**EXPERIMENT NO. 07**

|  |  |
| --- | --- |
| **DATE OF PERFORMANCE:** | **GRADE:** |
| **DATE OF ASSESSMENT:** | **SIGNATURE OF LECTURER/ TTA:** |

**AIM:  Introduction to Computer Arithmetic and Number Systems.**

**THEORY:**

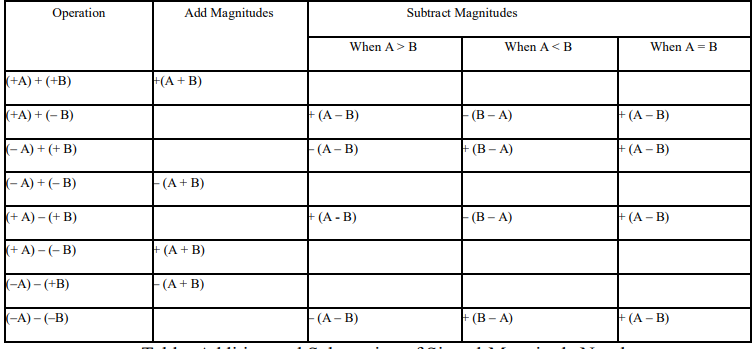
**We will discuss about different computer arithmetic operations:Addition , subtraction, multiplication, decimal unit.**

**Arithmetic Addition and Arithmetic Subtraction:**

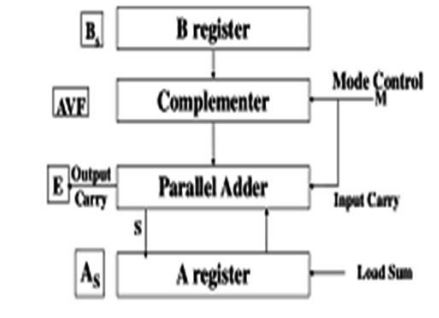
**The addition on two numbers in the signed-magnitude system follows the rules of ordinary arithmetic.**

**If the signs are the same,we add two magnitudes and give the sum the common sign.**

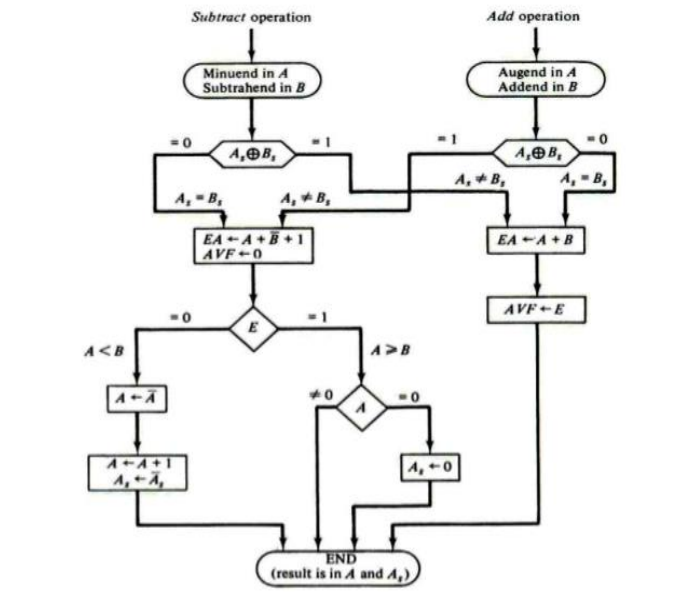
**We designate the magnitude of the two numbers by A and B. Where the signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed.**

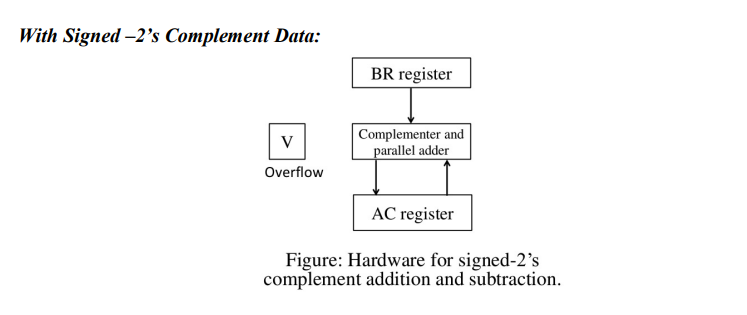


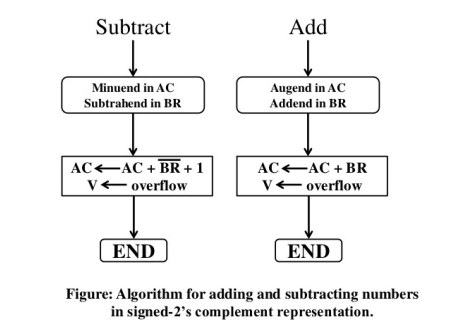
**When the signs of A and B are same, add the two magnitudes and attach the sign of result is that of A. When the signs of A and B are not same, compare the magnitudes and subtract the smaller number from the larger. Choose the sign of the result to be the same as A, if A > B or the complement of the sign of A if A < B. If the two magnitudes are equal, subtract B from A and make the sign of the result will be positive.**

****

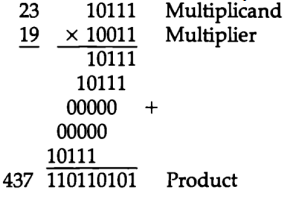
**Figure: Hardware Architecture for Addition and Subtraction of Signed-Magnitude Numbers.**



**Multiplication:** 



**Signed-Magnitude complements: Multiplication of two fixed-point binary numbers in signed magnitude representation is done with paper and pencil by a process of successive shift and adds operations. This process is best illustrated with a numerical example:**



**This process looks at successive bits of the multiplier, least significant bit first.**

**If the multiplier bit is 1, the multiplicand is copied as it is; otherwise, we copy zeros.**

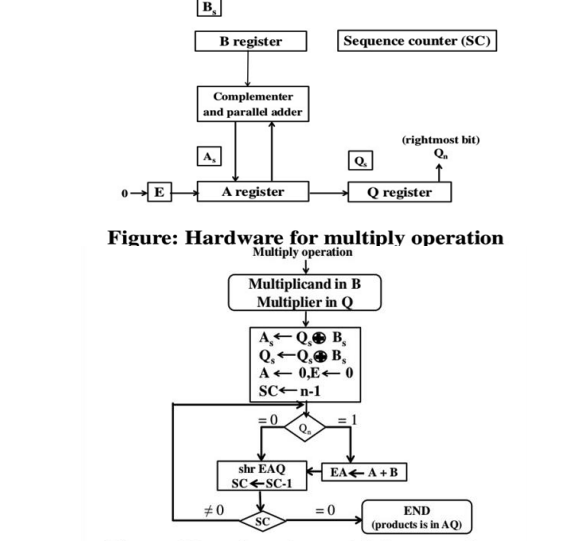
**Now we shift numbers copied down one position to the left from the previous numbers.**

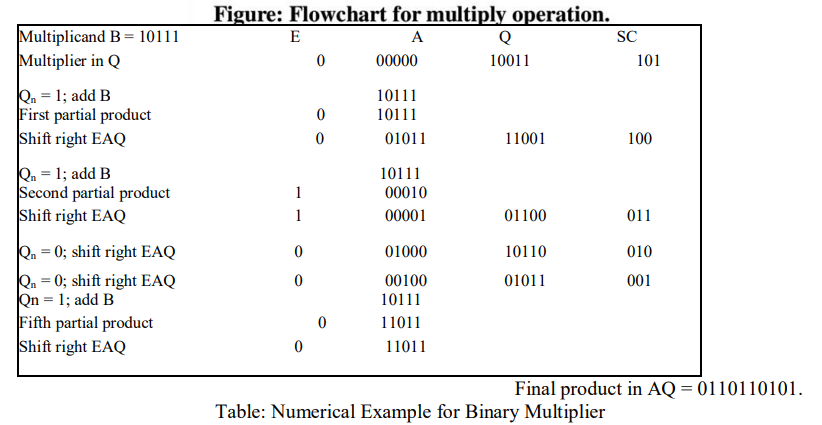
**Finally, the numbers are added and their sum produces the product. When multiplication is implemented in a digital computer, we change the process slightly.**

**Here, instead of providing registers to store and add simultaneously as many binary numbers as there are bits in the multiplier, it is convenient to provide an adder for the summation of only two binary numbers, and successively accumulate the partial products in a register.**

**Second, instead of shifting the multiplicand to left, the partial product is shifted to the right, which results in leaving the partial product and the multiplicand in the required relative positions.**

**Now, when the corresponding bit of the multiplier is 0, there is no need to add all zeros to the partial product since it will not alter its value. The hardware for multiplication consists of the equipment given in below Figure.**



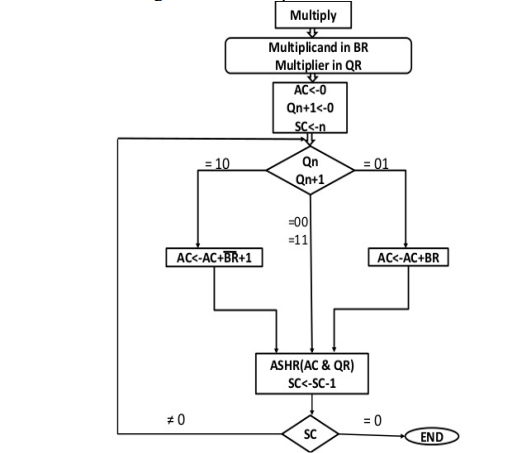


**Booth’s Multiplication Algorithm:**

**The booth algorithm is a multiplication algorithm that allows us to multiply the two signed binary integers in 2's complement, respectively.**

**It is also used to speed up the performance of the multiplication process. It is very efficient too. It works on the string bits 0's in the multiplier that requires no additional bit only shift the right-most string bits and a string of 1's in a multiplier bit weight 2k to weight 2m that can be considered as 2k+ 1 - 2m.**

**Following is the pictorial representation of the Booth's Algorithm:**



**In the above flowchart, initially, AC and Qn + 1 bits are set to 0, and the SC** **is a sequence counter that represents the total bits set n, which is equal to the number of bits in the multiplier.**

**There are BR** **that represent the multiplicand bits, and QR represents the multiplier bits**. **After that, we encountered two bits of the multiplier as Qn and Qn + 1, where Qn represents the last bit of QR, and Qn + 1represents the incremented bit of Qn by 1.**

**Suppose two bits of the multiplier is equal to 10; it means that we have to subtract the multiplier from the partial product in the accumulator AC and then perform the arithmetic shift operation (ashr).**

**If the two of the multipliers equal to 01, it means we need to perform the addition of the multiplicand to the partial product in accumulator AC and then perform the arithmetic shift operation (ashr), including Qn + 1**.

**The arithmetic shift operation is used in Booth's algorithm to shift AC and QR bits to the right by one and remains the sign bit in AC unchanged.**

**And the sequence counter is continuously decremented till the computational loop is repeated, equal to the number of bits (n).**

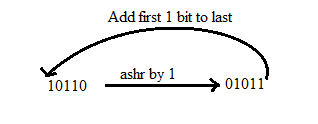
**The Booth Algorithm:**

1. **Set the Multiplicand and Multiplier binary bits as M and Q, respectively.**
2. **Initially, we set the AC and Qn + 1 registers value to 0.**
3. **SC represents the number of Multiplier bits (Q), and it is a sequence counter that is continuously decremented till equal to the number of bits (n) or reached to 0.**
4. **A Qn represents the last bit of the Q, and the Qn+1 shows the incremented bit of Qn by 1.**
5. **On each cycle of the booth algorithm, Qn and Qn + 1 bits will be checked on the following parameters as follows:**
   1. **When two bits Qn and Qn + 1 are 00 or 11, we simply perform the arithmetic shift right operation (ashr) to the partial product AC. And the bits of Qn and Qn + 1 is incremented by 1 bit.**
   2. **If the bits of Qn and Qn + 1 is shows to 01, the multiplicand bits (M) will be added to the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.**
   3. **If the bits of Qn and Qn + 1 is shows to 10, the multiplicand bits (M) will be subtracted from the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.**
6. **The operation continuously works till we reached n - 1 bit in the booth algorithm.**
7. **Results of the Multiplication binary bits will be stored in the AC and QR registers.**

**There are two methods used in Booth's Algorithm:**

**RSC (Right Shift Circular)**

**It shifts the right-most bit of the binary number, and then it is added to the beginning of the binary bits.**



### RSA (Right Shift Arithmetic)

**It adds the two binary bits and then shift the result to the right by 1-bit position.**

**Number systems:**

**The language we use to communicate with each other is comprised of words and characters. We understand numbers, characters and words. But this type of data is not suitable for computers. Computers only understand the numbers.**

**So, when we enter data, the data is converted into electronic pulse. Each pulse is identified as code and the code is converted into numeric format by ASCII. It gives each number, character and symbol a numeric value (number) that a computer understands. So to understand the language of computers, one must be familiar with the number systems.**

**The Number Systems used in computers are:**

* **Binary number system**
* **Octal number system**
* **Decimal number system**
* **Hexadecimal number system**

## **Binary number system**

**It has only two digits '0' and '1' so its base is 2. Accordingly, In this number system, there are only two types of electronic pulses; absence of electronic pulse which represents '0'and presence of electronic pulse which represents '1'.**

**Each digit is called a bit. A group of four bits (1101) is called a nibble and group of eight bits (11001010) is called a byte.**

**The position of each digit in a binary number represents a specific power of the base (2) of the number system.**

## **Octal number system**

**It has eight digits (0, 1, 2, 3, 4, 5, 6, 7) so its base is 8. Each digit in an octal number represents a specific power of its base (8).**

**As there are only eight digits, three bits (23=8) of binary number system can convert any octal number into binary number.**

**This number system is also used to shorten long binary numbers. The three binary digits can be represented with a single octal digit.**

## **Decimal number system**

**This number system has ten digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) so its base is 10. In this number system, the maximum value of a digit is 9 and the minimum value of a digit is 0.**

**The position of each digit in decimal number represents a specific power of the base (10) of the number system. This number system is widely used in our day to day life. It can represent any numeric value.**

## **Hexadecimal number system**

**This number system has 16 digits that ranges from 0 to 9 and A to F.**

**So, its base is 16. The A to F alphabets represent 10 to 15 decimal numbers.**

**The position of each digit in a hexadecimal number represents a specific power of base (16) of the number system.**

**As there are only sixteen digits, four bits (24=16) of binary number system can convert any hexadecimal number into binary number.**

**It is also known as alphanumeric number system as it uses both numeric digits and alphabets.**

# **What is ALU (Arithmetic Logic Unit)?**

**In the computer system, ALU is a main component of the central processing unit, which stands for arithmetic logic unit and performs arithmetic and logic operations.**

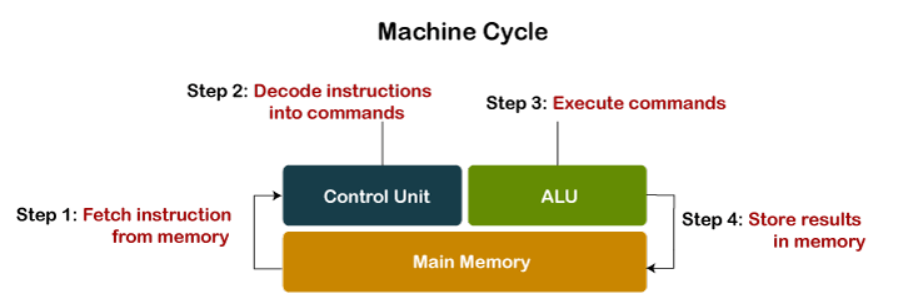
**It is also known as an integer unit (IU) that is an integrated circuit within a CPU or GPU, which is the last component to perform calculations in the processor.**

**It has the ability to perform all processes related to arithmetic and logic operations such as addition, subtraction, and shifting operations, including Boolean comparisons (XOR, OR, AND, and NOT operations).**

**Also, binary numbers can accomplish mathematical and bitwise operations. The arithmetic logic unit is split into AU (arithmetic unit) and LU (logic unit).**

**The operands and code used by the ALU tell it which operations have to perform according to input data.**

**When the ALU completes the processing of input, the information is sent to the computer's memory.**



**Except performing calculations related to addition and subtraction, ALUs handle the multiplication of two integers as they are designed to execute integer calculations; hence, its result is also an integer.**

**However, division operations commonly may not be performed by ALU as division operations may produce a result in a floating-point number.**

**Instead, the floating-point unit (FPU) usually handles the division operations; other non-integer calculations can also be performed by FPU.**

**Additionally, engineers can design the ALU to perform any type of operation. However, ALU becomes more costly as the operations become more complex because ALU destroys more heat and takes up more space in the CPU.**

**This is the reason to make powerful ALU by engineers, which provides the surety that the CPU is fast and powerful as well.**

**The calculations needed by the CPU are handled by the arithmetic logic unit (ALU); most of the operations among them are logical in nature.**

**If the CPU is made more powerful, which is made on the basis of the ALU is designed.**

**Then it creates more heat and takes more power or energy. Therefore, it must be moderation between how complex and powerful ALU is and not be more costly.**

**This is the main reason the faster CPUs are more costly; hence, they take much power and destroy more heat.**

**Arithmetic and logic operations are the main operations that are performed by the ALU; it also performs bit-shifting operations.**

**Although the ALU is a major component in the processor, the ALU's design and function may be different in the different processors.**

**For case, some ALUs are designed to perform only integer calculations, and some are for floating-point operations. Some processors include a single arithmetic logic unit to perform operations, and others may contain numerous ALUs to complete calculations.**

**The operations performed by ALU are:**

* **Logical Operations: The logical operations consist of NOR, NOT, AND, NAND, OR, XOR, and more.**
* **Bit-Shifting Operations: It is responsible for displacement in the locations of the bits to the by right or left by a certain number of places that are known as a multiplication operation.**
* **Arithmetic Operations: Although it performs multiplication and division, this refers to bit addition and subtraction. But multiplication and division operations are more costly to make. In the place of multiplication, addition can be used as a substitute and subtraction for division.**

## **Arithmetic Logic Unit (ALU) Signals:**

**A variety of input and output electrical connections are contained by the ALU, which led to casting the digital signals between the external electronics and ALU.**

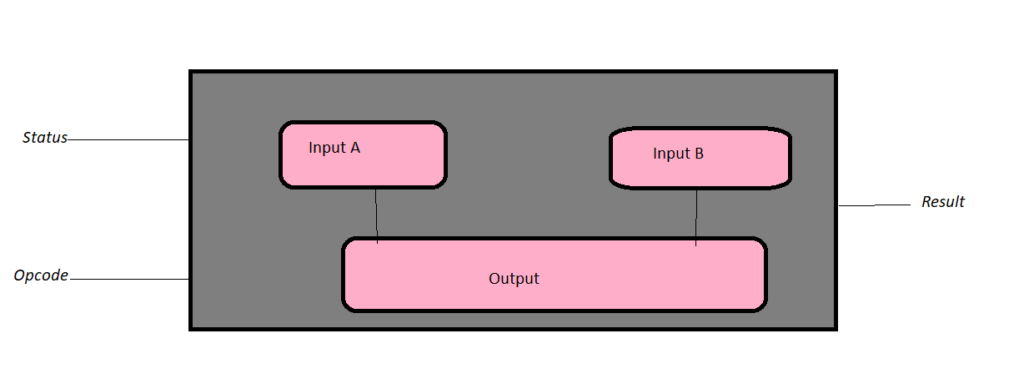
**ALU input gets signals from the external circuits, and in response, external electronics get outputs signals from ALU.**

**Data: Three parallel buses are contained by the ALU, which include two input and output operand. These three buses handle the number of signals, which are the same.**

**Opcode: When the ALU is going to perform the operation, it is described by the operation selection code what type of operation an ALU is going to perform arithmetic or logic operation.**

**Status**

* **Output: The results of the ALU operations are provided by the status outputs in the form of supplemental data as they are multiple signals.**
* **Usually, status signals like overflow, zero, carry out, negative, and more are contained by general ALUs.**
* **When the ALU completes each operation, the external registers contained the status output signals.**
* **These signals are stored in the external registers that led to making them available for future ALU operations.**
* **Input: When ALU once performs the operation, the status inputs allow ALU to access further information to complete the operation successfully. Furthermore, stored carry-out from a previous ALU operation is known as a single "carry-in" bit.**



## **Advantages of ALU**

**ALU has various advantages, which are as follows:**

* **It supports parallel architecture and applications with high performance.**
* **It has the ability to get the desired output simultaneously and combine integer and floating-point variables.**
* **It has the capability of performing instructions on a very large set and has a high range of accuracy.**
* **Two arithmetic operations in the same code like addition and multiplication or addition and subtraction, or any two operands can be combined by the ALU. For case, A+B\*C.**
* **Through the whole program, they remain uniform, and they are spaced in a way that they cannot interrupt part in between.**
* **In general, it is very fast; hence, it provides results quickly.**
* **There are no sensitivity issues and no memory wastage with ALU.**
* **They are less expensive and minimize the logic gate requirements.**

## **Disadvantages of ALU:**

**The disadvantages of ALU are discussed below:**

* **With the ALU, floating variables have more delays, and the designed controller is not easy to understand.**
* **The bugs would occur in our result if memory space were definite.**
* **It is difficult to understand amateurs as their circuit is complex; also, the concept of pipelining is complex to understand.**
* **A proven disadvantage of ALU is that there are irregularities in latencies.**
* **Another demerit is rounding off, which impacts accuracy.**