Computer

Computer can be defined on an electronic device that accept data, process then at a high speed according to a set of instructions provide to it and produces the desired output. So computer is a programmable machine.

Generation of computer:

Generation means variation between different hardware technologies

1st generation: (1946-1956)

First generation computers are made with the use of vacuum tubes. These computers used machine language for programming.

Disadvantages:

1. Occupied lot of space
2. Consumed lot of power
3. Produced lot of heat
4. Costly system

Example: ENIAC(Electronic numeric integrator and computer)
UNIVAC(Universal Accounting company)

2nd generation: (1957-1963)

The computer in which vacuum tubes were replaced by transistors are called second generation of computers these computers used assembly language for programming.

Advantages:

1. Less expensive
2. Consumed less power
3. Produced little heat
4. Less cost and work at higher speed

Example: IBM 7090, IBM 7094

3rd generation: (1964-1981)

The computers using the integrated circuits came to be known on the third generation of computers. These used high level language.

Example: IBM 370, Cyber 175

4th generation: (1982-1989)

The computers which were built with microprocessor are identified on the fourth generation computers these computers use VLSI chips for both cpu and memory.

Example: CRAY-2, IBM 3090

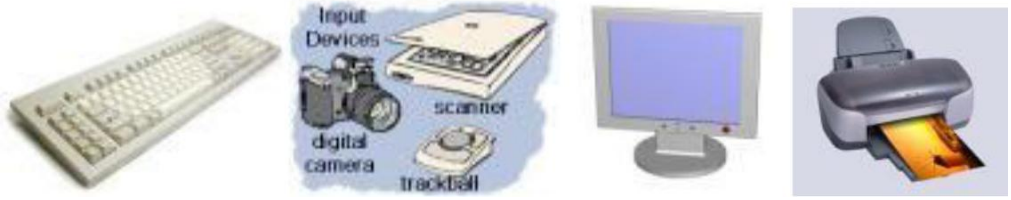
5th generation: (1990- till now)

Fifth generation computers are under development stage these computers use ULSI chips. ULSI chip contains thousands of components into a single IC. Aim of these computers is develop the artificial intelligence.

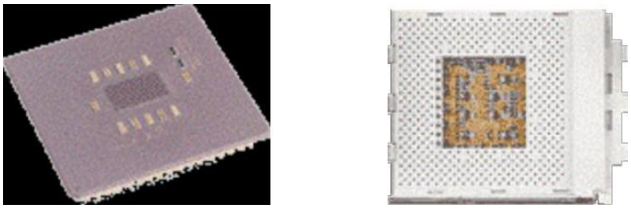
The Computer Hardware

A computer is made up of many parts:

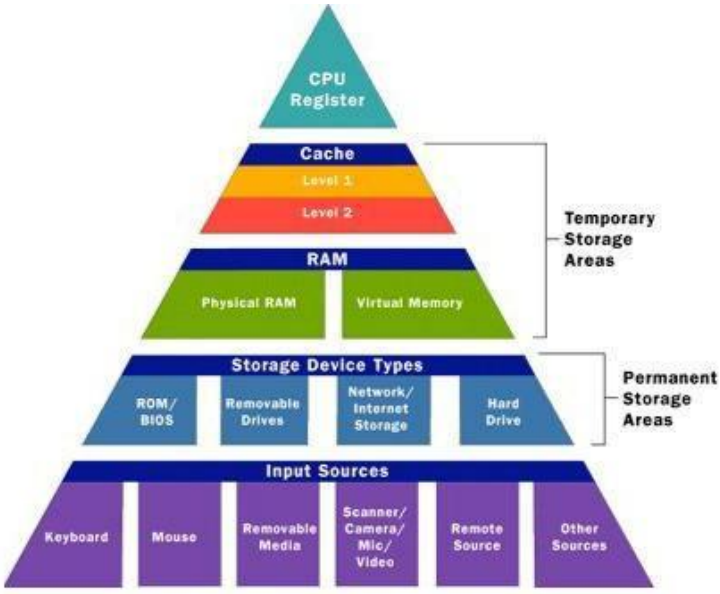
1. Input/Output (I/O) devices – These allow you to send information to the computer or get information from the computer.



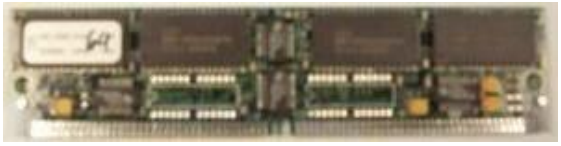
2. Central Processing Unit – CPU or Processor for short. The brain of a computer. Approximately 1.5 in X 1.5 in. Does all the computation/work for the computer.



3. Memory – Although memory is technically any form of electronic storage, it is used most often to identify fast, temporary forms of storage. Accessing the hard drive for information takes time. When the information is kept in memory, the CPU can access it much more quickly.



a. Random Access Memory – RAM. Where information is stored temporarily when a program is run. Information is automatically pulled into memory, we cannot control this. RAM is cleared automatically when the computer is shutdown or rebooted. RAM is volatile (non-permanent).



b. Read Only Memory – ROM. More permanent than RAM. Data stored in these chips is nonvolatile -- it is not lost when power is removed. Data stored in these chips is either unchangeable or requires a special operation to change. The BIOS is stored in the CMOS, read-only memory.

c. Hard Drive – Where you store information permanently most frequently. This is also nonvolatile.

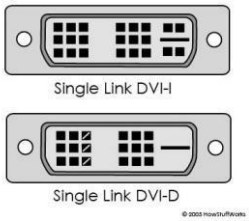
4. Motherboard – A circuit board that allows the CPU to interact with other parts of the computer.

5. Ports – Means of connecting peripheral devices to your computer.

a. Serial Port – Often used to connect a older mice, older external modems, older digital cameras, etc to the computer. The serial port has been replaced by USB in most cases. 9-pin connector. Small and short, often gray in color. Transmits data at 19 Kb/s.

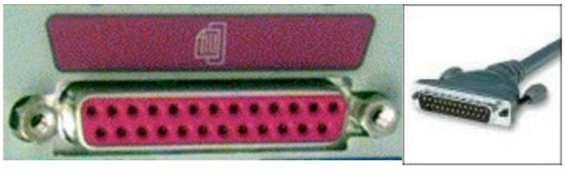
b. Monitor Ports – Used to connect a monitor to the computer. PCs usually use a VGA (Video Graphics Array) analog connector (also known as a D-Sub connector) that has 15 pins in three rows. Typically blue in color.

Because a VGA (analog) connector does not support the use of digital monitors, the Digital Video Interface (DVI) standard was developed.

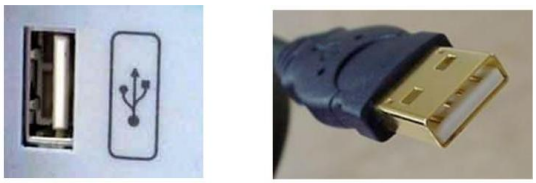


LCD monitors work in a digital mode and support the DVI format. At one time, a digital signal offered better image quality compared to analog technology. However, analog signal processing technology has improved over the years and the difference in quality is now minimal.

c. Parallel Port – Most often used to connect a printer to the computer. 25-pin connector. Long and skinny, often pink in color. Transmits data at 50-100 Kb/s.



d. USB Port – Universal Serial Bus. Now used to connect almost all peripheral devices to the computer. USB 1.1 transmits data at 1.5 Mb/s at low speed, 12 Mb/s at full speed. USB 2.0 transmits data at 480 Mb/s.



e. Firewire/ IEEE 1394 Port – Often found on Apple Computers. Often used with digital camcorders. Firewire transmits data at 400 Mb/s. Firewire 1394B (the new firewire) transmits data at 3.2 Gb/s.

f. PS/2 Port - sometimes called a mouse port, was developed by IBM. It is used to connect a computer mouse or keyboard. Most computers come with two PS/2 ports.

g. Ethernet Port – This port is used for networking and fast internet connections. Data moves through them at speeds of either 10 megabits or 100 megabits or 1 gigabit (1,000 megabits) depending on what speed the network card in the computer supports. Little monitor lights on these devices flicker when in use.

6. Power Supply – Gives your computer power by converting alternating current (AC) supplied by the wall connection to direct current (DC).

7. Expansion Cards – Used to add/improve functionality to the computer.

a. Sound Card – Used to input and output sound under program control. Sound cards provide better sound quality than the built in sound control provided with most computers.

b. Graphics Card – Used to convert the logical representation of an image to a signal that can be used as input for a monitor.

c. Network Card – Used to provide a computer connection over a network. Transmit data at10/100/1000Mb/s.

8. CD ROM – A device used to read CD-ROMs. If capable of writing to the CD-ROM, then these are usually referred to as a "burner" or CD-RW.

9. DVD ROM – A device that is used to read DVDs/CDs. If capable of writing to the DVD, then it is often referred to as a DVD-burner or a DVD-RW.

10. Floppy Drive – A device that is used to read/write to floppy diskettes.

11. Fan – Keeps your computer cool. If the inside of your computer becomes too hot, then the computer can overheat and damage parts.

12. Heatsink – Used to disperse the heat that is produced inside the computer by the CPU and other parts by increasing surface area.

13. The little parts – Capacitors – store energy, Resistors – allows a current through, Transistors – a valve which allows currents to be turned on or off.

14. Case – (Tower if standing upright.) What your motherboard, CPU, etc is contained in.

Computer Types:

Computers can be broadly classified by their speed and computing power.

S.No.	Type	Specifications
1	PC (Personal Computer)	It is a single user computer system having moderately powerful microprocessor
2	Workstation	It is also a single user computer system, similar to personal computer however has a more powerful microprocessor.
3	Mini Computer	It is a multi-user computer system, capable of supporting hundreds of users simultaneously.
4	Main Frame	It is a multi-user computer system, capable of supporting hundreds of users simultaneously. Software technology is different from minicomputer.
5	Supercomputer	It is an extremely fast computer, which can execute hundreds of millions of instructions per second.

Functional Units of Computer:

A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), output and control unit.

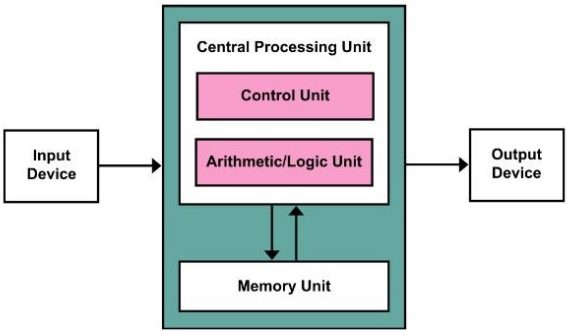


Fig : Functional units of computer

Input device accepts the coded information as source program i.e. high level language. This is either stored in the memory or immediately used by the processor to perform the desired operations. The program stored in the memory determines the processing steps. Basically the computer converts one source program to an object program. i.e. into machine language.

Finally the results are sent to the outside world through output device. All of these actions are coordinated by the control unit.

Input unit:

The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor.

Joysticks, trackballs, mouse, scanners etc are other input devices.

Memory unit:

It is used to store programs and data. It is basically to two types

1. Primary memory
2. Secondary memory

1. Primary memory:

It is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed. The memory contains a large number of semiconductors storage cells. Each ALU Processor Control Unit capable of storing one bit of information. These are processed in a group of fixed site called word.

Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM).

The time required to access one word in called memory access time. Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system.

Caches are the small fast RAM units, which are coupled with the processor and are often contained on the same IC chip to achieve high performance. Although primary storage is essential it tends to be expensive.

2 Secondary memory:

It is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.

Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

Arithmetic logic unit (ALU):

Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register. Then according to the instructions the operation is performed in the required sequence.

The control and the ALU are many times faster than other devices connected to a computer system. This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Control unit:

It effectively is the nerve center that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit.

Output unit:

These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world. **Examples:-** Printer, speakers, monitor etc.

Basic Operational Concepts

The computer works based on some given instruction. Let us consider an example:

Add A, R0

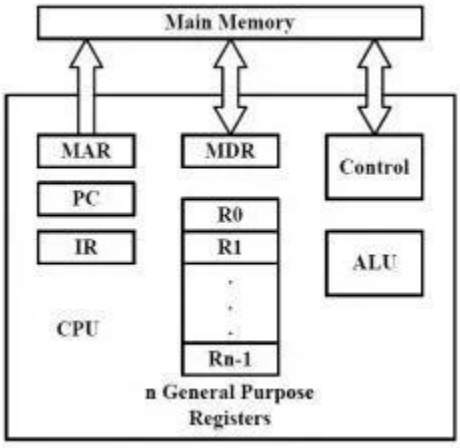
- This instruction adds the operands at memory location A to the operand in the register R0 and places the sum into the register R0.It seems that this instruction is done in one step, but actually it internally performs several steps
- First, the instruction is fetched from the memory into the processor. Next, the operand at A is fetched and added to the contents of R0.
- The above instruction can be written also as-

  Load A, R1
  Add R1, R0

Let us now analyze how the memory and processor are connected:-

The Processor contains a number of registers used for several purposes.

- IR: The IR(Instruction Register) holds the instruction that is currently being executed.
- PC: The PC(Program Counter) is another specialized register which contains the memory address of next instruction to be fetched.
- MAR: This register facilitates communication with the memory. The MAR holds the address of the location to be accessed.
- MDR: This register facilitates communication with the memory. The MDR contains the data to be written into or read out of the addressed location.
- There are n general purpose registers R0 to Rn-1.
- The Program execution starts when the PC is set to point the 1st instruction. The content of the PC is transferred to the MAR and Read control signal is sent to memory. Then the addressed word is read out of the memory and loaded into the MDR. Next the contents of the MDR are transferred to the IR. Then the program is decoded, it is sent to the ALU if it has some arithmetic or logical calculations. The n general purpose registers are used during these calculations to store the result. Then the result is sent to the MDR, and its address of location where result is stored is sent to MAR. And then a write cycle is initiated. Then PC is incremented and the process continues.



Bus Structure:

The simplest and most common way of interconnecting various parts of the computer. To achieve a reasonable speed of operation, a computer must be organized so that all its units can handle one full word of data at a given time. A group of lines that serve as a connecting port for several devices is called a bus.

System bus- This consists of data bus, address bus and control bus

Data bus- A bus which carries data to and from memory/IO is called as data bus

Address bus- This is used to carry the address of data in the memory and its width is equal to the number of bits in the MAR of the memory.

For ex. If computer memory of 64K has 32 bit words then the computer will have a data bus of 32 bits wide and the address bus of 16 bits wide

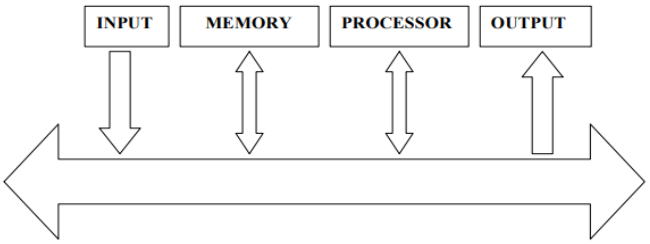
Control Bus- carries the control signals between the various units of the computer.

Ex: Memory Read/write, I/O Read/write

Two types of Bus organizations:

- Single Bus organization
- Two bus Organization

- **Single Bus Architecture**



- Three units share the single bus. At any given point of time, information can be transferred between any two units
- Here I/O units use the same memory address space (Memory mapped I/O)
- So no special instructions are required to address the I/O, it can be accessed like a memory location
- Since all the devices do not operate at the same speed, it is necessary to smooth out the differences in timings among all the devices A common approach used is to include buffer registers with the devices to hold the information during transfers

Ex: Communication between the processor and printer

- **Two Bus Architecture**

Two Bus Organization

- Various units are connected through two independent buses
- I/O units are connected to the processor though an I/O bus and Memory is connected to the processor through the memory bus

- I/O bus consists of address, data and control bus Memory bus also consists of address, data and control bus. In this type of arrangements processor completely supervises the transfer of information to and from I/O units. All the information is first taken to processor and from there to the memory. Such kind of transfers is called as program controlled transfer.

Software:

A set of computer programs are called software. Computer program is a series of instructions telling the computer what to do.

Types of software:

1. System software
2. Application software

1. System software:

- System software exists in the functioning of a computer system and includes the operating system, assembler, interpreter, compiler, linker and loader.
- Operating system is the interface between user applications and system hardware.
- Assembler is a translator that converse assembly language code into machine language.
- Interpreter converts source language program into executable code at once.
- Linker performs the important task of linking together several objects modules.
- The task of loading the linked object modules is performed by the loader.

System software includes three types of programs:

- **Operating System**: The combination of a particular hardware configuration and system software package is known as a computer system platform. System platforms are commonly termed as operating system (OS). Some common operating systems are DOS, UNIX, Mac, and Windows platform.
- **Language Translators**: These are interpreters and compilers for programs such as Pascal, BASIC, COBOL, C, and C++.
- **Common Utility Programs**: Communication tools, disk formatter, etc.

Examples: Language Translator, Operating System, Special Purpose Program, Utilities

2. Application software:

Application software is written to enable the computer to solve a specific data processing task. There are two categories of application software: pre-written software packages and user application programs.

A number of powerful application software packages that do not require significant programming knowledge have been developed. These are easy to learn and use compared to programming languages.

Examples:

- Database Management Software
- Spreadsheet Software, Word processing, Desktop Publishing (DTP), and Presentation Software
- Multimedia Software, Data Communication Software, Statistical and operational research Software

Performance:

- Performance means how quickly it can execute programs.
- For best performance, it is necessary to design the compiler, the machine instruction set, and the hardware in a coordinated way.
- Processor circuits are controlled by a timing signal called clock. The processor divides the action to be performed in basic steps, such that each step can be completed in one clock cycle.
- Basic Performance Equation is given by:- T=((NXS)/R) , Where N= actual no. of instruction executions , S= avg no. of basic step needed to execute one machine instruction , R- clock rate (cycles/sec)
- In order to achieve high performance, the T value should reduce which can be done by reducing N and S, or by increasing R.
- A Substantial improvement can also be done by overlapping the execution of successive instructions. This concept is known as pipelining

Multiprocessor and Multicomputer:

Multicomputer	Multiprocessors
1. A computer made up of several computers. Similar to parallel computing.	1. A multiprocessor system is simply a computer that has more than one CPU on its motherboard.
2. Distributed computing deals with hardware and software systems containing more than one processing element, multiple programs, running under a loosely or tightly controlled regime.	2. Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system.
3. Multicomputer has one physical address space per CPU.	3. Multiprocessors have a single physical address space (memory) shared by all the CPUs
4. It can run faster	4. A multiprocessor would run slower, because it would be in ONE computer.
5. A multi-computer is multiple computers, each of which can have multiple processors. Used for true parallel processing.	5. A multi-processor is a single system with multiple CPU's.

DataTypes:

There are four data types Integer, Real, Character, Logical.

All these types define data values of different kinds

- 128, 4500, 1, 0 (Integer)
- 0.5E19, 28.0, 0.214141E2 (Real)
- hari,maithoo,cs101, fortran (character strings)
- .True., .False. (logical)

Data Representation:

Binary Number System

- The computer can handle with two types of signals, therefore, to represent any information in computer; we have to take help of these two signals.
- These two signals correspond to two levels of electrical signals, and symbolically we represent them as 0 and 1.
- In our day to day activities for arithmetic, we use the Decimal Number System. The decimal number system is said to be of base, or radix 10, because it uses ten digits and the coefficients are multiplied by power of 10.
- A decimal number such as 5273 represents a quantity equal to 5 thousands plus 2 hundreds, plus 7 tens, plus 3 units. The thousands, hundreds, etc. are powers of 10 implied by the position of the coefficients. To be more precise, 5273 should be written as:

  $5x10^3 + 2x10^2 + 7x10^1 + 3x10^0$
- However, the convention is to write only the coefficient and from their position deduce the necessary power of 10.
- In decimal number system, we need 10 different symbols. But in computer we have provision to represent only two symbols. So directly we cannot use decimal number system in computer arithmetic.
- For computer arithmetic we use binary number system. The binary number system uses two symbols to represent the number and these two symbols are 0 and 1.
- The binary number system is said to be of base 2 or radix 2, because it uses two digits and the coefficients are multiplied by power of 2.
- The binary number 110011 represents the quantity equal to:

  $1x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 51$(in decimal)

There are two types of data representation.

1) Fixed point representation
2) Floating point representation

1) Fixed point representation

Representation Unsigned Integers:

Any integer can be stored in computer in binary form. As for example: The binary equivalent of integer 107 is 1101011, so 1101011 are stored to represent 107.

What is the size of Integer that can be stored in a Computer?

It depends on the word size of the Computer. If we are working with 8-bit computer, then we can use only 8 bits to represent the number. The eight bit computer means the storage organization for data is 8 bits.

In case of 8-bit numbers, the minimum number that can be stored in computer is 00000000 (0) and maximum number is 11111111 (255) (if we are working with natural numbers).

So, the domain of number is restricted by the storage capacity of the computer. Also it is related to number system; above range is for natural numbers.

In general, for n-bit number, the range for natural number is from 0 to 2^n-1.

For Example:

104	01101000
49	00110001
153	10011001

In the above example, the result is an 8-bit number, as it can be stored in the 8-bit computer, so we get the correct results.

129	10000001
178	10101010
307	**1**00101011

In the above example, the result is a 9-bit number, but we can store only 8 bits, and the most significant bit (msb) cannot be stored.

The result of this addition will be stored as (00101011) which is 43 and it is not the desired result. Since we cannot store the complete result of an operation, and it is known as the overflow case.

Representation of Signed Integer:

- 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 2^n different combination, and we use these different combinations to represent 2^n 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 representation, the range is from -2^{n-1}-1 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.

There are three different schemes to represent negative number:

a) Signed-Magnitude form
b) 1's complement form
c) 2's complement form

a) 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 and 11011001 represents -169

b) 1's complement form

- Consider the eight bit number 01011100, 1's complements of this number is 10100011. If we perform the following addition:

```
  0 1 0 1 1 1 0 0
  1 0 1 0 0 0 1 1
-----------------
  1 1 1 1 1 1 1 1
```

- If we add 1 to the number, the result is 100000000.
- Since we are considering an eight bit number, so the 9th bit (MSB) of the result cannot be stored. Therefore, the final result is 00000000.
- Since the addition of two numbers is 0, so one can be treated as the negative of the other number. So, 1's complement can be used to represent negative number.

c) 2's complement form

- Consider the eight bit number 01011100, 2's complements of this number is 10100100. If we perform the following addition:

```
  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 9th bit (MSB) of the result cannot be stored. Therefore, the final result is 00000000.
- Since the addition of two numbers 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.

Decimal	2's Complement	1's complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	-----	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	------	-------

b) Floating point Representation:

To convert the floating point into decimal, we have 3 elements in a 32-bit floating point representation:

i) Sign
ii) Exponent
iii) Mantissa

Sign bit is the first bit of the binary representation. '1' implies negative number and '0' implies positive number.

Example: 11000001110100000000000000000000 This is negative number.

Exponent is decided by the next 8 bits of binary representation. 127 is the unique number for 32 bit floating point representation. It is known as bias. It is determined by 2^{k-1} -1 where 'k' is the number of bits in exponent field.

There are 2 exponent bits in 8-bit representation and 8 exponent bits in 32-bit representation. Thus

bias = 3 for 8 bit conversion (2^{2-1} -1 = 4-1 = 3)

bias = 127 for 32 bit conversion. (2^{8-1} -1 = 128-1 = 127)

Example: 01000001110100000000000000000000

10000011 = $(131)_2$

131-127 = 4

Hence the exponent of 2 will be 4 i.e. 2^4 = 16.

Mantissa is calculated from the remaining 24 bits of the binary representation. It consists of '1' and a fractional part which is determined by:

Example: 01000001110100000000000000000000

The fractional part of mantissa is given by:

1*(1/2) + 0*(1/4) + 1*(1/8) + 0*(1/16) +…… = 0.625

Thus the mantissa will be 1 + 0.625 = 1.625

The decimal number hence given as: Sign*Exponent*Mantissa = (-1)*(16)*(1.625) = -26

To convert the decimal into floating point, we have 3 elements in a 32-bit floating point representation:

i) Sign (MSB)
ii) Exponent (8 bits after MSB)
iii) Mantissa (Remaining 23 bits)

Sign bit is the first bit of the binary representation. '1' implies negative number and '0' implies positive number.

Example: To convert -17 into 32-bit floating point representation Sign bit = 1

Exponent is decided by the nearest smaller or equal to 2^n number. For 17, 16 is the nearest 2^n. Hence the exponent of 2 will be 4 since 2^4 = 16. 127 is the unique number for 32 bit floating point representation. It is known as bias. It is determined by 2^{k-1} -1 where 'k' is the number of bits in exponent field.

Thus bias = 127 for 32 bit. (2^{8-1} -1 = 128-1 = 127)

Now, 127 + 4 = 131 i.e. 10000011 in binary representation.

Mantissa: 17 in binary = 10001.

Move the binary point so that there is only one bit from the left. Adjust the exponent of 2 so that the value does not change. This is normalizing the number. 1.0001 x 2^4. Now, consider the fractional part and represented as 23 bits by adding zeros.

00010000000000000000000

Thus the floating point representation of -17 is 1 10000011 00010000000000000000000

Error Detection Codes:

What is Error?

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error-Detecting codes:

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is **parity check**.

There are two ways to error detection.

a. Parity bit
b. Hamming Code

a. Parity bit :

A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. **Even parity** -- For a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 0.
2. **Odd parity** -- For a given set of bits, the number of 1's are counted. If that count is odd, the parity bit value is set to 1.

P	Data Bits
0	1011010

P	Data Bits
1	1010010

b. Hamming Code:

In Hamming code, the redundancy bits are placed at certain calculated positions in order to eliminate errors. The distance between the two redundancy bits is called "Hamming distance".

The number of parity bits to be added to a data string depends upon the number of information bits of the data string which is to be transmitted. Number of parity bits will be calculated by using the data bits. This relation is given below.

$$2^P >= n + P + 1$$

General Algorithm of Hamming code –

The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.
   a. Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
   b. Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).
   c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
   d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
   e. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.
5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Example:

7	6	5	4	3	2	1
D7	D6	D5	P4	D3	P2	P1

D7	D6	D5	P4	D3	P2	P1
1	**1**	**0**	**P4**	**1**	**P2**	**P1**

For the above word the parity bits are calculated as….,

P1	D3	D5	D7
P1	1	0	1
0	1	0	1

P2	D3	D6	D7
P2	1	1	1
1	1	1	1

P4	D5	D6	D7
P4	0	1	1
0	0	1	1

The final word is

D7	D6	D5	P4	D3	P2	P1
1	**1**	**0**	**0**	**1**	**1**	**0**

Computer Arithmetic:

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.

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 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 below Table. 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.

- When the signs of A and B are identical (different), add the two magnitudes and attach the sign of A to the result.
- When the signs 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.

Operation	Add Magnitude	Subtract Magnitude		
		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)

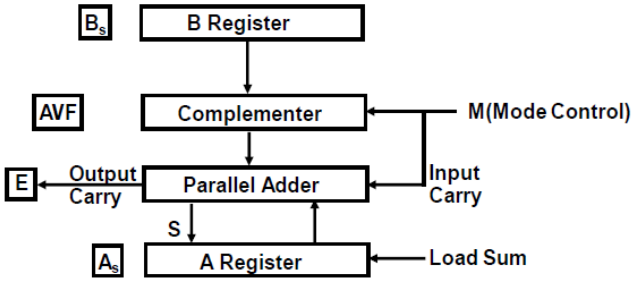
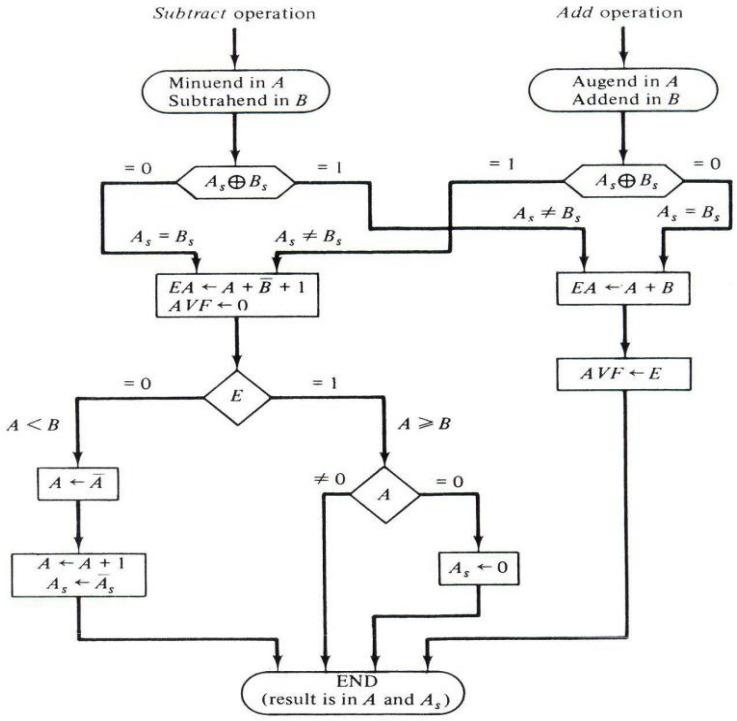


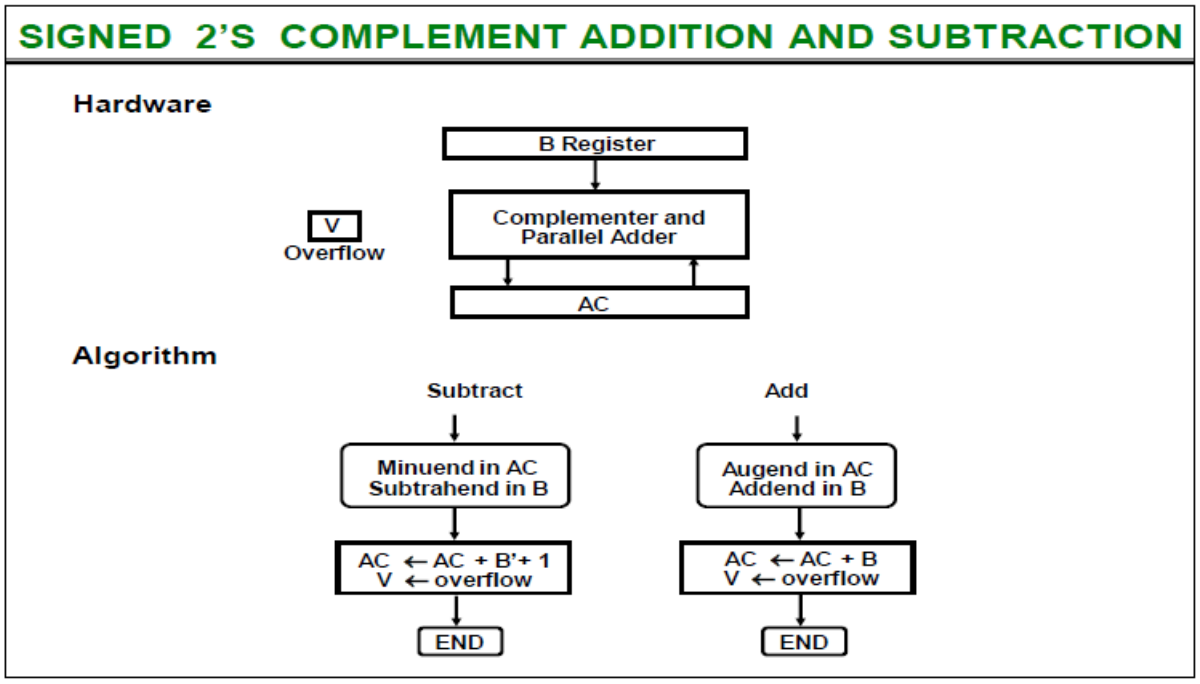
Fig: Hardware Implementation



Flowchart for add and subtract operations.

Addition and subtraction algorithm:

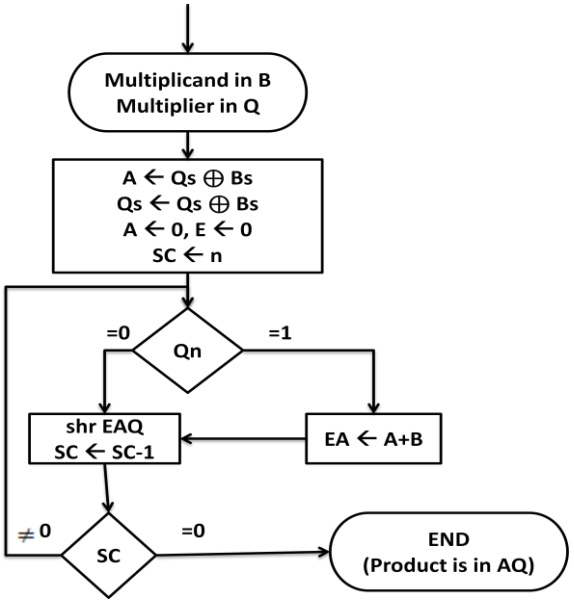
- The flowchart is shown in above Figure. The two signs A, and B, are compared by an exclusive-OR gate.
- If the output of the gate is 0 the signs are identical; If it is 1, the signs are different.
- For an add operation, identical signs dictate that the magnitudes be added. For a subtract operation, different signs dictate that the magnitudes be added.
- The magnitudes are added with a micro operation EA← 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 >= 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 micro operation A A' +1.
- However, we assume that the A register has circuits for micro operations complement and increment, so the 2's complement is obtained from these two micro operations.
- 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.



Multiplication with signed magnitude Data:

In the beginning, the multiplicand is in B and the multiplier in Q. Their corresponding signs are in Bs and Qs respectively. We compare the signs of both A and Q and set to corresponding sign of the product since a double-length product will be stored in registers A and Q. Registers A and E are cleared and the sequence counter SC is set to the number of bits of the multiplier. Since an operand must be stored with its sign, one bit of the word will be occupied by the sign and the magnitude will consist of n-1 bits.

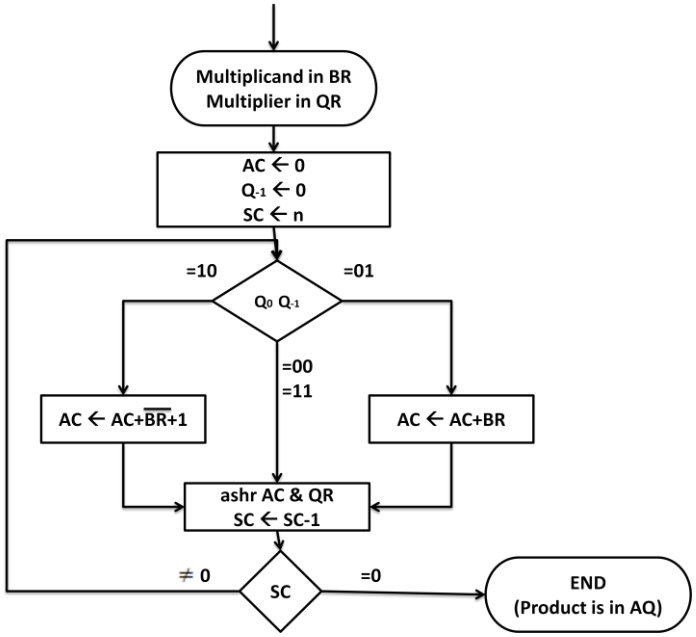
Now, the low order bit of the multiplier in Qn is tested. If it is 1, the multiplicand (B) is added to present partial product (A), 0 otherwise. Register EAQ is then shifted once to the right to form the new partial product. The sequence counter is decremented by 1 and its new value checked. If it is not equal to zero, the process is repeated and a new partial product is formed. When SC = 0 we stops the process.



Multiplicand B=10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Qn=1; add B		10111		
First Partial Product	0	10111		
Shift Right EAQ	0	01011	11001	100
Qn=1; add B		10111		
Second Partial Product	1	00010		
Shift Right EAQ	0	10001	01100	011
Qn=0; Shift right EAQ	0	01000	10110	010
Qn=0; Shift right EAQ	0	00100	01011	001
Qn=1; add B		10111		
Fifth Partial product	0	11011		
Shift Right EAQ	0	01101	10101	000
Final Product in AQ=0110110101				

Booths Multiplication Algorithm:

- Booth algorithm gives a procedure for multiplying binary integers in signed- 2's complement representation.
- It operates on the fact that strings of 0's in the multiplier require no addition but just shifting, and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.
- For example, the binary number 001110 (+14) has a string 1's from 23 to 21 (k=3, m=1). The number can be represented as $2^{k+1}-2^m=2^4 - 2^1 = 16 - 2 = 14$. Therefore, the multiplication M X 14, where M is the multiplicand and 14 the multiplier, can be done as M X 2^4 – M X 2^1.
- Thus the product can be obtained by shifting the binary multiplicand M four times to the left and subtracting M shifted left once.



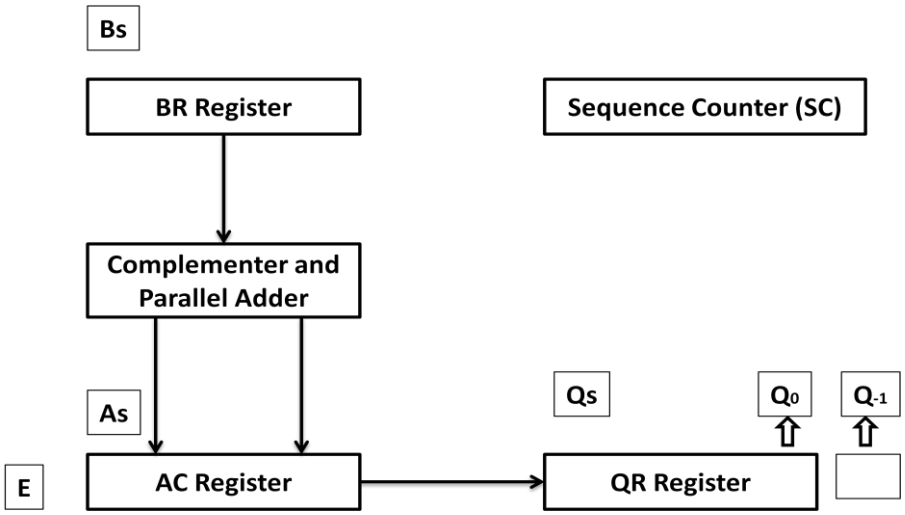
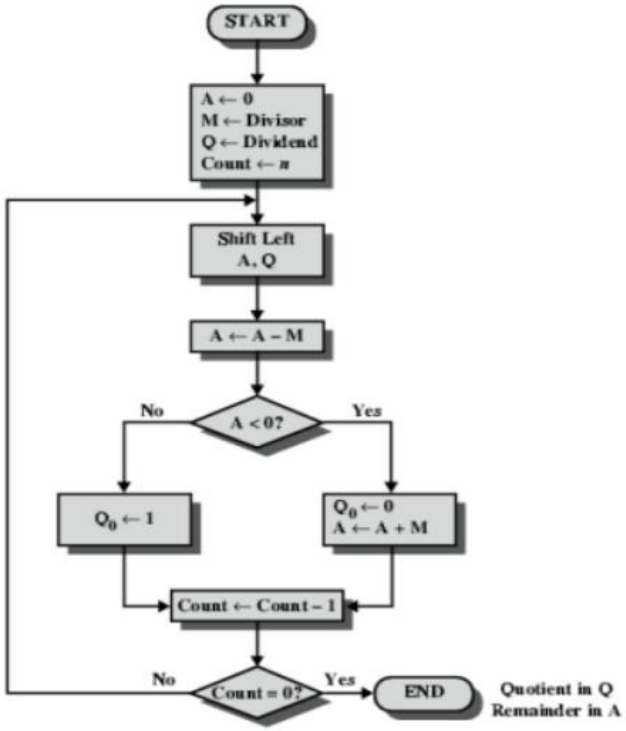


Fig: Hardware Implementation for Booths Multiplication

$Q_0 Q_{-1}$	BR = 10111 $\overline{BR}$+1 = 01001	AC	QR	Q_{-1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000

Division Algorithm:

- **Step 1:** Initialize A, Q and M registers to zero, dividend and divisor respectively and counter to n where n is the number of bits in the dividend.
- **Step 2:** Shift A, Q left one binary position.
- **Step 3:** Subtract M from A placing answer back in A. If sign of A is 1, set Q_0 to zero and add M back to A (restore A). If sign of A is 0, set Q_0 to 1.
- **Step 4:** Decrease counter; if counter > 0, repeat process from step 2 else stop the process. The final remainder will be in A and quotient will be in Q.



M=2=0010

$\bar{M}$+1 =(-M)= 1110

	A	Q	SC	
	0000	0111 (7)	100	
Shift Left AQ	0000	1110		
A←A-M	1110			
	1110			
A<0 A←A+M	0010			
Set Q_0←0	0000	1110		
Count←Count-1	0000	1110	011	
Shift Left AQ	0001	1100		
A←A-M	1110			
	1111			
A<0 A←A+M	0010			
Set Q_0←0	0001	1100		
Count←Count-1	0001	1100	010	
Shift Left AQ	0011	1000		
A←A-M	1110			
	0001			
A>0 Set Q_0←1	0001	1001		
Count←Count-1	0001	1001	001	
Shift Left AQ	0011	0010		
A←A-M	1110			
	0001			
A>0 Set Q0←1	0001	0011	000	Reminder(A)=1, Quotient(Q)=2

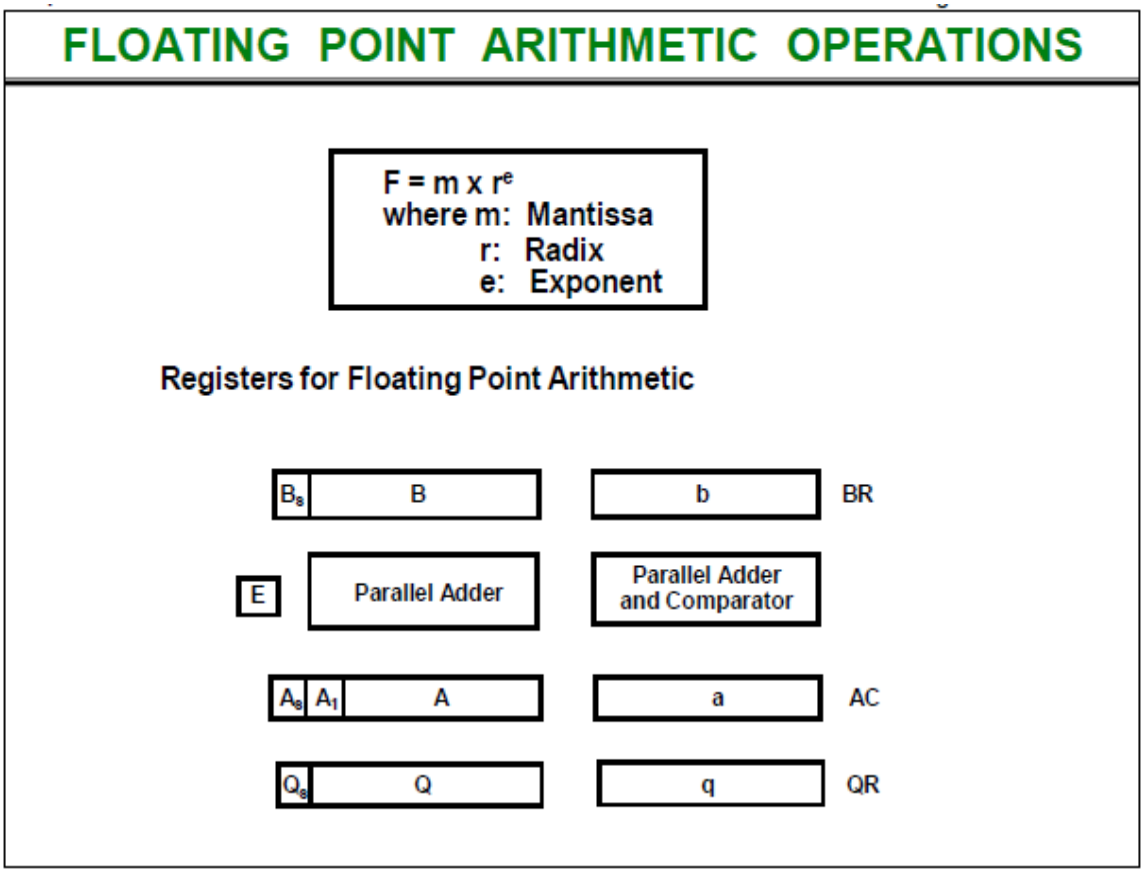
Floating point algorithms:

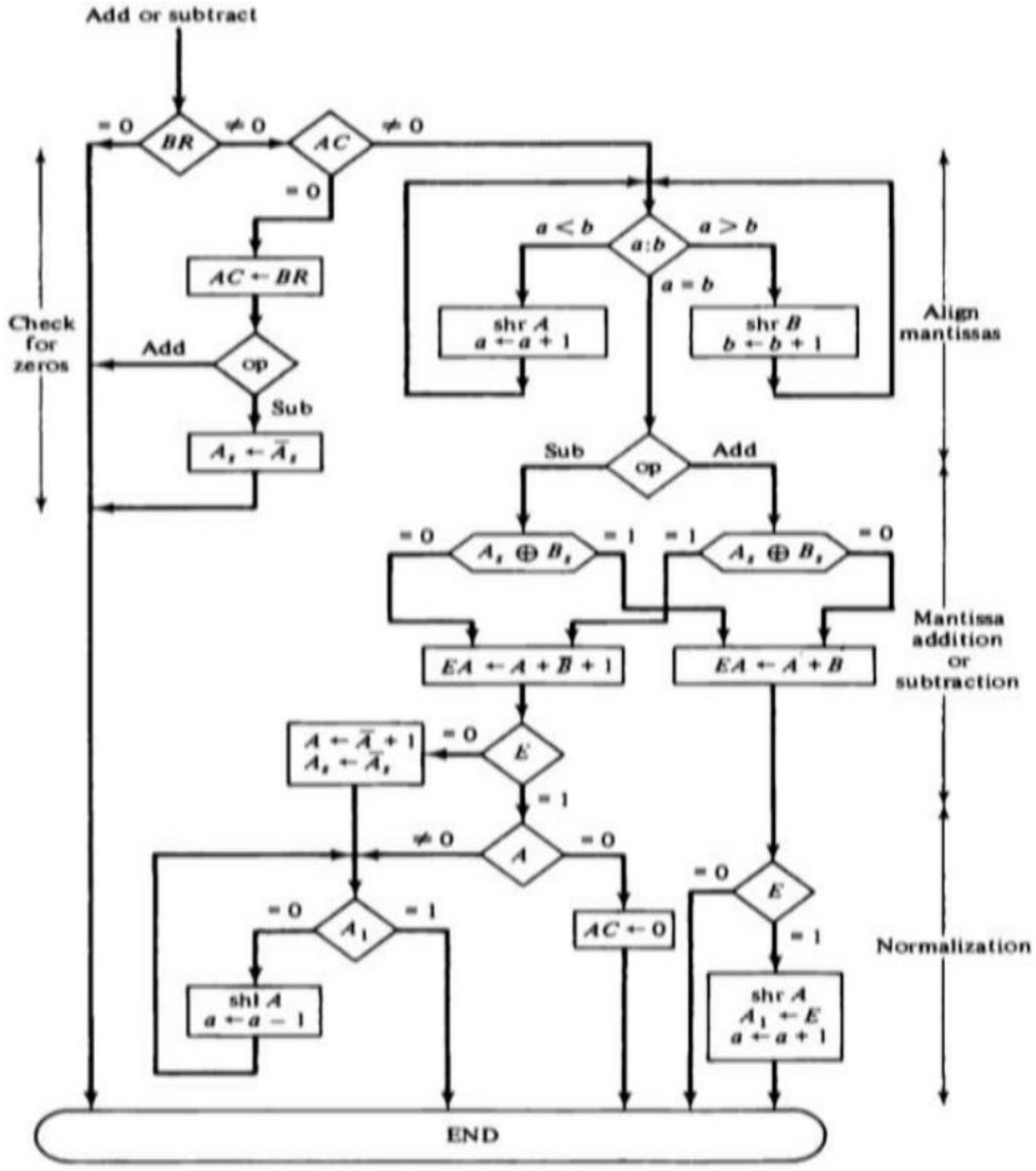
During addition or subtraction, the two floating-point operands are kept in AC and BR. The sum or difference is formed in the AC. The algorithm can be divided into four consecutive parts:

1. Check for zeros.
2. Align the mantissas.
3. Add or subtract the mantissas
4. Normalize the result

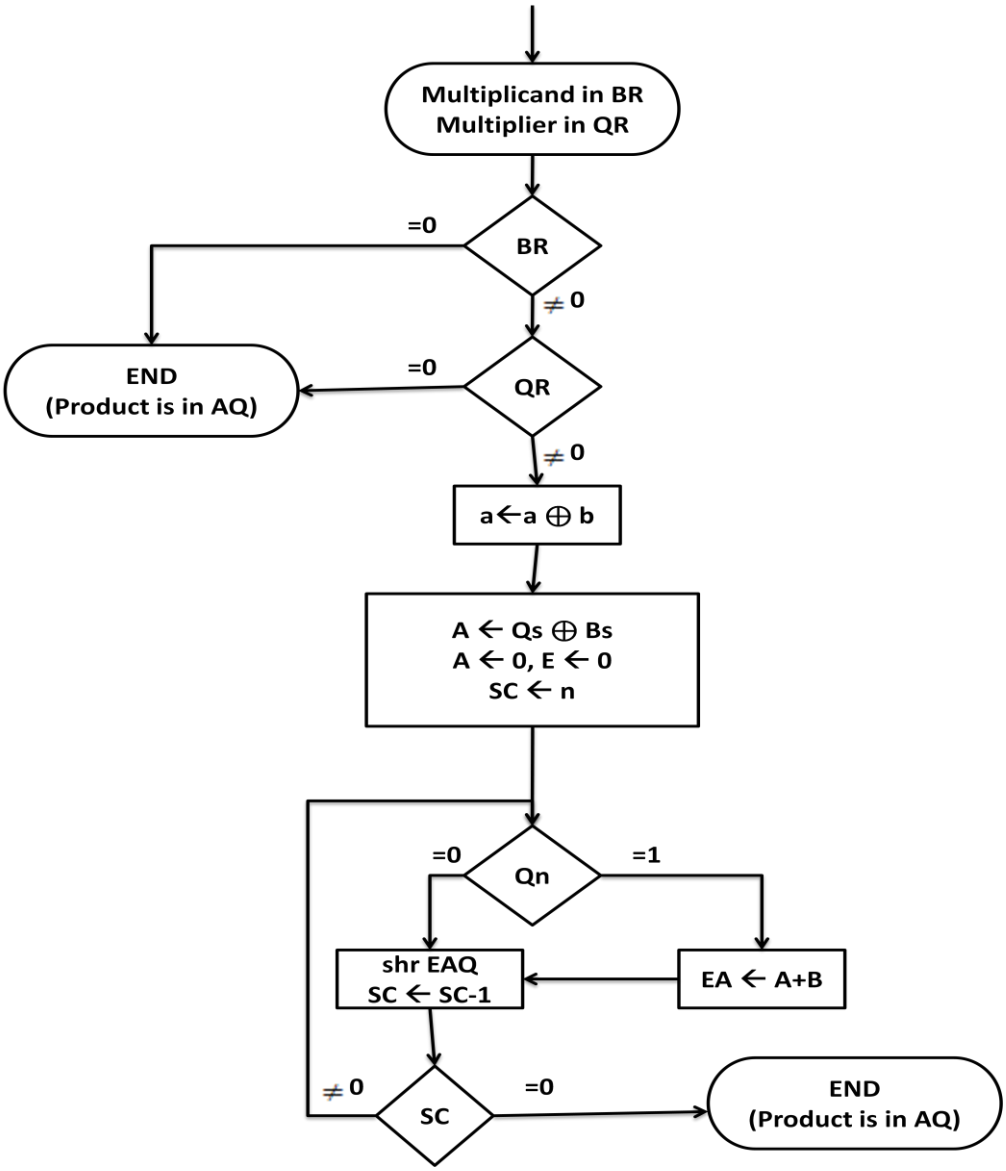
A floating-point number cannot be normalized, if it is 0. If this number is used for computation, the result may also be zero. Instead of checking for zeros during the normalization process we check for zeros at the beginning and terminate the process if necessary. The alignment of the mantissas must be carried out prior to their operation. After the mantissas are added or subtracted, the result may be un-normalized. The normalization procedure ensures that the result is normalized before it is transferred to memory.

If the magnitudes were subtracted, there may be zero or may have an underflow in the result. If the mantissa is equal to zero the entire floating-point number in the AC is cleared to zero. Otherwise, the mantissa must have at least one bit that is equal to 1. The mantissa has an underflow if the most significant bit in position A1, is 0. In that case, the mantissa is shifted left and the exponent decremented. The bit in A1 is checked again and the process is repeated until A1 = 1. When A1 = 1, the mantissa is normalized and the operation is completed.

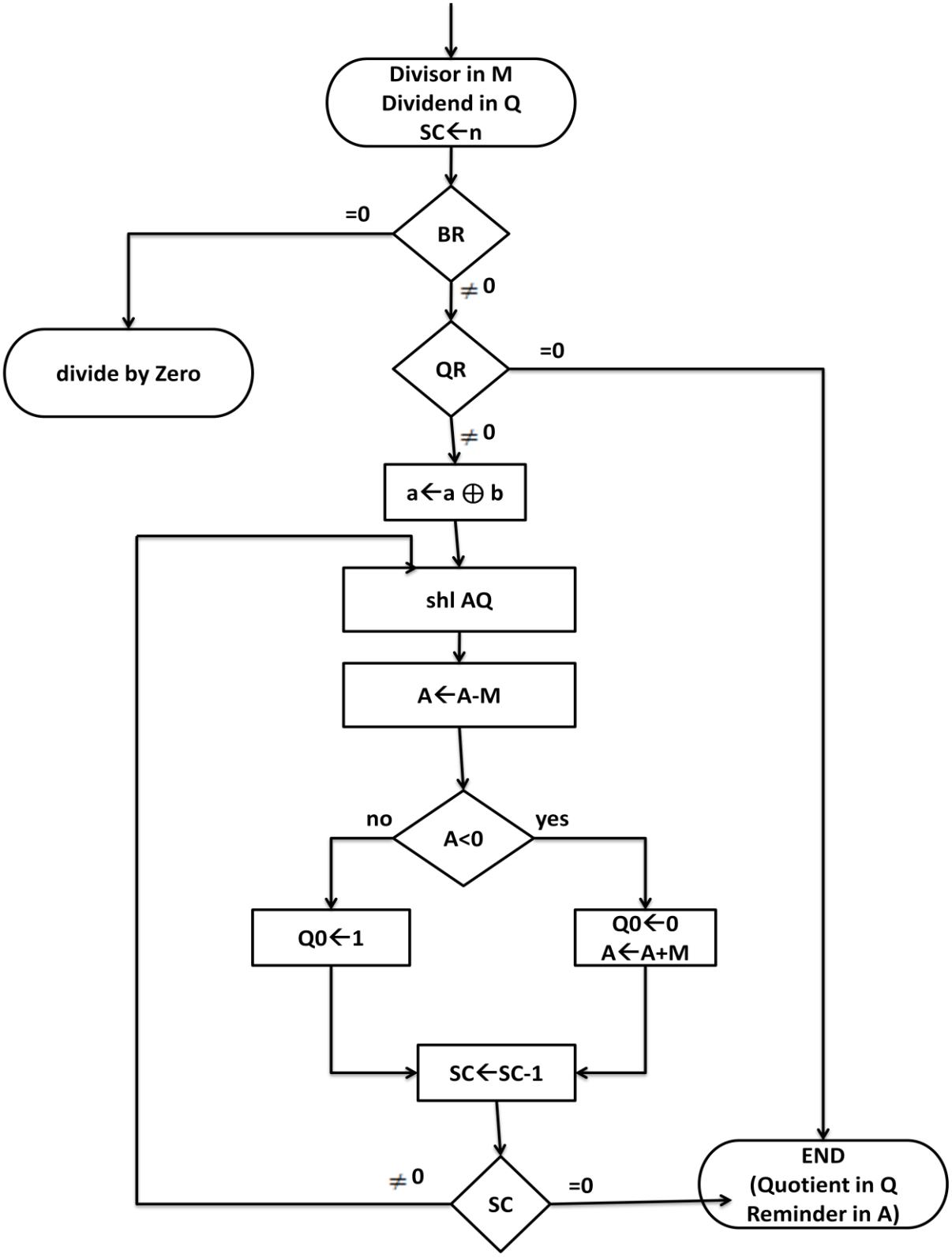




Addition and subtraction of floating-point numbers.



Floating Point Multiplication



Floating Point Division