

Computer Arithmetic

Introduction:

Data is manipulated by using the arithmetic instructions in digital computers. Data is manipulated to produce results necessary to give solution for the computation problems. The Addition, subtraction, multiplication and division are the four basic arithmetic operations. If we want then we can derive other operations by using these four operations.

To execute arithmetic operations there is a separate section called arithmetic processing unit in central processing unit. The arithmetic instructions are performed generally on binary or decimal data. Fixed-point numbers are used to represent integers or fractions. We can have signed or unsigned negative numbers. Fixed-point addition is the simplest arithmetic operation.

If we want to solve a problem then we use a sequence of well-defined steps. These steps are collectively called algorithm. To solve various problems we give algorithms.

In order to solve the computational problems, arithmetic instructions are used in digital computers that manipulate data. These instructions perform arithmetic calculations.

And these instructions perform a great activity in processing data in a digital computer. As we already stated that with the four basic arithmetic operations addition, subtraction, multiplication and division, it is possible to derive other arithmetic operations and solve scientific problems by means of numerical analysis methods.

A processor has an arithmetic processor(as a sub part of it) that executes arithmetic operations. The data type, assumed to reside in processor, registers during the execution of an arithmetic instruction. Negative numbers may be in a signed magnitude or signed complement representation. There are three ways of representing negative fixed point - binary numbers signed magnitude, signed 1's complement or signed 2's complement. Most computers use the signed magnitude representation for the mantissa.

Addition and Subtraction :

Addition and Subtraction with Signed –Magnitude Data

We designate the magnitude of the two numbers by A and B. Where the signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed. These conditions are listed in the first column of Table 4.1. The other columns in the table show the actual operation to be performed with the magnitude of the numbers. The last column is needed to present a negative zero. In other words, when two equal numbers are subtracted, the result should be +0 not -0.

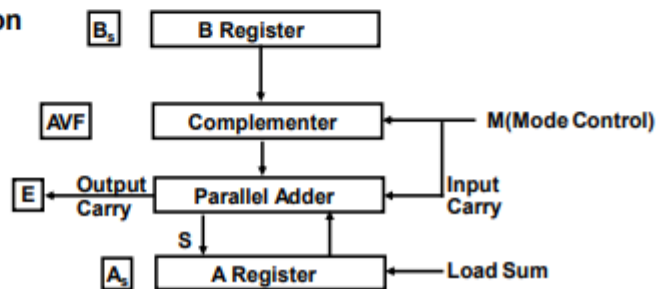
The algorithms for addition and subtraction are derived from the table and can be stated as follows (the words parentheses should be used for the subtraction algorithm)

SIGNED MAGNITUDE ADDITION AND SUBTRACTION

Addition: $A + B$; A: Augend; B: Addend
 Subtraction: $A - B$: A: Minuend; B: Subtrahend

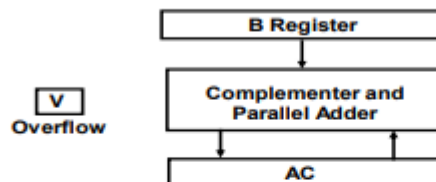
| Operation | Add Magnitude | Subtract Magnitude | | |
|---------------|---------------|--------------------|--------------|--------------|
| | | When $A > B$ | When $A < B$ | When $A = B$ |
| $(+A) + (+B)$ | $+(A + B)$ | $+(A - B)$ | $-(B - A)$ | $+(A - B)$ |
| $(+A) + (-B)$ | | | | |
| $(-A) + (+B)$ | | $-(A - B)$ | $+(B - A)$ | $+(A - B)$ |
| $(-A) + (-B)$ | $-(A + B)$ | | | |
| $(+A) - (+B)$ | | $+(A - B)$ | $-(B - A)$ | $+(A - B)$ |
| $(+A) - (-B)$ | $+(A + B)$ | | | |
| $(-A) - (+B)$ | $-(A + B)$ | | | |
| $(-A) - (-B)$ | | $-(A - B)$ | $+(B - A)$ | $+(A - B)$ |

Hardware Implementation

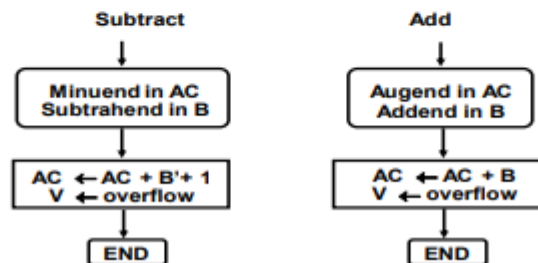


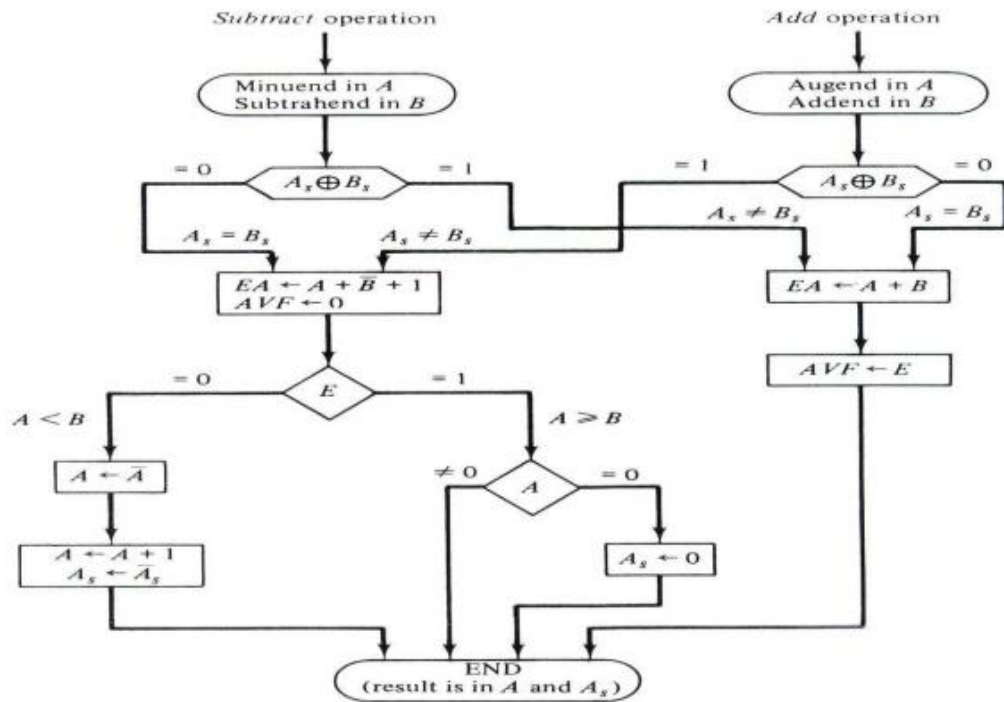
SIGNED 2'S COMPLEMENT ADDITION AND SUBTRACTION

Hardware



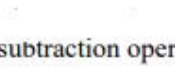
Algorithm





- The flowchart is shown The two signs A, and B, are compared by an exclusive-OR gate.

If the output of the gate is 0 the signs are identical; If it is 1, the signs are different.

- For an add operation, identical signs dictate that the magnitudes be added. For a subtract operation, different signs dictate that the magnitudes be added.
- The magnitudes are added with a microoperation $EA \leftarrow A + B$, where EA is a register that combines E and A. The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred into the add-overflow flip-flop AVF.
- The two magnitudes are subtracted if the signs are different for an add operation or identical for a subtract operation. The magnitudes are subtracted by adding A to the 2's complemented B. No overflow can occur if the numbers are subtracted so AVF is cleared to 0.
- 1 in E indicates that $A \geq B$ and the number in A is the correct result. If this number is zero, the sign A must be made positive to avoid a negative zero.
- 0 in E indicates that $A < B$. For this case it is necessary to take the 2's complement of the value in A. The operation can be done with one microoperation $A \leftarrow A' + 1$.
- However, we assume that the A register has circuits for microoperations complement and increment, so the 2's complement is obtained from these two microoperations.
- In other paths of the flowchart, the sign of the result is the same as the sign of A, so no change in A is required. However, when $A < B$, the sign of the result is the complement of the original sign of A. It is then necessary to complement A, to obtain the correct sign.
- The final result is found in register A and its sign in As. The value in AVF provides an overflow indication. The final value of E is immaterial.
-  a block diagram of the hardware for implementing the addition and subtraction operations.
- It consists of registers A and B and sign flip-flops As and Bs.
- Subtraction is done by adding A to the 2's complement of B.
- The output carry is transferred to flip-flop E, where it can be checked to determine the relative magnitudes of two numbers.