomas Brooks/Cole Publishing, Belmont, California, pp: 6-22. ISBN: 0-534-37804-8.

David, M.B., 1980. Elementary number theory. Allyn and Bacon Inc., Bolton, pp: 3-11. ISBN: 0205069657.

Henrich, C.J., 1991. Magic squares and linear algebra. Am. Mathe. Monthly, 98: 481-488.

Herman, L. and J. O'Malley, 1988. Fundamentals of computer engineering: Logic design and microprocessors. John Wiley and Sons, New York, USA, pp: 5-12. ISBN: 0-471-60501-8.

Irv, E., 2003. The Architecture of Computer Hardware and System Software: An Information Technology Approach. 3rd Edn. Wiley, pp: 11-30. ISBN: 978-0-471-36897-7.

Mi Lu, A., 2004. Logic in Computer Systems, John Wiley and Sons, Inc., Hoboken, New Jersey, pp: 1-15. ISBN: 0-471-46945-9.

Parhami, B., 2000. Computer Architecture: Algorithms and Hardware Designs. Oxford: University Press, New York, USA, pp: 3-18. ISBN: 0-19-512583-5.

Thomas, L.F., 1986. Digital Fundamentals. 3rd Edn. Charles E. Merrill Publishing, Columbus, USA, pp: 28-54. ISBN: 0-675-20517-4.