



Knowledge... Everywhere

Mathematics

Computing

Table of Content

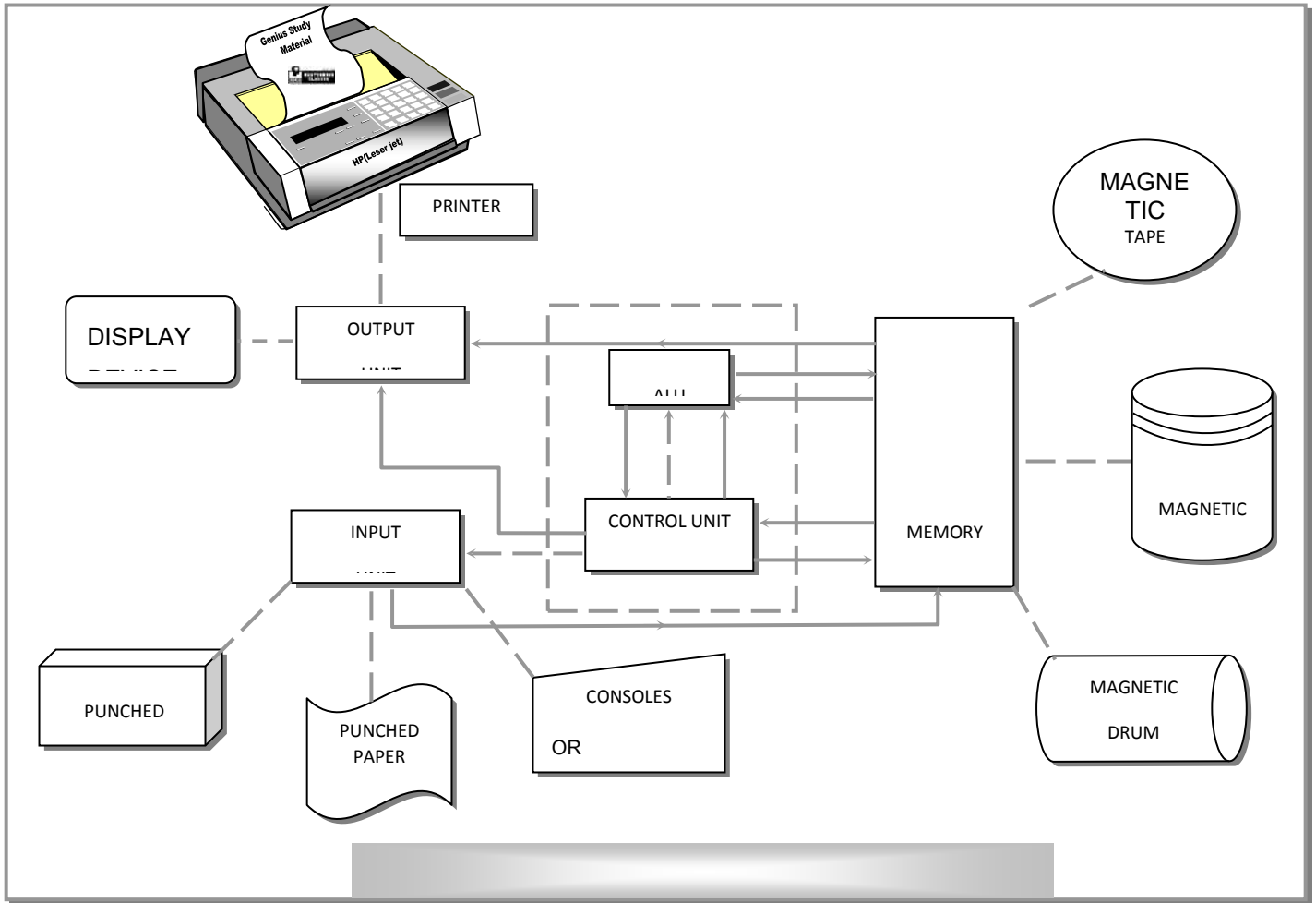
1. Introduction.
2. Memory.
3. Algorithms.
4. Pseudo language.
5. Pseudo language constructs.
6. Flow chart's (Representation of algorithm).
7. Number system.

TestprepKart



1. Introduction.

The modern digital computer or simply a computer is a general purpose electronic machine which can process a large amount of information at a very high speed. A computer can perform millions of computations in a few minutes. It can also perform arithmetical and logical operations.



A computer has five major components:

- (1) Input unit
- (2) Memory unit
- (3) Control unit
- (4) Arithmetic logical unit
- (5) Output unit

(1) **Input unit:** The input unit is the means where the user communicates data or information to the computer.

(2) **Memory unit:** The memory unit stores instructions, data and intermediate results. It supplies, when required, the stored information to the other units of the computer.

(3) **Control unit:** The control unit controls all the activities in the computer by sending electronic command signals to other components of the computer.

(4) **Arithmetic Logical Units (ALU):** ALU is the unit where the arithmetic and logical (e.g., less than, greater than) computations are carried out. Control unit and ALU taken together is called Central Processing Unit (CPU).

(5) **Output unit:** The output unit receives the stored result from the memory unit converts it into a form. The user can understand and produces it in the desired format.

A computer may have more than one input and output units. For example, printer and display screen are two different output units attached to the same computer.



2. Memory.

Our aim is to see how we can use the computer to solve some problems. For that purpose, it is useful to know a little more about main memory. From the user's point of view, main memory can be thought of as a collection of compartments (or locations) as shown in fig. (i) Each compartment is assigned a number called its address (starting with zero as shown in the fig. (ii)). The total number of compartments gives us the size of the memory.

0	1	2
3	4	

fig. (i) Main memory as a collection of compartments (locations)

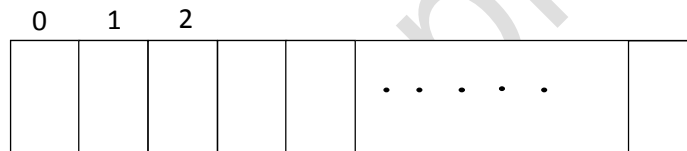


fig. (ii) Bits in a memory location

Each compartment of memory (as well as a register in ALU) consists of sub-compartments fig. (ii). Each sub-compartment can store either a zero or a 1. Any information to be stored inside a computer is put using zeros and 1's. The digits 0 and 1 are called binary digits (bits in short). The acronym bit is formed by taking the letter b from the word 'binary' and the letters i, t from the word 'digit'. Similarly, we have the acronym dit for decimal digit, hit for hexadecimal digit etc. The number system that uses only two digits is called binary number system. Computers use binary number system for computation.



3. Algorithms.

An algorithm is defined as a finite set of rules, which gives a sequence of operations for solving a specific type of problem.

In other words, algorithm is a step-by-step procedure for solving problems.

An algorithm has following five important features:

- (1) Finiteness
- (2) Definiteness
- (3) Completeness
- (4) Input
- (5) Output

(1) **Finiteness:** An algorithm should always terminate after a finite number of steps.

(2) **Definiteness:** Each step of algorithm should be precisely defined. This means that the rules should be consistent and unambiguous.

(3) **Completeness:** The rules must be complete so that the algorithm can solve all problems of a particular type for which the algorithm is designed.

(4) **Input:** An algorithm has certain inputs.

(5) **Output:** An algorithm has certain outputs which are in specific relation to the inputs.

An important consideration for an algorithm concerns its efficiency. Some algorithms are far more efficient than others in that, when programmed, one may require fewer steps or perhaps less memory than another and will therefore, be more satisfactory or economical in actually solving problems on a computer. We shall often deal with considerations of this type in the subsequent work.

In the development of an algorithm, sequence, selection and repetition (or interaction) play an important role.



(1) **Sequence:** Suppose that we want to find the value of the expression $a^3 + 4ab + b^2$ for given values of a and b . Algorithm (i.e., step by step procedure) for achieving this will consist of steps given in fig. to be carried out one after the other.

1. Get the value of a
2. Get the value of b
3. Calculate a^3 , call it S
4. Calculate $4ab$, call it T
5. Calculate b^2 , call it V
6. Find the sum $S + T + V$, call it M
7. Write the value of M as answer.

Steps of an algorithm to evaluate $a^3 + 4ab + b^2$, given the values of a, b

This algorithm, you will agree, is very straightforward, consisting of simple steps which are to be carried out one after the other. We say that such an algorithm is a sequence of steps, meaning that

- (i) At a time only one step of the algorithm is to be carried out.
- (ii) Every step of the algorithm is to be carried out once and only once; none is repeated and none is omitted.
- (iii) The order of carrying out the steps of the algorithm is the same as that in which they are written.
- (iv) Termination of the last step of the algorithm indicates the end of the algorithm.

Here afterwards we shall follow the convention that (i) the successive steps in a sequence will be written on successive lines and hence (ii) steps will not be necessarily numbered as they are in fig.

(2) **Selection:** An algorithm which consists of only a sequence, is not sufficient for solving any type of problem. Let us consider the problem of solving an equation of the type $m + nx = r$ (where m, n, r are given integers) for integral values of x . We immediately use laws of algebra to find $x = (r - m) \div n, n \neq 0$. Let us call an algorithm that works for only some (not necessarily all) possible sets of input values, a semi-algorithm.

Semi-algorithm (for the above problem) :

- Step 1:** Get the values of m, r and n .
- Step 2:** Subtract m from r , call this difference b .
- Step 3:** Divide b by n ; print this result as the value of x .



The above steps are certainly efficient, As an example, let $m = 9, n = 5$ and $r = 24$, in which case we have

$9 + 5x = 24$. Then in step 2, we have $24 - 9 = b$ i.e., $b = 15$ and in step 3, we have $\frac{b}{n} = \frac{15}{5} = 3$, and so we

print $x = 3$.

The above steps have two fatal flaws, however. First, if n equals 0, then either $m = r$ and x can have any integral value, or $m \neq r$ and no solution is possible i.e., there is no integer x which may satisfy the given equation. Second, if there is a non-zero remainder when b is divided by n then again there is no integer x which may satisfy the given equation. So we must modify our algorithm to deal with all such situations as may arise. Given below is the modified algorithm which suits all the possible situations that may arise.

```

Step 1 : Get the value of  $m, n$  and  $r$ 
Step 2 : If  $n = 0$  and  $m = r$ 
           then go to step 7
           else go to step 3
Step 3 : If  $n = 0$  and  $m \neq r$ 
           then go to step 6
           else go to step 4
Step 4 : Subtract  $m$  from  $r$ , call this difference  $b$  (i.e.,
            $b = r - m$ )
Step 5 : Divide  $b$  by  $n$ ;
           If there is a remainder
           then go to step 6
           else print the value of  $\frac{b}{n}$ , which is the required
           value of  $x$ .
Step 6 : Print 'No integer satisfies this equation'. Stop
Step 7 : Print 'Any integer satisfies this equation'. Stop
    
```

Algorithm to solve an equation of the form $m + nx = r$, where m, n, r, x are integers

The above algorithm provides the person or computer that will execute the algorithm with an ability to choose the step to be carried out depending upon the values of m, n and r (and subsequently, the value of b). This ability is called selection. The power of selection is that it permits that different paths could be followed, depending upon the requirement of the problem, by the one who executes the algorithm.

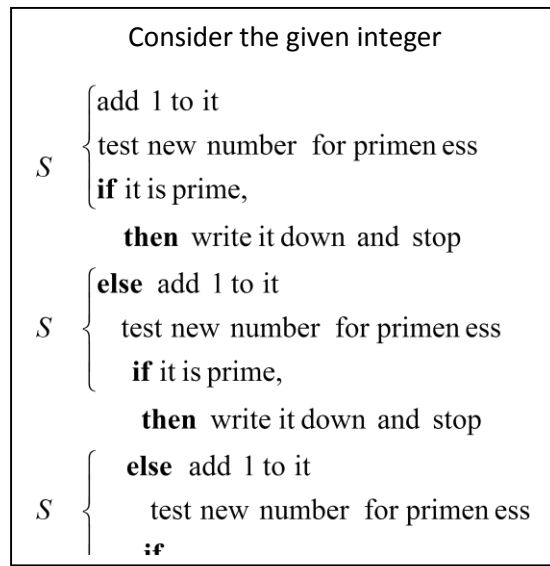
In the above algorithm, selection is expressed by using the special words 'if', 'then', 'else'. Further, it may be noted that all that is written using these special words (once) constitutes one step. Note the way it is



written. Nothing appears below the word 'if' till that step is over. This is known as indentation. The words 'then' and 'else' come with exactly same indentation with respect to the word 'if'.

(3) **Iteration or Repetition:** In forming an algorithm certain steps are required to be repeated before algorithm terminates after giving an answer. This is known as iteration or repetition.

Let us consider the problem of finding the just prime number greater than a given positive integer. The following list of steps shows the step by step procedure to be followed for solving the problem.



Algorithm for finding a prime number greater than a given positive integer

We see in the above procedure that the steps

“add 1 to it
test new number for primeness
if it is prime
then write it down and stop ”

are repeated again and again till (after a finite number of repetitions) we get a prime number and print it. If this sequence (which involves a decision also) is denoted by S, then S is repeated again and again till, we get the result and print the result. This is technically known as iteration or repetition. The way of writing adopted in fig. poses a problem as we do not know the number of times S is repeated. This number depends upon the given positive integer. The difficulty presented above is overcome by introducing a new way of writing iterations in algorithms. The algorithm shown in fig. is (in new ways) then written as shown below



Consider the given number

repeat

add 1 to it

Or

Consider the given number

add 1 to it

while the new number is not
prime

Two different ways of writing iteration
occurring in fig.

4. Pseudo language.

The languages used by human beings for talking and writing among themselves are called natural languages. Expression in a natural language can be ambiguous.

Computer, being a machine, requires that there should be no ambiguity at all when we give instructions to it. Languages used to communicate with a computer are known as programming languages.

We shall use meaningful mnemonic variable names, assignment, symbol ← constructions employing **If-then-else, Repeat-until, While-Do** and other constructions employing word **For** for writing an algorithm. We shall also require instructions to input data in an algorithm as well as instructions to output computed results from an algorithm. All these will constitute our language to present any algorithm. This language will not resemble in total with any actual existing programming language but will have desirable characteristics of a good programming language. We shall call it a pseudo-language.



5. Pseudo language Constructs.

(1) **If-then-else construct** : The general form of this construct which is used to provide selection of actions is

If condition
then step 1

When an instruction using this type of construct is executed, condition determines which of the step 1 and step 2 is to be executed. If condition is true, step 1 is executed, otherwise (i.e., if condition is not true) step 2 is executed.

A particular case of this construct does not have the word else. The general form of this construct is

If condition
then step

Clearly when this form is used, no action is taken when condition is false, and step is executed when condition is true. In other words the following constructs are equivalent as they do the same thing.

If condition	and	If condition
then step		then step

(2) **Repeat until construct**: The general form of this construct is

Repeat
Part of the algorithm
... ..

This construct is used when repetition of certain action is required. Note that "Part of algorithm" is always executed at least once as the condition is tested at the end, unlike the 'while-do' construct where the condition is tested in the beginning.

(3) **While-do-construct:** This construct is an alternative to the 'repeat-until' construct. The general form of this construct, which is also used to provide repetition of instruction is

while	condition
do	T

Where T is a sequence of instructions. When this construct is executed, condition is evaluated first. If the condition is true, the sequence T of instructions is executed and the condition is evaluated again and so on. If the condition is false, execution of T is skipped and the execution of algorithm proceeds with the portion that appears after T. Thus condition is tested again and again till it is false. Every execution of T modifies some variables in the algorithm and eventually after some repetitions, the condition becomes false. This completes the execution of the 'while-do-construct', the execution proceeds to the portion appearing after T.

(4) **For construct:** When we know in advance how many times a part of the algorithm is to be executed we use 'for construct', whose general form is

For identifier = initial value to test value by increment **Do** S.

The word, **For, To, By** and **Do** are reserved words for this construct. Initial value gives the starting value that the identifier should take, when the S is executed. The value of identifier is increased by the increment after each execution. The execution of S continues until the value of identifier exceeds the test value.

6. Flow Charts (Presentation of Algorithm)

A graphic representation of an algorithm is called a 'flow-chart'. A flow chart constitutes a schematic and pictorial representation of the sequence of steps, which are to be executed in solving a problem.

A flow-chart consists of some boxes linked by arrows. In each box, some instruction to be carried out is mentioned. Arrows on the lines connecting the boxes indicate the direction, in which we should proceed. The boxes are of different shapes. Each particular shape is associated with a specific type of instruction as shown in fig.

Flow-chart conventions:

While drawing a flow-chart, the following conventions are observed.

- (i) The general direction of flow is from left to right and from top to bottom.
- (ii) Only one flow-line should leave a process symbol.
- (iii) Only one flow line should enter a decision box and atleast two lines must leave it.



(iv) A flow line that goes in upward direction, completes and iteration (or repetition) or a loop.

Basic operations and flow-charts:

The three basic operations are:

- (1) Sequence
- (2) Selection
- (3) Iteration

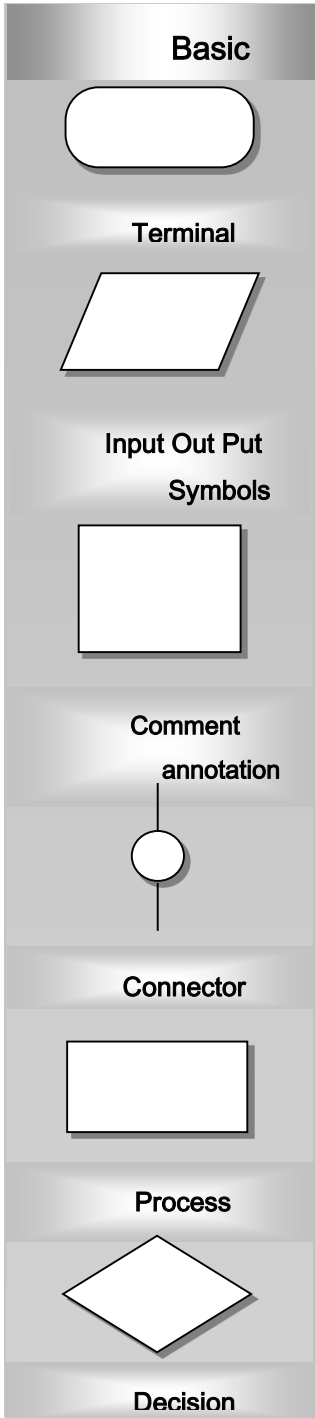
The selection of a flow-chart corresponding to an iteration, or the Repeat-Until construct or the While-Do construct gives rise to a cycle, usually called a loop. There are two types of loops:

(i) When an operation is repeated, a fixed number of times, whatever the value of the variables involved may be, then the corresponding section of the flow diagrams gives rise to a fixed loop.

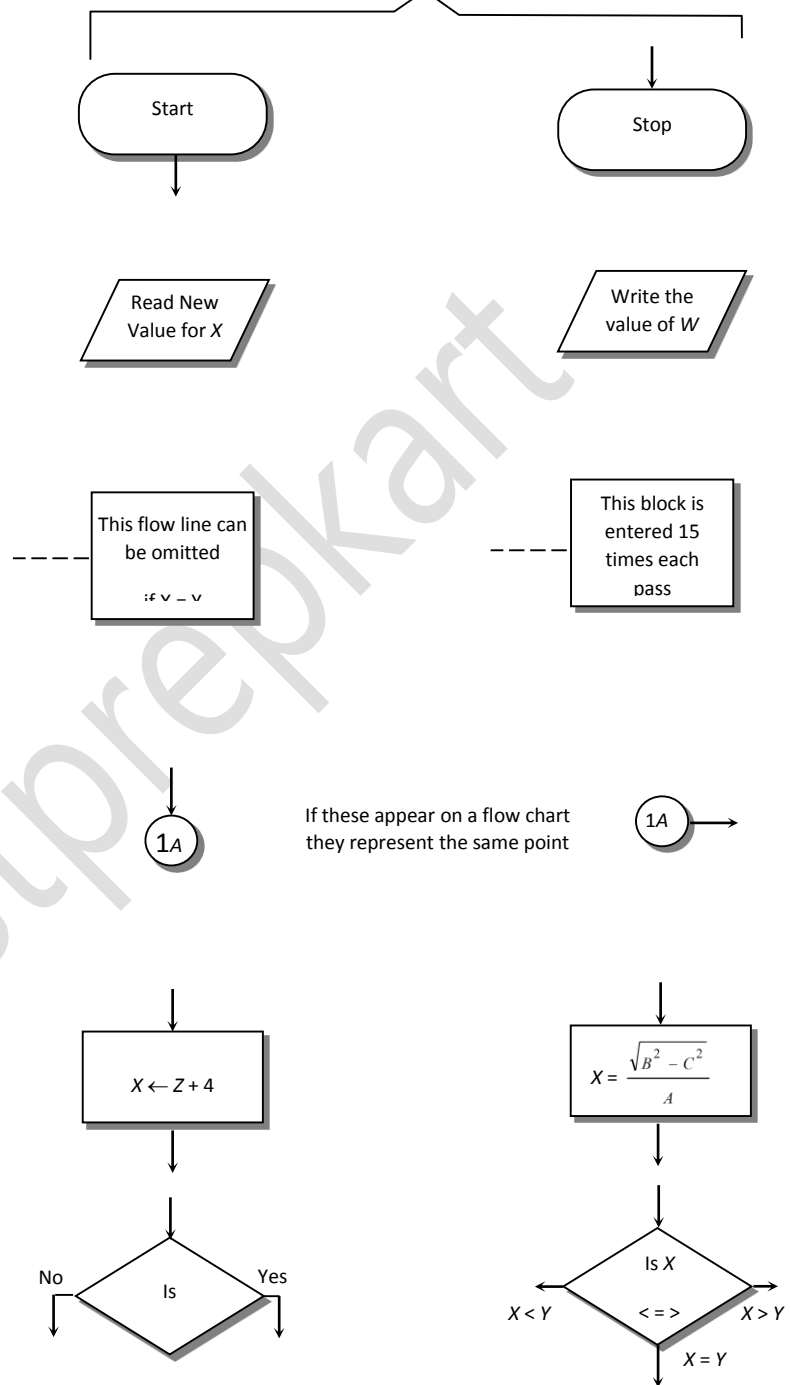
TestprepKart



(ii) When the number of times an iteration is to be carried out depends upon the values of the variables, then the corresponding section of the flow diagrams gives rise to variable loop. This loop is also known as backward jump.



Examples of Use



FLOW CHART SYMBOLS



7. Number system.

(1) **Decimal system:** Number system which we use in our daily life is the decimal system. In decimal system we use the digits namely 0, 1, 2,.....8, 9 and with the help of these 10 digits we are able to write any rational number. The decimal system is a place-value system, meaning thereby that the value represented by a digit depends upon the place of the digit within the numeral. The value assigned to consecutive places in the decimal system are $10^4, 10^3, \dots, 10^0, 10^{-1}, 10^{-2}, \dots$ (from left to right)

Example: Number 3864.342 can be written as

$$3864.342 = 3 \times 10^3 + 8 \times 10^2 + 6 \times 10^1 + 4 \times 10^0 + 3 \times 10^{-1} + 4 \times 10^{-2} + 2 \times 10^{-3}$$

As ten basic symbols are used for representing the numbers, ten is called the base of the system and the system is called base-ten system or decimal system.

(2) **Binary number system:** The number system for which the base is two is called the binary system. In this system numbers are represented with the help of two basic symbols namely 0 and 1. The values assigned to consecutive places in the system are (when expressed in the decimal system)....

$2^4, 2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, \dots$ Where 2^0 place is the unit place. The binary numeral can be converted into the decimal numeral and vice-versa.

(3) **Octal number system:** As the name implies this is base eight (2^3) system. The numerals are written with the help of eight basic symbols namely 0, 1, 2,.....,7. The value (expressed in the decimal system) assigned to consecutive places are $8^3, 8^2, 8^1, 8^0, 8^{-1}, 8^{-2}, \dots$, where 8^0 place is the unit's place. The procedures for converting a decimal numeral into an octal numeral and the other way round are similar to the procedures discussed in connection with binary system.

(4) **Hexadecimal system:** As the name implies, this is the base sixteen system. The numerals are written with the help of sixteen symbols, namely 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Note that the symbol A represents ten (in decimal system). Similarly B, C, D, E, F represent respectively the numbers 11, 12, 13, 14 and 15. The value assigned to consecutive places are, $16^2, 16^1, 16^0, 16^{-1}, \dots$ (as expressed in decimal system), where 16^0 place is unit's place.

