UNIT 4: COMPUTER ARITHMETIC

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 Integer Representation

4.3.1 Signed Integer Representation

4.3.2 Fixed Point Representation

4.3.3 Floating Point Representation

4.4 Signed 0 Magnitude Form

4.5 The Concept of Complement

4.5.1 Diminished Radix Complement

4.6 One's Complement

4.6.1 One's Complement Arithmetic

4.7 Two's Complement Arithmetic

4.8 Let Us Sum Up

4.9 Further Reading

4.10 Answers to Check Your Progress

4.11 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about integer both signed and unsigned representation
- learn how to represent both fixed point and floating point
- describe signed and magnitude form of integers.
- learn the concept of complements
- learn how to find 1's complement of binary numbers
- calculate 1's complement arithmetic
- learn how to find 2's complement and its arithmetic.

4.2 INTRODUCTION

Computer arithmetic is a branch of computer engineering that deals with methods of representing integers and real values in digital systems and

efficient algorithms for manipulating such numbers by means of hardware circuits or software routines. On the hardware side, various types of adders, subtractors, multipliers, dividers, square-rooters, and circuit techniques for function evaluation are considered. Both abstract structures and technology-specific designs are dealt with. Software aspects of computer arithmetic include complexity, error characteristics, stability, and certifiability of computational algorithms.

Computer arithmetic is a subfield of digital computer organization that deals with the hardware realization of arithmetic functions which has a major thrust to design of hardware algorithms and circuits to enhance the speed of numeric operations. It encompasses the study of number representation, algorithms for operations on numbers, implementations of arithmetic units in hardware, and their use. The basic application of computer arithmetic is in the general purpose systems for fast primitive operations for processor data paths. On the other hand, it has also applications on special purpose systems which are given below:

i) Signal and image processing
ii) Real-time 3D graphics
iii) HDTV, image compression
iv) Network processors (data compression, encryption/decryption)

In this unit we will learn about basic numeric operations. Here we will see how two integer numbers are represented in different format and do arithmetic operations on both unsigned and signed integers that a digital computer system understand.

4.3 INTEGER REPRESENTATION

There are basically two types of integer representation in digital computer system. They are:

i) Unsigned integer representation.
ii) Signed integer representation.

We have already learn how to convert an unsigned integer from one base to another. For example, 25_{10} decimal equivalent to 11001_2 binary numbers. So, for unsigned integer, it is very simple and easy to convert in

digital computer system. But for signed integer its needs some additional calculations.

Signed numbers require additional issues to be addressed. When an integer variable is declared in a program, many programming languages automatically allocate a storage area that includes a sign as the first bit of the storage location. By convention, a “1” in the high-order bit indicates a negative number. The storage location can be as small as an 8-bits (byte) or as large as several words, depending on the programming language and the computer system. The remaining bits (after the sign bit) are used to represent the number itself. How this number is represented depends on the method used.

In a computer system that uses signed-magnitude representation and 8 bits to store integers, 7 bits can be used for the actual representation of the magnitude of the number. This means that the largest integer an 8-bit word can represent is $2^7 – 1$ or 127 (a zero in the high-order bit, followed by 7 ones). The smallest integer is 8 ones, or –127. Therefore, N bits can represent $–2^{(N–1)} – 1$ to $+2^{(N–1)} – 1$.

We know that for n-bit number, the range for natural number is from 0 to $2^n –1$. There are three different schemes or methods to represent signed (negative) and unsigned integer. These are

i) Signed and Magnitude form.
ii) 1’s complement form.
iii) 2’s complement form

4.3.1 Signed Integer Representation

We know that for n-bit number, the range for natural number is from 0 to $2^n–1$.

For n-bit, we have all together different combination, and we use these different combination to represent numbers, which ranges from 0 to $2^n–1$.

If we want to include the negative number, naturally, the range will decrease. Half of the combinations are used for positive number and other half is used for negative number.

For n-bit representations, the range is from:

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

For example, if we consider 8-bit number, then range for natural number is from 0 to 255.

But for signed integer the range is from –127 to +127.

4.3.2 Fixed-Point Representation

Let us take an example to show and make clearly how binary numbers is represented in fixed-point.

We know that binary representation of 41.6875 is 101001.1011.

To store this number, we have to store two part of information, the part before decimal point and the part after decimal point. This is known as *fixed-point representation* where the position of decimal point is fixed and number of bits before and after decimal point are also predefined.

If we use 16 bits before decimal point and 8 bits after decimal point, in signed magnitude form, the range is from $-2^{16} - 1$ to $+2^{16} - 1$ and precision is $2^{(-8)}$. One bit is required for sign information, so the total size of the number is 25 bits.

[1(sign) + 16 (before decimal point) + 8 (after decimal point)]

1 bit (Sign bit)	16 bits (before decimal point)	8 bits (after decimal point)

Figure 4.1: Fixed-Point Representation

4.3.3 Floating-Point Representation

In digital computers, floating-point numbers consist of three parts: (i) a Sign bit, (ii) an Exponent part (representing the exponent on a power of 2), and (iii) a fractional part called a Significant (which is a fancy word for a Mantissa). For example a floating point representation is given below which a 14 bits model consists of 1 bit as sign bit, 5 bits as Exponent part and 8 bits as Mantissa part.

1 bit (Sign bit)	5 bits (Exponent)	8 bits (Mantissa)

Figure 4.2: Floating-Point Representation

In this representation, numbers are represented by a Mantissa comprising the significant digits and an Exponent part of Radix R.

The format is: *Mantissa* * $R^{Exponent}$

Numbers are often normalized, such that the decimal point is placed to the right of the first non zero digit.

For example, the decimal number, 5236 is equivalent to $5.236*10^3$

To store this number in floating point representation, we store 5236 in Mantissa part and 3 in Exponent part.

4.4 SIGNED MAGNITUDE FORM

In signed magnitude form, one particular bit is used to indicate the sign of the number, whether it is a positive number or a negative number. Other bits are used to represent the magnitude of the number.

For an n-bit number, one bit is used to indicate the signed information and remaining (n–1) bits are used to represent the magnitude. Therefore, the range is from $–2^{n–1}–1$ to $+ 2^{n–1}–1$.

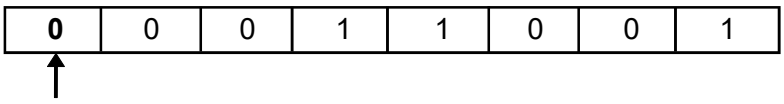
Generally, Most Significant Bit (MSB) is used to indicate the sign and it is termed as signed bit. 0 in signed bit indicates positive number and 1 in signed bit indicates negative number.

For example, 01011001 represents +169

but 11011001 represents –169

For example, if in a computer, the word size is 1 byte (8 bits), then, +25 will be represented as follows:

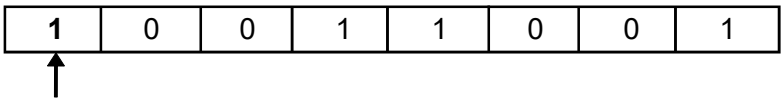
Number in binary notation as binary equivalent of 25_{10} is 11001



MSB (0 for + sign)

–25 will be represented as follows:

Number in binary notation as binary equivalent of 25_{10} is 11001



MSB (1 for - sign)

4.5 THE CONCEPT OF COMPLEMENT

Complement Systems Number theorists have known for hundreds of years that one decimal number can be subtracted from another by adding the difference of the subtrahend from all nines and adding back a carry. This is called taking the nine's complement of the subtrahend, or more formally, finding the diminished radix complement of the subtrahend. The advantage that complement systems give us over signed magnitude is that there is no need to process sign bits separately, but we can still easily check the sign of a number by looking at its high-order bit.

The concept of complements is used to represent signed number.

Let us consider a number system of base-r or radix-r. There are two types of complements.

i) The radix complement or the r's complement.

ii) The diminished radix complement or the (r–1)'s complement.

4.5.1 Diminished Radix Complement

Given a number N in base r having n digits, the (r–1)'s complement of N is defined as $(r^n – 1) – N$.

For decimal numbers, r = 10 and r – 1 = 9, so the 9's complement of N is $(10^n – 1) – N$

For example, 9's complement of 5642 is: $(10^4 – 1) – 5642 = 9999 – 5642 = 4357$

4.6 ONE'S (1'S) COMPLEMENT

It is important to note at this point that although we can find the nine's complement of any decimal number or the one's complement of any binary number, we are most interested in using complement notation to represent negative numbers. We know that performing a subtraction, such as 10–7, can be also be thought of as "adding the opposite'', as in 10 + (–7). Complement notation allows us to simplify subtraction by turning it into addition, but it also gives us a method to represent negative numbers. Because we do not wish to use a special bit to represent the sign (as we did in signed-

magnitude representation), we need to remember that if a number is negative, we should convert it to its complement. The result should have a 1 in the leftmost bit position to indicate the number is negative. If the number is positive, we do not have to convert it to its complement. All positive numbers should have a zero in the leftmost bit position.

4.6.1 One's (1's) Complement Arithmetic

The one's complement form of any number is obtained by simply changing each 0 in the number to 1 and each 1 in the number to 0.

For example, 1's complement of binary number 100010 will be 011101 and for 011011 will be 100100.

Example 3.1: Express 23_{10} and -9_{10} in 8-bit binary one's complement form.

$23_{10} = +(00010111)_2 = 00010111_2$

$-9_{10} = -(00001001)_2 = 11110110_2$

a) If Subtranhend is smaller than the Minuend

In this case we follow the following three simple steps:

Step 1: Find one's complement of the subtrahend.

Step 2: Proceed as in addition

Step 3: Ignored the carry and add 1 to the total end-around-carry.

Example 3.2: Add 23_{10} to -9_{10} using one's complement arithmetic.

Or, Subtract 9 from 23.

Solution:	0 0 0 1 0 1 1 1	(+23)
	+ 1 1 1 1 0 1 1 0	+(–9)
The last carry is added:	0 0 0 0 1 1 0 1	
	+1	
Sum is:	0 0 0 0 1 1 1 0	$+(14)_{10}$

b) If Subtranhend is larger than the minuend

In this case we follow the following three simple steps:

Step 1: Find one's complement of the subtrahend.

Step 2: Proceed as in addition.

Step 3: Complement the result and place a negative sign in front of the result.

Example 3.3: Add 9_{10} to -23_{10} using one's complement arithmetic.

Or, subtract 23 from 9

Solution:

	0 0 0 0 1 0 0 1	(+9)
	+1 1 1 0 1 0 0 0	+(–23)
No carry	1 1 1 1 0 0 0 1	

Now, complementing the result and putting a –ve sign we get,

$0\ 0\ 0\ 0\ 1\ 1\ 1\ 0 = -14_{10}$

The primary disadvantage of one's complement is that we still have two representations for zero: 00000000 and 11111111 i.e., +ve 0 and –ve 0. For this and other reasons, computer engineers long ago stopped using one's complement in favor of the more efficient two's complement representation for binary numbers.

Note that total numbers that can be represented using 1's complement are 2^N-1, where N is the word size (number of bits in a word).

4.7 TWO'S (2'S) COMPLEMENT ARITHMETIC

We know that all negative numbers can be represented in 2's complement form. Subtraction between two numbers can be achieved by adding the 2's complement of the number. Two's complement method represents positive numbers in their true form that means their binary equivalents and negative numbers in 2's complement form.

Now we will discuss how to represent of signed numbers in 2's complement form.

Consider the eight bit number 01011100, 2's complements of this number is 10100100. If we perform the addition between these numbers we have:

	0	1	0	1	1	1	0	0
	1	0	1	0	0	0	1	1
1	0	0	0	0	0	0	0	0

Since we are considering an eight bit number, so the 9^{th} bit (MSB) of the result can not be stored. Therefore, the final result is 00000000.

Since the addition of two number is 0, so one can be treated as the negative of the other number. So, 2's complement can be used to represent negative number.

The 2's complement form of any number is obtained by simply adding 1 to LSB of the 1's complement of that number.

For example 2's complement of 101110 is calculated as follows:

First find the 1's complement of 101110, which is 010001.

Now add 1 with LSB of 010001 is 010010, which is the required 2's complement of 101110.

For example,

i) Using 2's complement representation subtract 20 from 25

Solution: We have $25_{10} = 1\ 1\ 0\ 0\ 1$

Ignore the carry $-20_{10} = 0\ 1\ 1\ 0\ 0$ (2's Complement of 20_{10})

$1\ 0\ 0\ 1\ 0\ 1$

So, answer is +5

ii) Add (–20) and (–25) using Using 2's complement representation

Solution: We have $-25_{10} = 1\ 0\ 0\ 1\ 1\ 1$ (2's Complement of 25_{10})

$-20_{10} = 1\ 0\ 1\ 1\ 0\ 0$ (2's Complement of 20_{10})

Ignore carry $1\ 0\ 1\ 0\ 0\ 1\ 1$

and the result is –(2's complement of 010011)

So, answer is -101101 = - 45

iii) Using 2's complement representation subtract 25 from 20.

Solution: We have $-25_{10} = 0\ 0\ 1\ 1\ 1$ (2's Complement of 25_{10})

$+\ 20_{10} = 1\ 0\ 1\ 0\ 0$

$1\ 1\ 0\ 1\ 1$

As no carry, so the answer is: –(2's complement of 11011)

$= -0\ 0\ 1\ 0\ 1$

$= -5$

Note that total numbers which a word of N bits can present, using 2's complement representation, are 2^N.



CHECK YOUR PROGRESS

Q.1: What are three different schemes or methods to represent signed (negative) and unsigned integer?

Q.2: What is the range of signed integer representation of n bits?

Q.3: What is the format of the floating point representation?

Q.4: What is the signed and magnitude representation of (–20) in 8 bit word length?

Q.5: Find 1's and 2's complement of followings:

i) 100010 ii) 1010110

Q.6: State whether the following statements are true (T) or false (F):

a) For n-bit number, the range for natural number is from 0 to 2^n–1.

b) Most Significant Bit (MSB) is used to indicate the sign and it is termed as signed bit. 0 is used for negative number.

c) The format of floating point representation is The format is: $Mantissa * R^{Exponent}$.

d) The two's complement form of any number is obtained by simply changing each 0 in the number to 1 and each 1 in the number to 0.

e) Given a number N in base r having n digits, the (r–1)'s complement of N is defined as (r^n–1) – N.



4.8 LET US SUM UP

- Computer arithmetic is a branch of computer engineering that deals with methods of representing integers and real values in digital systems and efficient algorithms for manipulating such numbers by means of hardware circuits or software routines.
- There are three different schemes or methods to represent signed (negative) and unsigned integer. These are:

i) Signed and Magnitude form.
ii) 1's complement form.
iii) 2's complement form

- We know that for n-bit number, the range for natural number is from 0 to 2^n–1.
- In digital computers, floating-point numbers consist of three parts: (i) a Sign bit, (ii) an Exponent part (representing the exponent on a power of 2), and (iii) a fractional part called a Significant (which is a fancy word for a Mantissa).
- Generally, Most Significant Bit (MSB) is used to indicate the sign and it is termed as signed bit. 0 in signed bit indicates positive number and 1 in signed bit indicates negative number.
- Consider a number system of base-r or radix-r. There are two types of complements.
  i) The radix complement or the r's complement.
  ii) The diminished radix complement or the (r–1)'s complement
- Given a number N in base r having n digits, the (r–1)'s complement of N is defined as $(r^n–1) – N$ **.**
- The one's complement form of any number is obtained by simply changing each 0 in the number to 1 and each 1 in the number to 0.
- The total numbers that can be represented using 1's complement are 2^N–1, where N is the word size (number of bits in a word).
- The 2's complement form of any number is obtained by simply adding 1 to LSB of the 1's complement of that number.
- The total numbers which a word of N bits can present, using 2's complement representation, are 2^N.



4.9 FURTHER READING

1) Chaudhuri, P. Pal. (2nd Edition, 2003). *Compter Organization and Design*. PHI Learning Pvt. Ltd.
2) Hammacher, Carl. (Fifth Edition) (International Edition, 2002). *Computer Organization*. McGrawHill.

3) Rajaraman, V. (Fouth Edition, 2008). *Fundamentals of Computers*.

4) Stallings, William. (2004). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.

4.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: There are three different schemes or methods to represent signed (negative) and unsigned integer. These are

i) Signed and Magnitude form

ii) 1's complement form

iii) 2's complement form

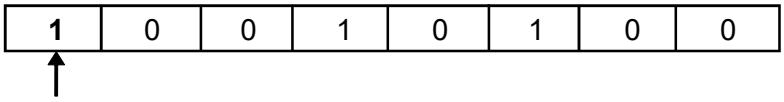
Ans. to Q. No. 2: For n-bit representations of signed integer, the range is from $-2^{n-1}-1$ to $+2^{n-1}-1$

Ans. to Q. No. 3: In the representation of floating point, numbers are represented by a Mantissa comprising the significant digits and an Exponent part of Radix R.

The format is: $Mantissa * R^{Exponent}$

Ans. to Q. No. 4: –20 will be represented as follows:

Number in binary notation as binary equivalent of 20_{10} is 10100



MSB (1 for - sign)

Ans. to Q. No. 5: The 1's and 2's complement of followings are

i) 1's complement of 100010 is 011101

2's complement of 100010 is 011110

ii) 1's complement of 1010110 is 0101001

2's complement of 1010110 is 0101010

Ans. to Q. No. 6: a) True, b) False, c) True, d) False, e) True.



4.11 MODEL QUESTIONS

Q.1: What are different schemes to represent signed and unsigned integer in computer arithmetic?

Q.2: How is signed integer represented in signed and magnitude form?

Q.3: What do you mean by floating-point representation of any number in digital computer? What are its different parts?

Q.4: What is 1's complement ? Explain with example.

Q.5: What do you mean by 2's complement? Explain how can a negative integer represented using 2's complement concept.

*** ***** ***