# Addition and Subtraction:→

* Most Computers use the signed 2's complement representation when performing arithmetic operations with integers.
* For floating-point operations, most Computers use the signed-magnitude representation for Mantissa.

## Addition and Subtraction with Signed-Magnitude

### Data:→

* we designate the magnitude of the two numbers by A and B. When the signed numbers are added or subtracted, we find that there are eight different condition to consider, depending on the sign of the numbers and operation performed. These condition are showed in the below (fig) Table.

Table: Addition and subtraction of signed Magnitude numbers

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

* The algorithm for addition and subtraction and derived from the table and can be stated as follows (The word inside parentheses should be used for the subtraction algorithm)

## Addition (Subtraction) Algorithm:→

* When the signal of A and B are identical (different), add the two magnitude and attach the sign of A to the result.
* When the sign of A and B are different (identical), compare the magnitudes and subtract the smaller number from the larger.
* Choose the sign of the result to be the same as A if A > B or the complement of the sign of A if A < B.
* If the two magnitudes are equal, subtract B from A and make the sign of the result positive.
* The two algorithms are similar except for the sign comparision. The procedure to be followed for identical signs in the addition algorithm is the same as for as different signs in the subtraction algorithm, and vice versa.

## Hardware Implementation:→

* Let A and B be two registers that holds the [illegible] two numbers, and As, and Bs be two F-F that holds the corresponding signs.

* The result of the operation may be transferred to a third register; however, a saving is achieved if the result is transferred into A and Ap. Then A and Ap together form an accumulator register.
* First, a parallel-adder is needed to perform the micro operation A+B.
* Second the comparator circuit is needed to establish if A>B, A=B, or A<B.
* Third two parallel-subtractor circuits are needed to perform the micro-operation A-B and B-A.
* The sign relationship can be determined from an Ex-OR gate with As and Bs as i/ps.
* Careful investigation of the alternatives reveals that the use of 2's complement of for subtraction and comparision is an efficient procedure that requires only one adder and a complementor.
* Figure on the next page shows the HW for implementing the addition and subtraction operation.
* It consists of register A and B and sign F-F As and Bs. Subtraction is done by adding A to the 2's complement of B. The o/p carry is transferred to F-F E, where it can be checked to determine the relative magnitude of the

of the two numbers. The add-overflow F-F AVF holds the overflow bit when A and B are added.

* The A register provides other microoperations that may be needed when we specify the sequence of steps in the algorithm.

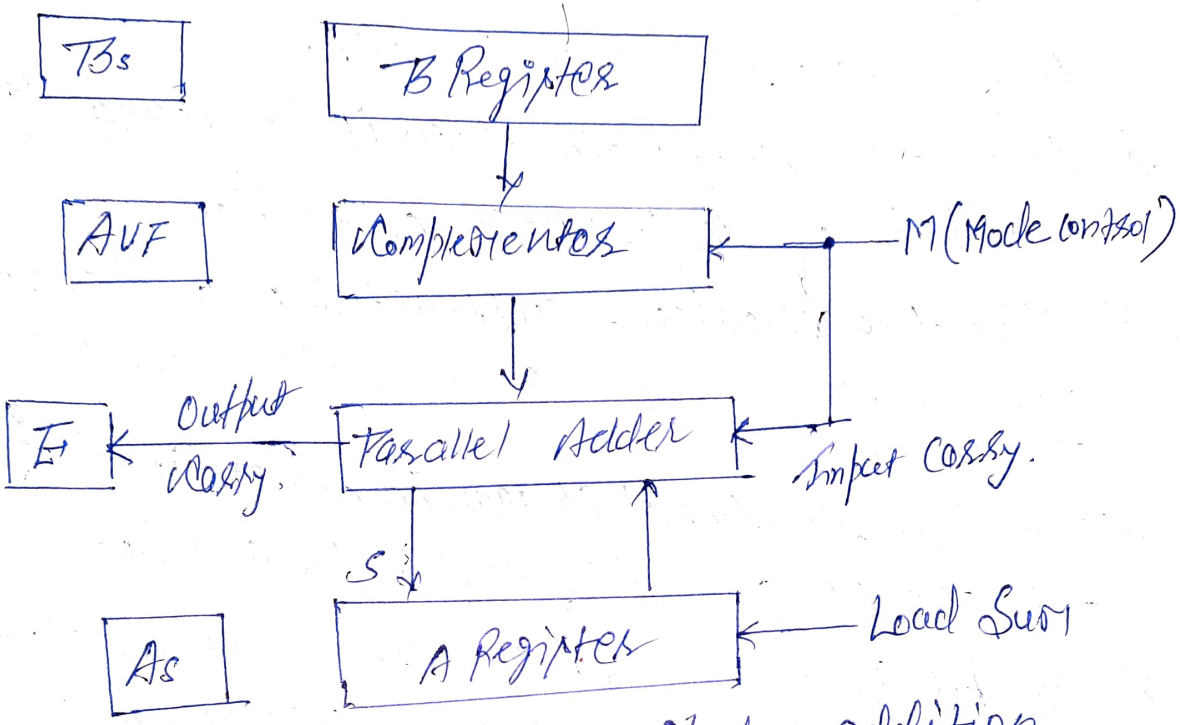



Fig: HW for signed-magnitude addition and subtraction.

* The addition of A plus B is done through the parallel adder. The S (sum) o/p of the adder is applied to the i/p of the A register.
* The complementer consists of Ex-OR gates and the parallel adder consists of full-adder circuit.
* The M signal is also applied to the i/p carry of the adder. When M=0 the o/p of B is transferred to the adder, the i/p carry is 0, and the o/p of the adder is equal to the sum A+B.

* When M=1, the 1's complement of B is applied to the adder, the i/p carry is 1, and output S= A+B+1. This is equal to A+2's complement of B, which is equal to the subtraction A-B.

## Hardware Algorithm:→

* The flowchart for the HW algorithm is presented in the below fig.
* The two signs As and Bs are compared by an Ex-OR gates. If the o/p is of the gate is 0, the signs are identical; if it is 1, the signs are different.
* The magnitudes are added with a microoperation EA ← A+B, where EA is a register that combines E and A.
* The carry in E is transferred into the add-overflow ~~with~~ ~~A~~ F-F AVF.
* The magnitudes are subtracted by adding A to the 2's complement of B. No overflow can occur if the number are subtracted so AVF is cleared to 0.
* A ~~1~~ 1 in E indicates that A<B for this case it is necessary to takes the 2's complement of the value in A. This operation can be done with one micro-operation A ← Ā+1.
* However, we assumes that the A register has circuits for microoperations complement and increment, so the 2's complement is obtained from these two micro-operations.

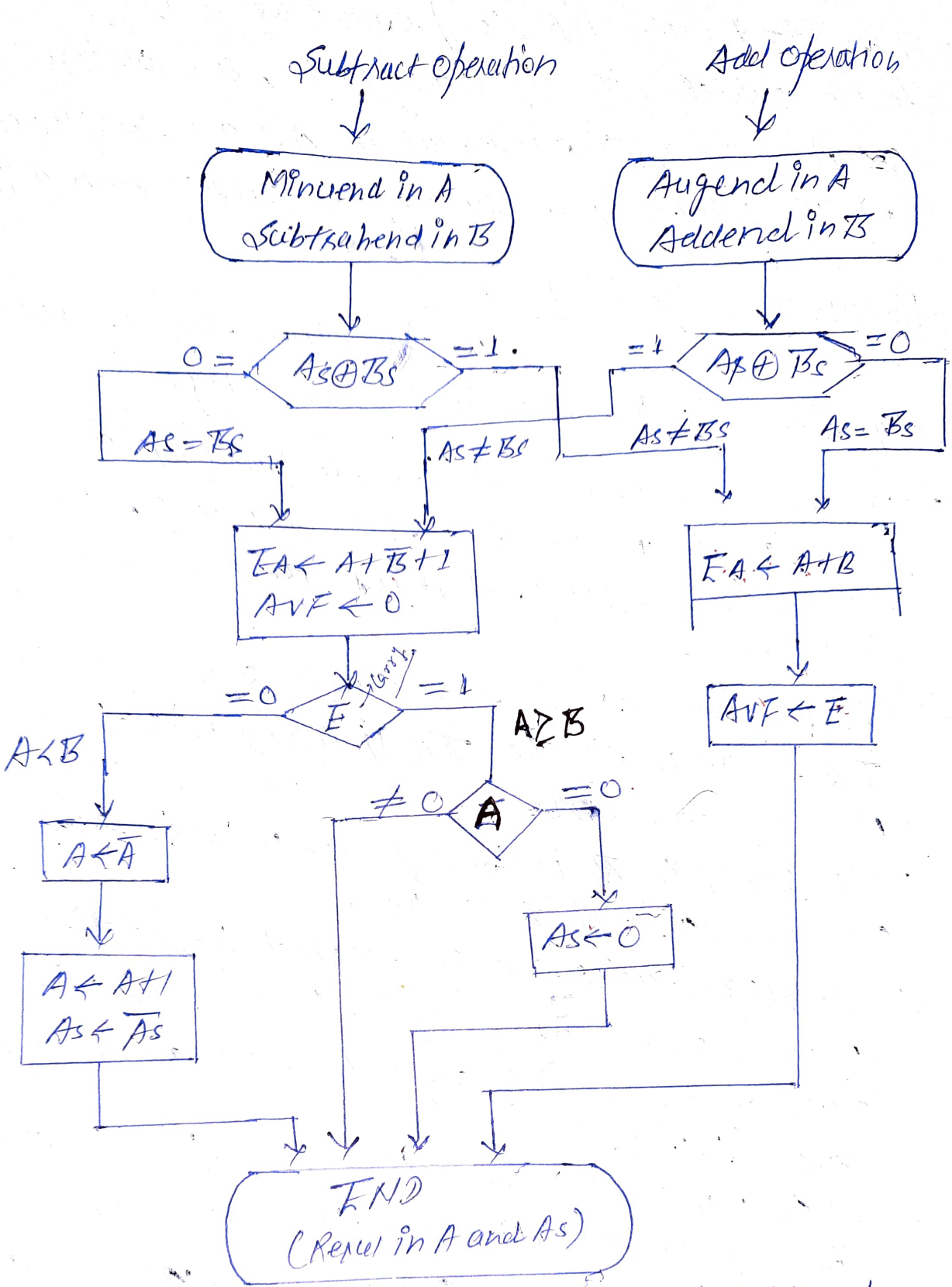



Fig: Flow chart for Add and Subtract

* In other parts of the flowchart, the sign of the result is the same as the sign of A, so no change in As is required.

* However, when A<B, the sign of the result is the complement of the original sign of A. It is necessary to to complement As to obtain the correct sign.
* The Final result is found in the register A and its sign in As. The value of AVF provides an overflow indication. The final value of F is immaterial

## Addition and subtraction with Signed 2's Complement Data:→

* The addition of two numbers in signed-2's complement form consists of adding the number with the sign bits treated the same as the other bits of the number.
* A carry-out of the sign-bit position is discarded
* The subtraction consists of first taking the 2's complement of the subtrahend and then adding it to the Minuend.
* When two numbers of n digits each are added and the sum occupies n+1 digits, we say that an overflow occured. An overflow can be detected by inspecting the last two carries out of the addition. When the two carries are applied to an Ex-OR gate, the overflow is detected when the O/p of the gate is equal to 1.
* The register configuration for the HW HW implementation is shown in fig on the next page.

* We name the A ~~Register~~ Register AC and the B Register BR. The leftmost bit in AC and BR represent the sign bits of the numbers.
* The two sign bits are added or subtracted together with the other bits in the complementer and parallel adder.
* The overflow ~~F~~ V is set to 1 if there is an overflow. The o/p carry in this case is discarded.

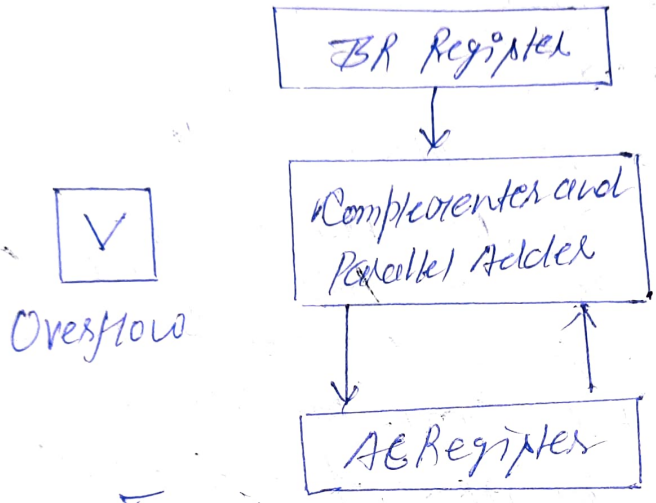
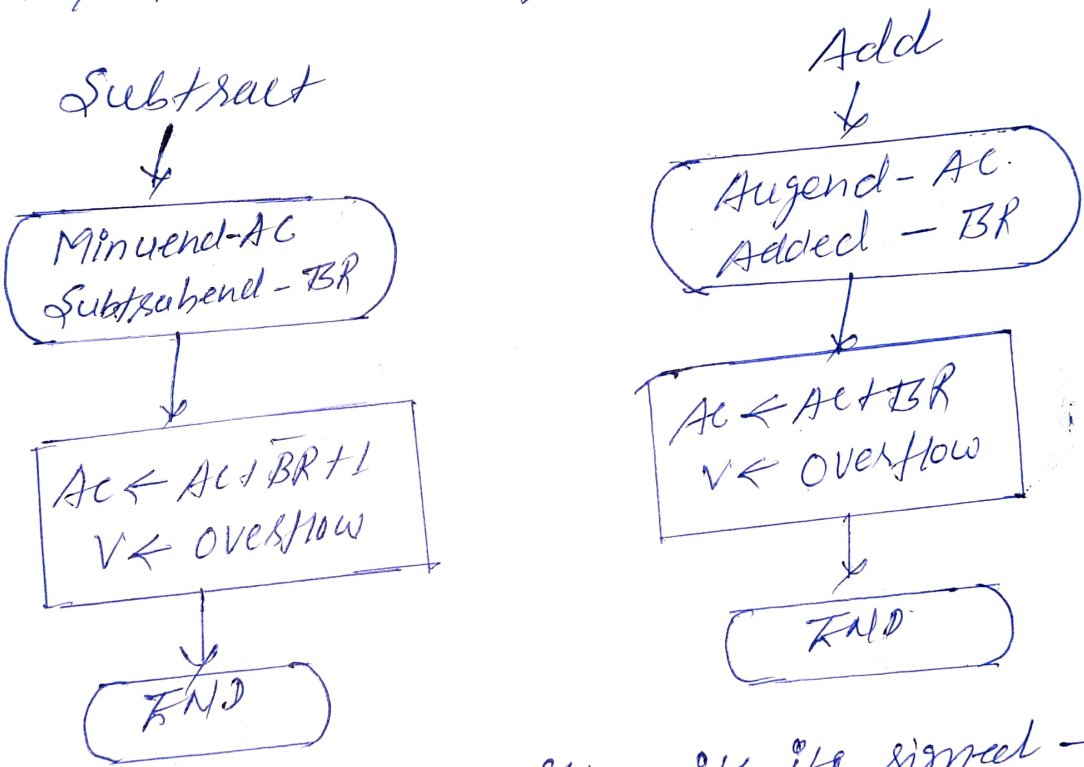



Fig: HW for signed 2's complement addition and subtraction.

* The algorithm for adding and subtracting two binary number in signed 2's complement representation is shown in the flowchart in the next page.
* The sum is obtained by adding the content of AC and BR (including their sign bits). The overflow bit V is set to 1 if the Ex-OR of the last two carries is 1, and it is cleared to 0 otherwise.

* The subtraction operation is accomplished by adding the content of AC to the 2's complement of BR. Taking the 2's complement of BR has the effect of changing a positive number to negative, and vice versa.
* An overflow must be checked during this operation because the two numbers added could have the same sign.
* The programmer must realize that if an overflow occurs, there will be an erroneous result in the AC register.




* Comparing this algorithm with its signed-magnitude counterpart, we noted that it is much simpler to add and subtract numbers if negative numbers are maintained in signed-2's complement representation. That's why most of computers adopt this representation.