

Problem -1

- You all have just completed your B.Sc. from your respective universities. Suppose that you are asked to find out, using a computer, the details of the student who has obtained highest marks in the final year Mathematics exam in your class under the assumption that only one student has topped the exam. Also presume that you have access to all the required data.

Steps to follow to solve the problem on a Computer

- 1) Carefully analyze the given problem and write out the Algorithm to find the solution to the problem.
- 2) Draw a flowchart to enable the correct coding of your algorithm
- 3) Write out a pseudo code in English Language

What is an Algorithm?

What is a Flowchart?

What is a Pseudo code?

Algorithm:

A finite number of precise and concise unambiguous instructions, to solve a given problem, which can be executed in a finite time on a computer to obtain the envisaged solution.

Algorithm for the Problem 1:

Input: Decide on the format of the data, say data consists of
a) Student Name, b) Student Roll Number, c) Marks obtained

Method:

1. Read the first student's paper and note down the required data details (Name, Roll No, Marks). Set this as the highest mark to begin with.
2. Repeat the steps 3 and 4 until no papers are left.
3. Read the next student's data details. Compare the marks of the current student with the highest marks.
4. If the current marks is greater than the highest marks noted so far then replace all the details corresponding to the highest marks by those read in step 3. Else do nothing.
5. Print the data details corresponding to the highest marks

Flowchart: It is a graphical illustration of the steps under the algorithm used to understand and check the algorithm.

Some standard conventions for flowchart:

•Rectangles with rounded ends are used to indicate start and stop



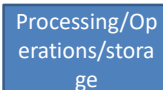
•Parallelograms are used to represent input and output operations

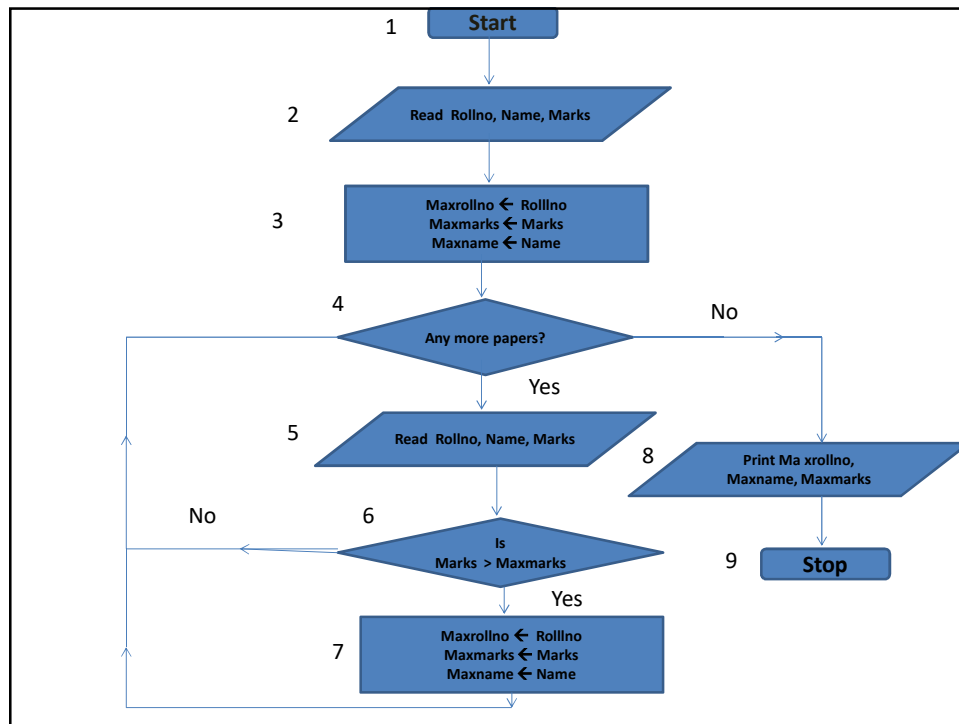


•Diamond shaped boxes are used to indicate questions asked or conditions tested based on whose answers appropriate exits are taken by a procedure. The exits from the diamond shaped box are labelled with the answers to the questions.



•Rectangles are used to indicate any processing operation such as storage and arithmetic.





Pseudocode: A dummy computer language for "tentative discussion"

```

Program <FIND_TOPMARKS(Input, Output)>
Variables Name, Rollno, Marks, Maxname, Maxrollno,
Maxmarks;
Begin
    Readdata(Name, Rollno, Marks);
    Set {
        Maxmarks ← Marks;
        Maxrollno ← Rollno;
        Maxname ← Name; }
    While not_EOF(Input) do
        begin
            Readdata(Name, Rollno, Marks);
            if (Marks > Maxmarks) then
                Set {
                    Maxmarks ← Marks;
                    Maxrollno ← Rollno;
                    Maxname ← Name; }
            end;
        Printdata(Maxname, Maxrollno, Maxmarks);
    End;
  
```

Pseudocode Keywords

- Program <name(I,O)>
- ;
- Variables
- BeginEnd
- begin ... end
- Readdata(.,.,.)
- Set {..}
- Printdata(.,.,.)

Pencil Trace of the Pseudo code

Pencil Trace

Name	Roll No	Marks
X1	2020MIT01	78
X2	2020MIT02	61
X3	2020MIT03	95
X4	2020MIT04	54
X5	2020MIT05	81
X6	2020MIT06	98
X7	2020MIT07	91
X8	2020MIT08	85
X9	2020MIT09	99
-	-	-

```

set
Maxmarks =78
Maxrollno=2020MIT01
Maxname=X1
While not EOF(input) do → End of stack of papers not reached so proceed
  Readdata(Name, Rollno, Marks); → Name = X2, Rollno=2020MIT02,
  Marks=61
  If ( Marks > Maxmarks) then → if (61>78) → FALSE
  While not EOF(input) do → End of stack of papers not reached so proceed
    Readdata(Name, Rollno, Marks); → Name = X3, Rollno=2020MIT03,
    Marks=95
    If ( Marks > Maxmarks) then → if (95>78) → TRUE
    So Maxmarks = 95, Maxrollno=2020MIT03, Maxname=X3
    While not EOF(input) do → End of stack of papers not reached so proceed
      Readdata(Name, Rollno, Marks); → Name = X3, Rollno=2020MIT03,
      Marks=54
      If ( Marks > Maxmarks) then → if (54>95) → FALSE
      .....
      .....
    While not EOF(input) do → End of stack of papers not reached so proceed
      Readdata(Name, Rollno, Marks); → Name = X9, Rollno=2020MIT09,
      Marks=99
      If ( Marks > Maxmarks) then → if (99>95) → TRUE
      So Maxmarks = 99, Maxrollno=2020MIT09, Maxname=X9
      While not EOF(input) do → End of stack of papers has been reached so print
      result
OUTPUT:
X9 2020MITX09 99
  
```

Computer Programming Languages

A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. Most programming languages consist of instructions for computers.

Different Programming Paradigms:

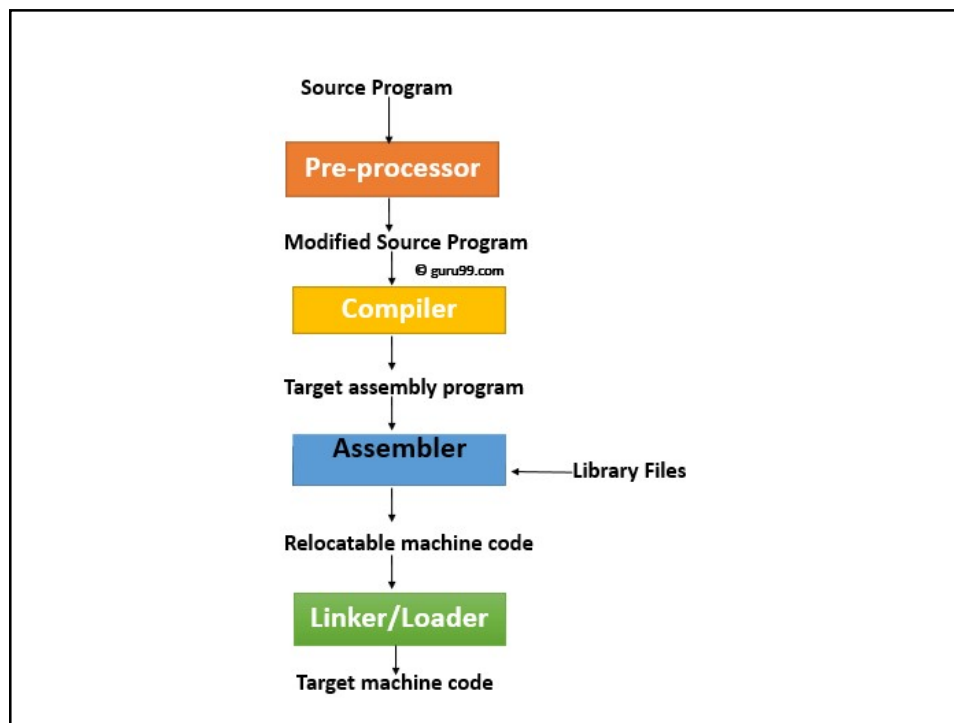
Based on the **programming** philosophy, style, or general approach to writing code there different programming paradigms such as:

- **imperative** in which the programmer instructs the machine how to change its state,
- **procedural** which groups instructions into procedures, ← **(OUR CURRENT FOCUS)**
- **object-oriented** which groups instructions together with the part of the state they operate on,
- **declarative** in which the programmer merely declares properties of the desired result, but not how to compute it
- **functional** in which the desired result is declared as the value of a series of function applications,
- **logic** in which the desired result is declared as the answer to a question about a system of facts and rules,
- **mathematical** in which the desired result is declared as the solution of an optimization problem
- **Symbolic** techniques such as **reflection**, which allow the program to refer to itself, might also be considered as a programming paradigm. However, this is compatible with the major paradigms and thus is not a real paradigm in its own right

Classification of Programming Languages

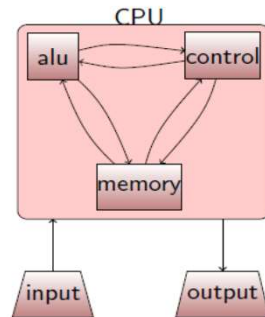
The programming language in terms of their performance reliability and robustness can be grouped into five different generations:

- **First generation languages (1GL)**
 - [Low-level languages](#) that are [machine language](#).
- **Second generation languages (2GL)**
 - Low-level [assembly languages](#). They are sometimes used in [kernels](#) and hardware [drives](#), but more commonly used for video editing and video games.
- **Third generation languages (3GL)**
 - [High –Level languages](#), such as [C](#), [C++](#), [Java](#), [JavaScript](#), and [Visual Basic](#).
- **Fourth generation languages (4GL)**
 - languages that consist of statements similar to statements in a human language. Fourth generation languages are commonly used in database programming and scripts examples include [Perl](#), [PHP](#), [Python](#), [Ruby](#), and [SQL](#).
- **Fifth generation languages (5GL)**
 - programming languages that contain visual tools to help develop a program. Examples of fifth generation languages include Mercury, OPS5, and [Prolog](#)



Quick Look at 1GL & 2GL

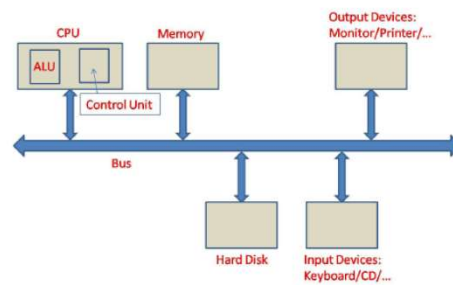
von Neumann architecture



Basic instructions

- ▶ Read/write data from memory/IO devices,
- ▶ Perform arithmetic logical operations on data,
- ▶ Read instructions from memory and interpret them.
- ▶ Proceed according to the instructions.

FUNCTIONAL UNITS



CPU: Central Processing Unit (**ALU:** Arithmetic and Logic Unit; **Control Unit:** Executes instructions)
Memory: Storage area; quickly accessible from CPU
Hard Disk: Storage area is not so quickly accessible to CPU

BINARY FORMAT

- In a computer, everything is stored in **binary format**: a sequence of 0's and 1's.
- The components of a computer understand only binary format.
- Number 4 is stored as 00000100, 1 is stored as 00000001 etc.

EXECUTION IN A COMPUTER

- To begin with, all the data and commands related to a computation is stored in Memory.
- Commands are then brought into the CPU through the Bus, one at a time.
- Each command is executed inside the CPU in the following way:
 - ▶ If the command requires data, it is brought to CPU from Memory
 - ▶ Command is then executed using the data
 - ▶ The command may be for storing data present inside the CPU to Memory
- A **program** is a collection of commands.

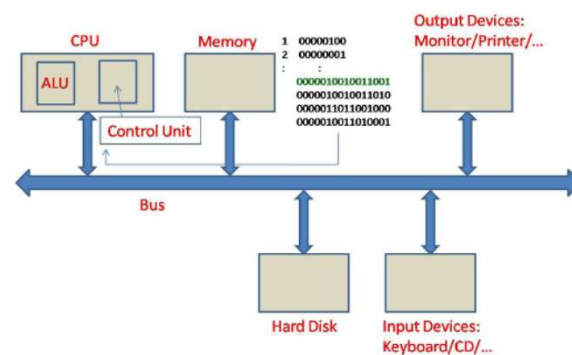
A SMALL PROGRAM

```

0000010010011001 - read memory location 001
0000010010011010 - read memory location 010
0000011011001000 - add two numbers read
0000010011010001 - store the result in memory location 001

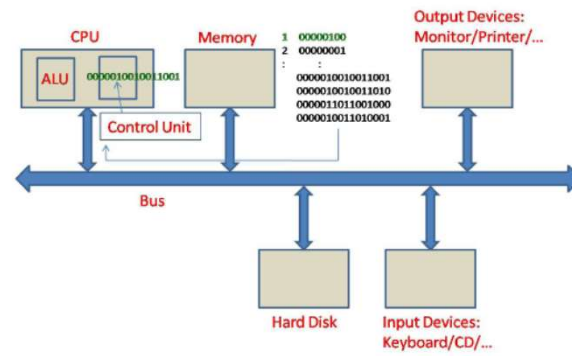
```

EXECUTION



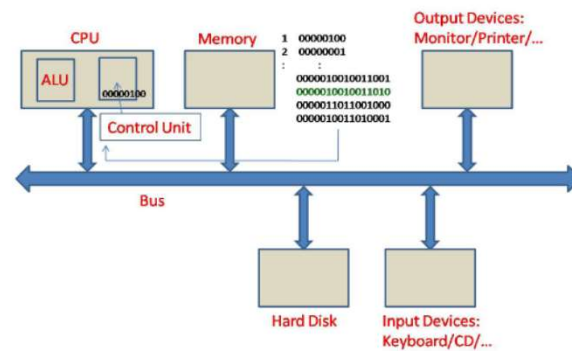
Step 1: Bring 0000010010011001 to CPU
 0000010010011001 = Bring data stored in memory location 001 to CPU

EXECUTION



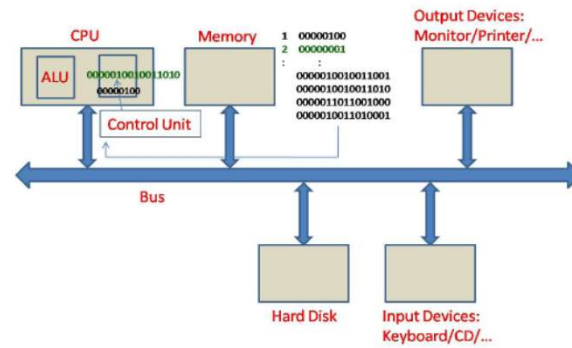
Step 2: Execution of 0000010010011001 in Control Unit
Brings data stored in location 1, number 4, to CPU

EXECUTION



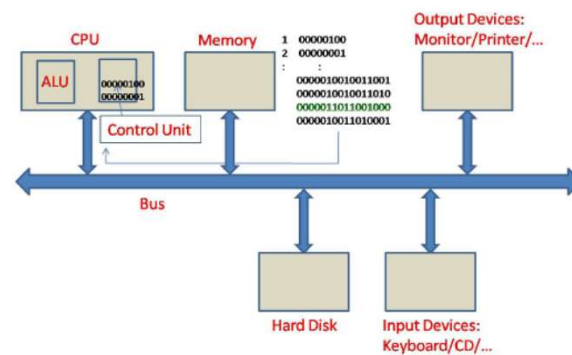
Step 3: Bring 0000010010011010 to CPU
0000010010011010 = Bring data stored in memory location 010 to CPU

EXECUTION



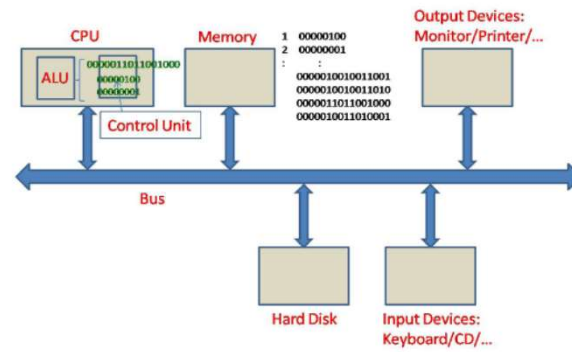
Step 4: Execution of 0000010010011010 in Control Unit
Brings data stored in location 2, number 1, to CPU

EXECUTION

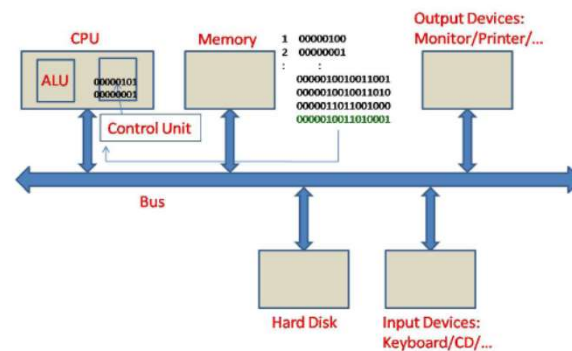


Step 5: Bring 0000011011001000 to CPU
0000011011001000 = Add two stored numbers

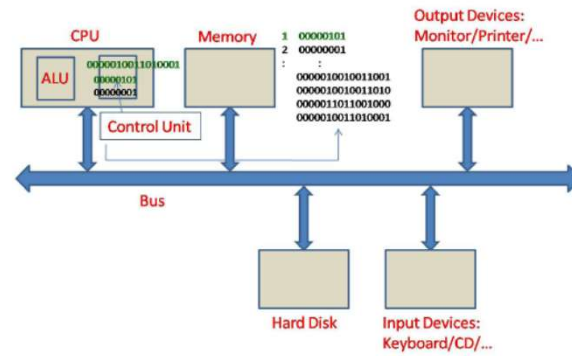
EXECUTION



EXECUTION



EXECUTION



Step 8: Execution of 0000010011010001 in Control Unit
Stores data stored in ALU , number 5, in location 1