② Functional Units:

A computer consists of five functionally independent main Parts called as Functional Units.

These are as follows.

i) Input unit
ii) memory unit
iii) Arithmetic and logic unit
iv) output unit and
v) Control unit.

* The input unit accepts coded information from human operators from electromechanical devices such as keyboards, or from other Computers over digital communication lines.

* The information received is either stored in the computer's memory for later reference or immediately used by the arithmetic and logic circuit to perform the desired operations.

* The processing steps are determined by a program stored in the memory.

* Finally, the results are sent back to the outside world through the output unit.

* All of these actions are coordinated and maintained by the control unit.

* The following figure shows the Basic functional units of a computer.

* It does not show any connections between functional units, because these connections can be made in several ways.

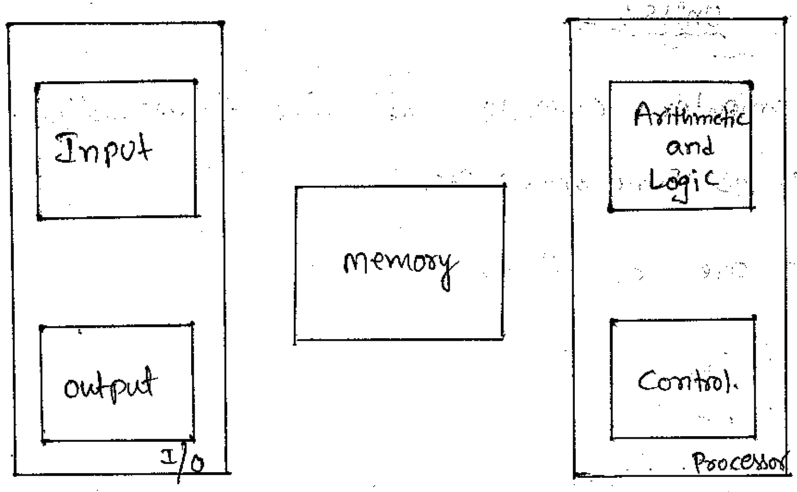


fig: Basic functional units of a computer.

I Illustration of functional units:

i) Input unit:

* Computer accepts coded information through input units, which read the data.

* The most well-known input device is a keyboard.

* whenever a key is pressed, the corresponding letter or digit is automatically translated over a cable to either the memory or the processor.

* Various other Input units include,

    ⓐ Joysticks
    ⓑ Track balls
    ⓒ mouses
    ⓓ microphones (used to capture audio input).

ii) Memory unit:

* The function of the memory unit is to store programs and data.

There are two classes of storage, called

ⓐ primary and
ⓑ secondary.

ⓐ Primary Storage:

* Primary storage is a fast memory that operates at electronic speeds.
* The memory contains a large number of semiconductor storage cells, each capable of storing one bit of information.
* These cells are read and written in groups of fixed size called words.
* To provide easy access to any word in the memory, a distinct address is associated with each word location.
* Address are numbers that identify succesive locations.
* A given word is Accessed by specifying its address and issuing a control command that starts the storage or retrieval process.
* The Number of bits in each word is reffered as word length of the computer.
* Typical word lengths range from 16 to 64 bits.
* The capacity of the memory is one factor that characterizes the size of the computer.
* 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 is called the memory Access time.

ⓑ Secondaly Storage:

* AS primary is expensive, cheaper secondaly storage

is used when large amounts of data and many programs have to be stored, particularly for information that is accessed infrequently.

* we have various secondary storage devices available, which includes magnetic disks, tapes and optical disks (CD-Rom's).

iii) Arithmetic and logic unit:

* most computer operations are executed in the Arithmetic and logic unit (ALU) of the processor.

* for Example,

Two numbers located in the memory are to be added, they are brought into the processor and the actual addition is carried out by the ALU.

* The sum may then be stored in memory or retained in the processor for immediate use.

* Any other Arithmetic and logic operations like, multiplication, division or comparision of numbers is initiated by bringing the required operands in to the processor, where the operation is performed by the ALU.

* when operands are brought in to the processor, they are stored in the highspeed storage elements called "Registers".

* The control and ALU are manytimes faster than other devices connected to a computer system.

This enables a single processor to control a number of External devices such as keyboards, displays, magnetic and optical disks.

iv) Output unit:

* The outputs unit's function is to send processed results to the outside world.

* most familiar example for output unit is a printer.

* It is possible to produce printers capable of printing as many as 10,000 lines per minute.

* Graphic displays provides both output function and input functions, thats why we call as I/o unit in some cases.

v) Control unit:

* The operations performed by memory, ALU, Input and output units must be coordinated in some way, this is the task of the control unit.

* The timing signals which transfers the instructions and data are generated by the control signals.

* Timing signals are signals that determine, when a given action is to take place.

* Data transfers between the processor and the memory are also controlled by the control unit through timing signals.

→ The operation of a computer can be summarized as follows:

① The computer accepts information in the form of Programs and data through an input unit and stores it in the memory.

④ Bus Structure:

* To from an operational system, all the functional units of a computer must be connected in some organized way.

* This can be done in many ways. let's consider the simplest among them.

* A group of lines that serves as a connecting path for several devices is called a "bus".

* 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.

* When a word of data is transferred between units, all its bits are transferred in parallel. i.e., the bits are transferred simultaneously over many wires or lines as one bit per line.

* The simplest way to interconnect all the functional units is to use a single bus as shown below.

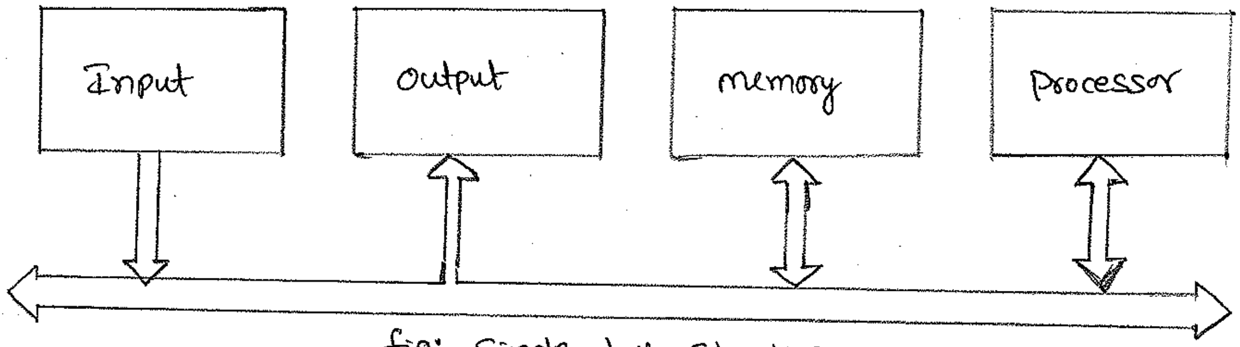


fig: Single-bus structure.

* All units are connected to the BUS.

* The main advantage of the single-bus structure is its low cost and its flexibility for attaching peripheral devices.

* Systems that contain multiple buses achieve more operations at the same time. This leads to better performance but at an increased cost.

* The devices connected to a Bus differs widely in their speed of operations.

* Keyboards and printers are relatively slow when compared to magnetic or optical disks.

* An Efficient transfer mechanism is needed to reduce timing difference among processors, memories and External devices.

* A common approach is to include buffer registers with the devices to hold the information during transfers.

* for Example, Consider the transfer of an encoded character from a processor to a character printer.

* The processor Sends the character over the bus to the Printer buffer.

* Since the buffer is an electronic register, this transfer requires relatively little time.

* once the buffer is loaded, the printer can Start printing, without further action by the processor.

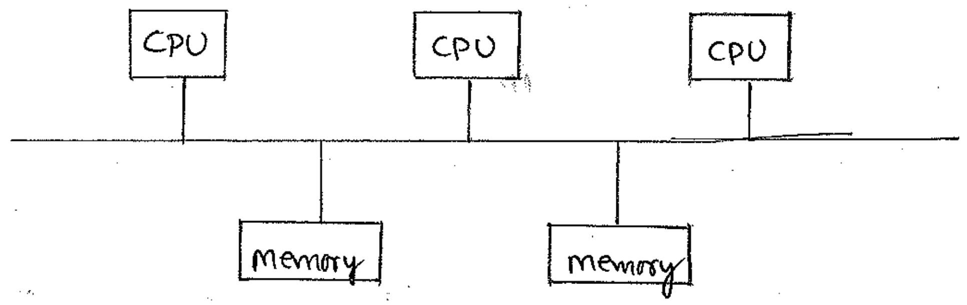
* The bus and the processor are no longer needed and can be released for other activity.

* The printer continues printing the character in its buffer and is not available for further transfers until this process is completed.

* Thus buffer registers Smooth out time differences among processors, memories and I/o devices.

Multiprocessors and Multicomputers:

* Large computer systems may contain more number of processor units, called as multiprocessor systems.
* The Block diagram for Multiprocessors is as shown below.



* These systems either execute a number of different application tasks in parallel, or they execute sub tasks of a single large task in parallel.

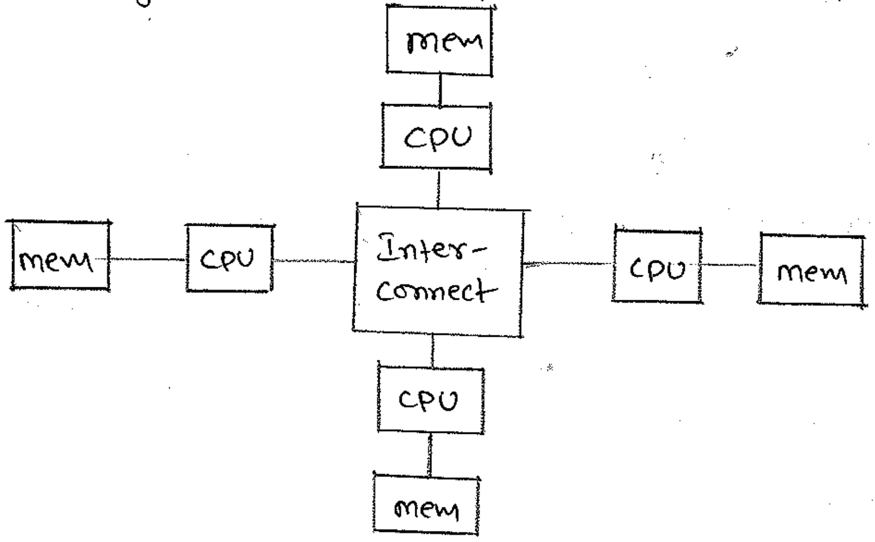
* In such systems, all processors usually have access to all of the memory units. and thus these systems are called as "Shared memory multiprocessor systems".

* Use of such systems gives high performance but with much increased complexity and cost.

* In contrast to multiprocessor systems, it is also possible to use an interconnected group of complete computers to achieve high total computational power, which are known

as multicomputers.

* The Block diagram for multicomputers is as shown below.



* This computers normally have access only to their own memory units.

* TO communicate data, messages are exchanged over a communication network.

* Thus these systems are known as "message - passing multicomputers".

⑧ Data Representation:

* In ordinary arithmetic, a negative number is indicated by a minus sign and a positive number by a plus sign.

* Because of the hardware limitations, computer must represent everything with 1's and 0's, including the sign of a number.

* So, to represent the sign, a bit placed in the left most position of the number.

* Generally the sign bit is ~~[illegible]~~ equal to '0' for positive numbers and '1' for negative numbers.

* In addition to the sign, a number may have a binary (or decimal) point.

* The position of the binary point is needed to represent fractions, integers or mixed-integer fraction numbers.

* There are two ways of specifying the position of the binary point in a register.

    i) Fixed-Point Representation. and.
    ii) Floating-Point Representation.

Fixed-point Representation:

The fixed point method assumes that the binary point is always fixed in one position.

The two positions most widely used are,

i) A binary point in the Extreme left of the registers to make the stored number a fraction. and

ii) A binary point in the extreme right of the register to make the stored number an integer.

Integer Representation:

* when an integer binary number is positive the sign is represented by '0' and the magnitude by a positive binary number.

* when the number is negative, the sign is represented by '1'. but the rest of the number may be represented in one of three possible ways.

i) Signed-magnitude representation.
ii) Signed - 1's Complement representation. and.
iii) signed - 2's Complement representation

i) signed - magnitude representation :-

* In Signed magnitude representation, the Signed bit is inserted at the most Significant position. i.e, the '1' is inserted for negative numbers and '0' is inserted for the positive numbers at the most significant position.

*It Contains the magnitude of its normal binary value.

Example:

$+14 \quad \boxed{0}0001110$

$-14 \quad \underset{\substack{\downarrow \\ \text{Sign} \\ \text{bit}}}{\boxed{1}}0001110$

ii) Signed 1's complement Representation:

* In this Signed 1's complement representation, the negative number is represented in 1's complement of its Positive Value. i.e, the most significant bit represents a negative number by '1'. and the magnitude of the number is written in 1's complement form.

* Consider the signed number '−14' stored in 8-bit register.

* In signed 1's Complement representation '−14' ~~[illegible]~~ is written as follows.

Ex:

$+14 : \underline{0}0001110$

$-14 : \underbrace{1}_{\text{Sign bit.}}\underbrace{1110001}_{\text{1's complement of '+14'}}$

ii) Signed 2's Complement Representation:

* In this 2's Complement representation, the negative numbers are represented in such a way that, its magnitude is represented by 2's complement of its positive value and the first bit represents the negative sign by '1'.

* Consider the signed number '−14' stored in an 8-bit register.

* In Signed 2's Complement representation '−14' is written as follows.

Ex: $-14 : \underbrace{1}_{\text{Sign bit.}}\underbrace{1110010}_{\text{2's complement of '+14' (00001110)}}$

i.e, The only way to represent +14 is, 00001110.

* But a negative number (-14 in our example) can be represented in three different forms. i.e,

In Signed-magnitude representation, Signed 1's complement representation and signed 2's complement representation.

Arithmetic addition

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

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

* If the signs are different, we subtract the smaller magnitude from the larger and give the result, the sign of the larger magnitude.

* for Example,

$$(+25) + (-37)$$

$$\Rightarrow -(37-25)$$

$$\Rightarrow -12.$$

* This process requires comparision of the Signs and the magnitudes and then performing either addition or subtraction.

* The rule for adding numbers in the signed 2's Complement System does not require a comparision or Substraction. but only requires addition and complementation.

* The procedure is very simple, which is, add the two

numbers, including their sign bits and discard any carryout of the sign (left most) bit position.

* Numerical Examples for addition are shown below.

+6	00000110	−6	11111010
+13	00001101	+13	00001101
+19	00010011	+7	00000111

+6	00000110	−6	11111010
−13	11110011	−13	11110011
−7	11111001	−19	11101101

* Note that negative numbers must initially be in 2's Complement and if the sum obtained after the addition is negative, it is also in 2's complement form.

* In each of the four cases, the operation performed is always addition, including the sign bits.

* Any carry out of the sign bit position is discarded, and negative results are automatically in 2's complement form.

<u>Arithmetic Subtraction</u>:

* Subtraction of two signed binary numbers when negative numbers are in 2's complement form is very simple.

* The procedure is, take the 2's complement of the subtrahend and add it to the minuend.

* A carry out of the sign bit position is discarded.

* This procedure shows that a subtraction operation can be changed to an addition operation if the sign of the subtrahend is changed.

* This is demonstrated by the following relationship

$$\boxed{\begin{aligned}(\pm A) - (+B) &= (\pm A) + (-B) \\ (\pm A) - (-B) &= (\pm A) + (+B)\end{aligned}}$$

Overflow:

* when two numbers of 'n' digits each are added and the sum occupies 'n+1' digits, then that case is said to be an overflow occured.

* An overflow is a problem in digital computers. because the width of register is finite.

* A result that contains 'n+1' bits cannot be accomodated in a register with a standard length of 'n' bits.

* For this reason, many computers detect the occurrence of an overflow, and when it occurs, a corresponding flip-flop is set which can be checked by the user.

* The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed or unsigned.

* when two unsigned numbers are added, an overflow is

detected from the end carry out of the most significant position.

* When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.

* An overflow cannot occur after an addition, if one number is positive and the other is negative.

Decimal fixed point Representation:

* The representation of decimal numbers in registers is a function of the binary code used to represent a decimal digit.

* A 4-bit decimal code requires four flip-flops for each decimal digit.

* The Representation of 4385 in BCD requires 16 flip-flops, four flip-flops for each digit.

* The number will be represented in a register with 16 flip-flops as follows.

0100 0011 1000 0101

* By representing numbers in decimal we are wasting a considerable amount of storage space, since the no. of bits needed to store a decimal number in a binary code is greaterthan the number of bits needed for its equivalent binary representation.

* And also the circuits required to perform decimal arithmetic are more complex.

*. However, there are some advantages in the use of decimal

representation because computer input and output data are generated by people who use the decimal System.

* Some applications, Such as business data processing, require Small amounts of arithmetic computations compared to the amount required for input and output of decimal data.

* For this reason, Some computers and all electronic calculators Perform arithmetic operations directly with the decimal data (in a binary code) and thus eliminate the need for conversion.

* The Representation of Signed decimal numbers in BCD is Similar to the representation of signed numbers in binary.

* we can either use the familiar signed-magnitude system or the signed-complement System.

* The Sign of a decimal number is usually represented with four bits i.e, to represent plus, four 0's are used. and to represent minus, BCD equivalent of '9' is used, which is 1001.

* The signed magnitude system is difficult to use with computers.
* The signed ~~magnitude~~ complement System is either the 9's or 10's complement.
* The 10's complement is ~~the~~ most often used system.
* To obtain the 10's complement, we first take the 9's Complement and then add one to the least significant digit.

* Addition is done by adding all digits, including the sign digit, and discarding the end carry.

Ex: $(+375) + (-240) \Longrightarrow +135$, addition done in signed 10's complement system.

$$\begin{array}{rl} 0375 & (0000\ 0011\ 0111\ 0101)_{BCD} \\ +\ 9760 & (1001\ 0111\ 0110\ 0000)_{BCD} \\ \hline 0135 & (0000\ 0001\ 0011\ 0101)_{BCD} \\ \hline \end{array}$$

Floating point Representation:

* The floating point representation of a number has two parts.

  * The first part represents a signed, fixed-Point number called the mantissa.
  * The second part designates the position of the decimal (or binary) point and is called as the exponent.
  * The fixed-Point mantissa may be a fraction or an integer.

Example: The decimal number +6132.789 is represented in floating-point with a Fraction and an exponent as follows.

Fraction	Exponent
+0.6132789	+04

* The value of the exponent indicates that the actual position of the decimal point is four positions to the right of the indicated decimal point in the fraction.

* This representation is equivalent to the scientific notation,

  $+0.6132789 \times 10^{+4}$

* Floating point is always interpreted to represent a numbers in the following form.

$$m \times r^{e}$$

where, m = mantissa

r = radix

e = exponent

* A Floating-point binary number is represented in a similar manner except that it uses base 2 for the exponent.

* For example, the binary number +1001.11 is represented with an 8-bit fraction and 6-bit exponent as follows.

Fraction	Exponent
01001110	000100

* The fraction has a '0' in the leftmost position to denote positive.
* The exponent has the equivalent binary number +4.
* The floating-point number is equivalent to,

$$m \times 2^e \implies +(.1001110)_2 \times 2^{+4}$$

* A Floating-point number is said to be normalized if the most significant digit of the mantissa is non zero.

* for Example, decimal number 350 is normalized but, 000350 is not normalized.

* The number is normalized only if its leftmost digit is nonzero.
* For Example, the 8-bit binary number 00011010 is not normalized because of the three leading 0's.
* The number can be normalized by shifting it three positions to the left and discarding the leading 0's to obtain 11010000.
* The three shifts multiply the number by $2^3 = 8$.
* To keep the same value for the floating-point number, the exponent must be subtracted by 3.

* Normalized numbers provide the maximum possible precision for the floating-point number.

(*) Multiplication Algorithm.

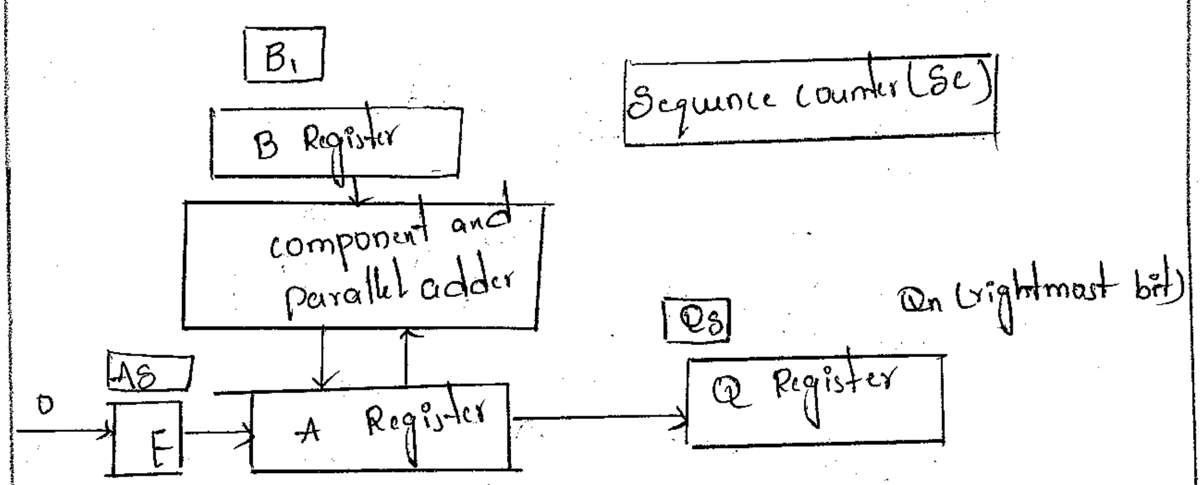
Consider an Example for general multiplication.

```
23      10111   multiplicand
19    x 10011   multiplier
      ---------
          10111
         10111
        00000
       00000
      10111
    -----------
437   110110101
    -----------
```

Multiplication algorithm for Signed magnitude data.

Hardware implementation.

343



Numerical Example:

305 Numerical example for Binary Multiplier

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n=1$; add B		10111		
first partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n=1$; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n=0$; Shift right EAQ	0	01000	10110	010
$Q_n=0$; Shift right EAQ	0	00100	01011	001
$Q_n=1$; add B		10111		
fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000.

final product in

AQ = 0110110101

Booth multiplication Algorithm. (or) multiplication algorithm for signed 2's complement data:-

→ Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation.

Booth Algorithm:

Step 1: The multiplicand is subtracted from the partial Product upon encountering the first least Significant 1 in a String of 1's in the multiplier.

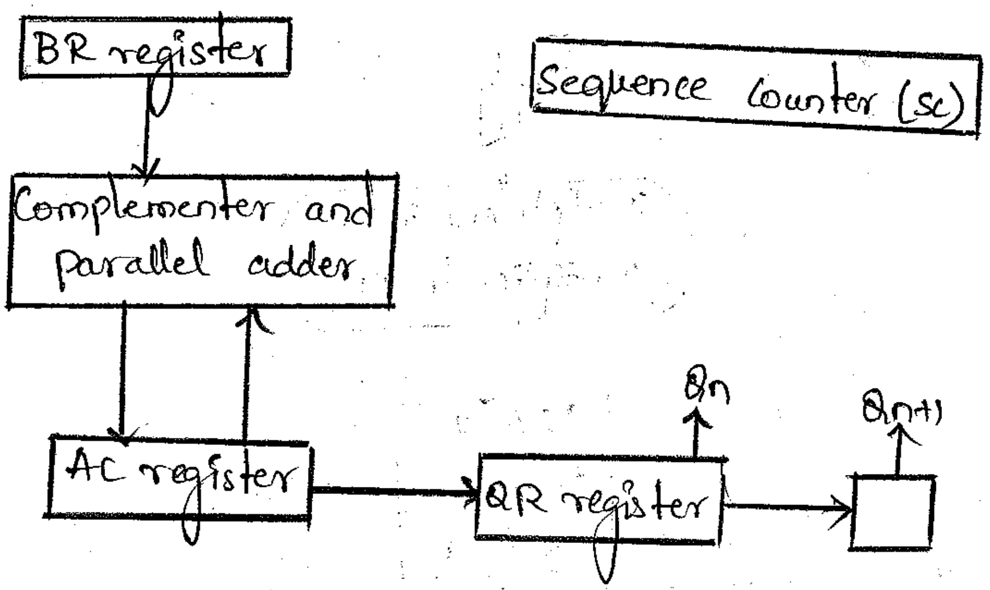
Step 2: The multiplicand is added to the partial Product upon encountering the first 0 (provided that there was a previous 1) in a String of 0's in the multiplier.

Step 3: The partial product does not Change when the multiplier bit is identical to the previous multiplier bit.

→ This algorithm works for positive or negative multipliers in 2's complement representation.

Hardware Implementation.

346



→ The Hardware implementation of Booth algorithm requires the register configuration as shown in figure..

→ In this we Rename registers A, B and Q as, AC, BR and QR respectively.

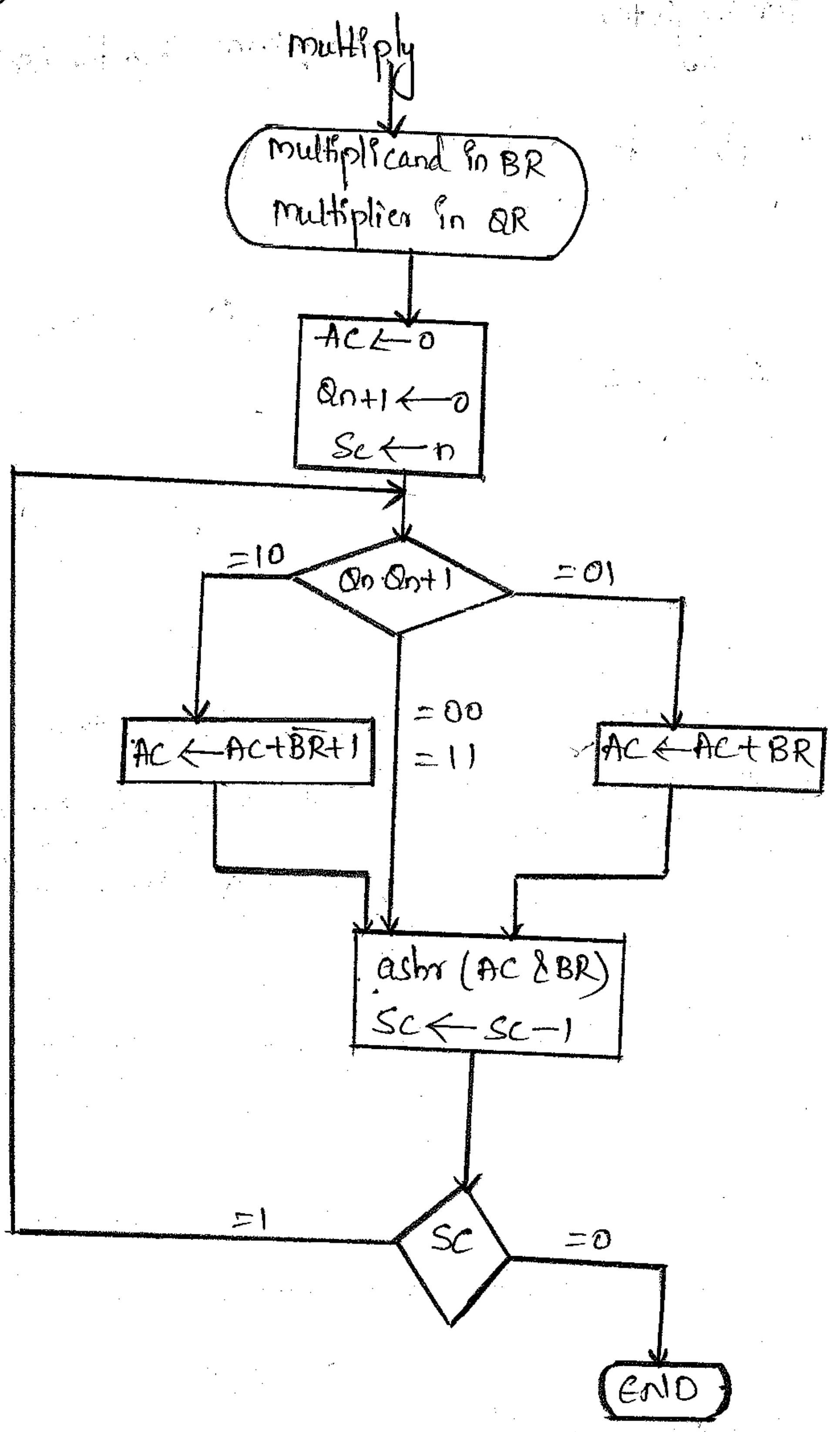
→ Qn designates the least Significant bit of the multiplier in register QR.

→ An Extra flip-flop Q_{n+1} is appended to QR to facilitate a double bit inspection of the multiplier.

→ The flowchart of Booth algorithm is as shown below.

Flow chart – Booth algorithm for multiplication of Signed 2's complement numbers.

347



Numerical Example:

TABLE: EXAMPLE OF MULTIPLICATION WTH BOOTH ALGORITHM

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