TOPIC 3 – INTEGERS

LEARNING OUTCOMES

By the end of this topics, you will be able to:

1. describe unsigned, BCD number representation, sign magnitude and negative numbers.
2. explain overflow, carry, floating points, format and exponents.
3. use unsigned, BCD number representation, sign magnitude and negative numbers.
4. use overflow, carry, floating points, format and exponents.

INTRODUCTION

This topic explains about unsigned, BCD number representation, signed magnitude and negative numbers. It also covers the overflow, floating points, format and exponents.

3.1 UNSIGNED

Unsigned represents the positive integers. For example, 157_{10} can be represented as shown in Table 3.1:

Table 3.1

Position	7	6	5	4	3	2	1	0
Bit pattern	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
Contribution	2^7			2^4	2^3	2^2		2^0

Following are the advantages and disadvantages of unsigned notation.

Advantages:

- It consists of one representation of zero.
- It is simple in addition.

Disadvantages

- Negative numbers cannot be represented.
- The need of different notation to represent negative numbers.

SELF CHECK 3.1

1. What is unsigned data representation?

3.2 BINARY-CODED-DECIMAL (BCD) NUMBER REPRESENTATION

BCD number representation is a way to express each of the decimal digits with a binary code. In BCD, the decimal digit is represented by a 4-bit binary number. It can represent sixteen numbers (0000 to 1111) however in BCS code, it is only first ten of these are used (000 to 1001). The remaining are invalid in BCD.

It is shown in Table 3.2.

Table 3.2

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

The advantages of BCD representation are:

- The code is similar to decimal system.
- It is required to remember binary equivalent of decimal numbers 0 to 9 only.

The disadvantages of BCD representation are:

- There are different rules for addition and subtraction BCD.
- The BCD arithmetic a bit complicated.
- BCD is less efficient than binary due to it needs a greater number of bits than binary for decimal number representation.

SELF CHECK 3.2

1. Define BCD number representation.
2. List TWO (2) advantages and TWO (2) disadvantages of BCD number representation.

3.3 SIGN AND MAGNITUDE

Sign magnitude notation is one of the methods used to represent the negative numbers. In signed magnitude the left-most bit represents the sign of the integer where zero (0) for positive numbers and one (1) for negative numbers. The remaining bits represent magnitude of the numbers.

Example 1:
0 0 1 1 0 1 0 1 = +53 Positive magnitude bits sign

1 0 1 1 0 1 0 1 = -53 Negative magnitude bits sign

Example 2:

Suppose 10011101_2 is a signed magnitude representation. The sign bit is 1, then the number represented is negative. It can be represented as in Table 3.3:

Table 3.3

Position	7	6	5	4	3	2	1	0
Bit pattern	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
Contribution	**-**			2^4	2^3	2^2		2^0

The magnitude is 0011101_2 with a value $2^4+2^3+2^2+2^0= 29_{10}$. So, the number represented by 10011101_2 is –29.

The advantage of signed magnitude representation is it represents positive and negative numbers. The disadvantages of signed magnitude representation; it has two representations of zero and arithmetic operations are difficult.

SELF CHECK 3.3

1. Convert the following decimal numbers to signed magnitude binary number:
   - 30_{10}
   - 25_{10}
2. Convert the following binary numbers to signed magnitude decimal number:
   - 10101010_2
   - 11110000_2

3.4 USE OF COMPLEMENTS TO REPRESENT NEGATIVE NUMBERS

The complements are used to represent negative numbers. It can be categorized into ones complement and two's complement.

3.4.1 ONE'S COMPLEMENT

In one's complement, the positive value of the number in binary will be written. Then, each bit number will be reversed so 1's become 0's and 0's become 1's.

If the decimal number is positive, simply convert it to binary. If the decimal number is negative, write the positive value of the number in binary.

For example, given the binary number 0101_2 then the result after reversal is 1010_2. Refer Table 3.4.

Table 3.4

Given Number	0	1	0	1
One's Complement	1	0	1	0

3.4.2 TWO'S COMPLEMENT

The two's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of one's complement of the number. It is the most used representation for integers.

All positive numbers begin with 0, all negative numbers begin with 1 and one representation of zero.

It can be summarized as two's complement = one's complement + 1

For example, given the binary number 10101_2 then the result in one's complement is 01010_2. The result in two's complement is 01011_2. Refer Table 3.5.

Table 3.5

Given Number	1	0	1	0	1
One's Complement	0	1	0	1	0
Add 1					1
Two's Complement	0	1	0	1	1

SELF CHECK 3.4

1. Write the following binary numbers in One's complement representation:
   - 10101010_2
   - 11110000_2
2. Write the following binary numbers in Two's complement representation:
   - 10101010_2
   - 11110000_2

3.5 OVERFLOW AND CARRY

Numbers are added or subtracted on the number circle by traversing **clockwise for addition** and **counterclockwise for subtraction**.

Unlike the number line (a) where overflow never occurs, overflow occurs when a transition is made from +3 to -4 while proceeding around the number circle when adding, or from -4 to +3 while subtracting. For example:

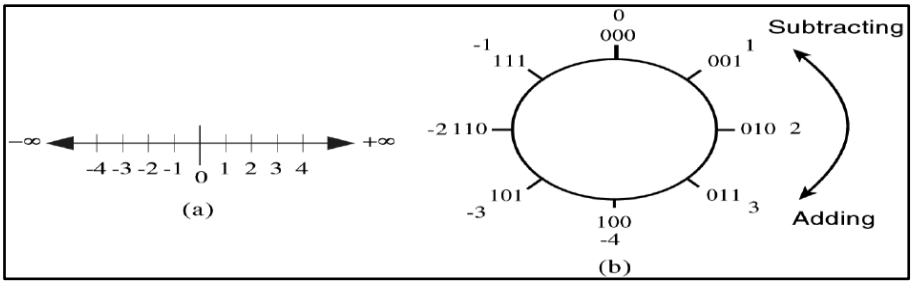

Figure 3.18

Usually, overflow occurs when the addition of two positive numbers produces a negative result, or when the addition of two negative numbers produces a positive result. Unlike signs, adding the operands never produces an overflow.

Notice that discarding the carry out of the most significant bit during two's complement addition is a normal occurrence, and does not by itself indicate overflow.

For example, consider adding $(7 + 1 = 8)_{10}$, which produces a result of -8_{10} in an 8-bit two's complement format. It cannot be represented with 4-bit two's complement number as it is out of range. The carry is also 0. So,

0001 = 1

+ 0111 = 7

1000 = -8 (*not* 8 because the sign bit is 1.)

There are two conditions of overflow. It occurs when two negative numbers are added and an answer comes positive or two positive numbers are added and an answer comes as negative.

SELF CHECK 3.5

1. Explain the overflow condition in data representation.

3.6 FLOATING POINTS

Floating point arithmetic differs from integer arithmetic where the exponents must be handled as well as the magnitudes of the operands. The exponents of the operands must be made equal for addition and subtraction. The fractions are then added or subtracted as appropriate, and the result is normalized.

For example: Perform the floating-point operation for $(.101 \times 2^3 + .111 \times 2^4)_2$

1. First, start by adjusting the *smaller* exponent to be equal to the larger exponent, and adjust the fraction accordingly.
2. Then it became $.101 \times 2^3 = .010 \times 2^4$, losing $.001 \times 2^3$ of precision in the process.
3. The result is $(.010 + .111) \times 2^4 = 1.001 \times 2^4 = .1001 \times 2^5$, and rounding to three significant digits, $.100 \times 2^5$, and it lost another 0.001×2^4 in the rounding process.
4. If it is simply added, the numbers will use as much precision and then applied rounding only in the final normalization step.
5. Then the calculation would go like this:
   $= .101 \times 2^3 + .111 \times 2^4$
   $= .0101 \times 2^4 + .111 \times 2^4$
   $= 1.0011 \times 2^4$
6. Normalize yields $\mathbf{.10011 \times 2^5}$, and round up to three significant digits using the round to nearest even method yields $\mathbf{.101 \times 2^5}$.
7. According to the IEEE 754 standard, the final result should be the same as if the maximum precision needed is used before applying the rounding method, and so the correct result is $\mathbf{.101 \times 2^5}$.

SELF CHECK 3.6

1. Describe the floating point in data representation.

3.7 FORMAT AND EXPONENTS

The exponents consist of 8 bits for single precision and 11 bits for double precision. With 8 bits, the exponents are represented between -126 and + 127. All-zeroes value is reserved for the zeroes and denormalized numbers. All-ones value is reserved for the infinities and NaNs (Not a Number).

Exponents are represented using a biased notation where the **Stored value = actual exponent + bias.**

As an example, for 8-bit exponents, bias is 127 where it stored value of 1 corresponds to –126 and stored value of 254 corresponds to +127. The 0 and 255 are reserved for special values.

Biased notation simplifies the comparisons if two normalized floating-point numbers have different exponents, the one with the bigger exponent is the bigger of the two.

SELF CHECK 3.7

1. Explain the function of exponents.

SUMMARY

In this topic you have learnt that:

Unsigned, BCD number representation, sign magnitude, negative numbers, overflow, carry, floating points, format and exponents.

KEY TERMS

Arithmetic Deals with numbers and numerical computation.

REFERENCES

Stallings, W., (2019). *Computer Organization and Architecture Designing for Performance.* 11th ed. New York: Pearson.