CHAPTER I: Introduction to Computer Systems

1. General Introduction

1.1. Computer Science

The Computer Science is the science that concerns the automatic processing and storage of information. The term 'information' refers to any set of data (text, numbers, images, audio, video, etc.). All information is manipulated in binary form ('0' and '1').

1.2. The Computer

A computer is a machine capable of storing and processing information in a purely automatic manner. It can solve problems by applying pre-defined instructions (actions and operations).

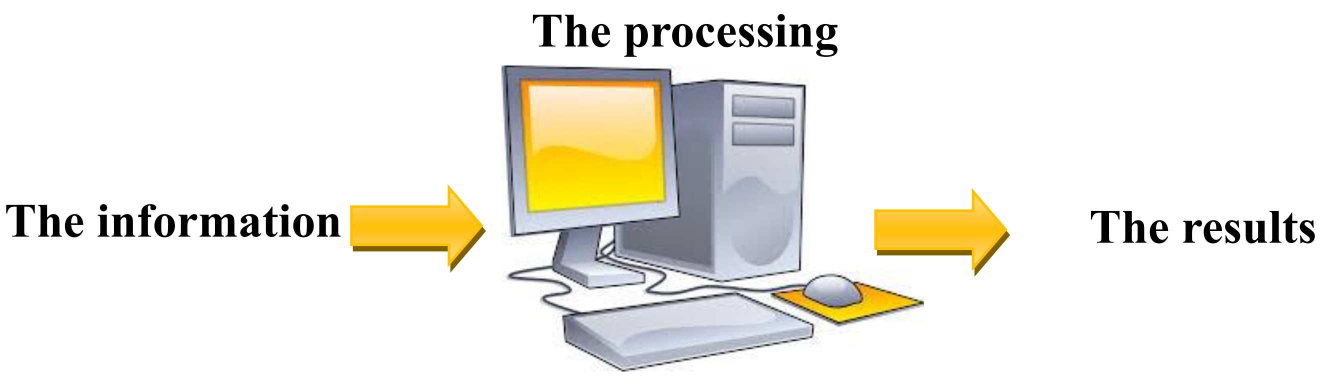


Fig1. How a computer works

1.2.1. The Importance and Utility of a Computer:

This computer tool allows us to accelerate the processing of large volumes of data and obtain correct, precise, and error-free calculation results within reasonable timeframes.

1.3. Architecture

The architecture of a computer system refers to the functional appearance presented to the user [Amdahl 1964]. It provides a description of what happens from the user's perspective (the programmer).

The architecture can be seen as the system's appearance.

2. Concept of Computer System

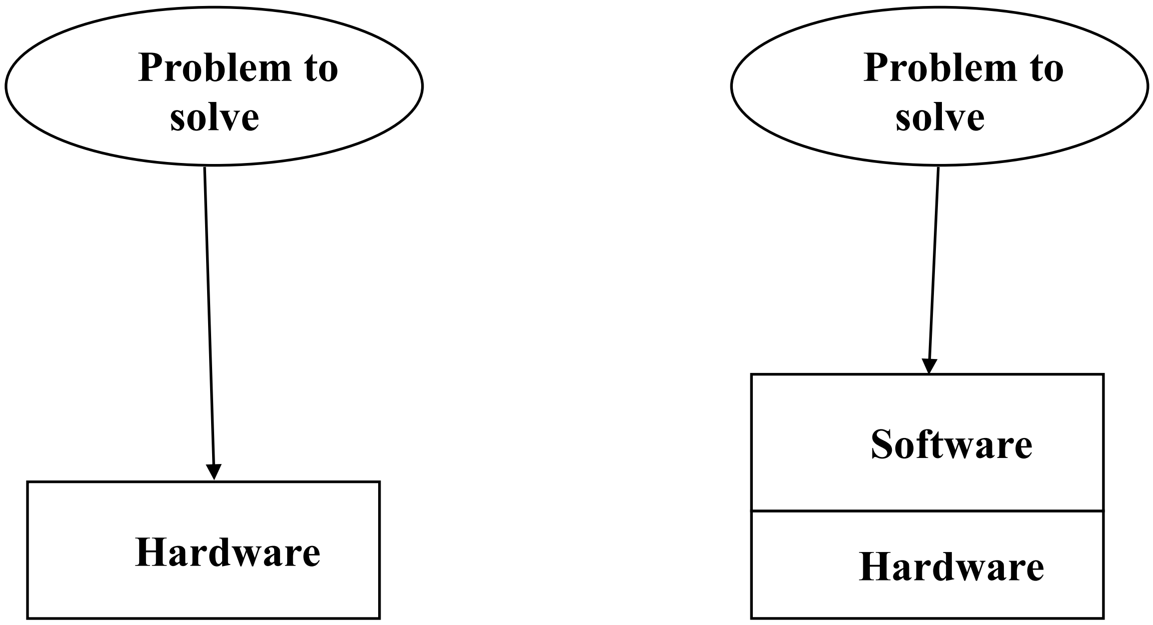
2.1. Description of the Computer

The computer is composed of two parts:

- **The hardware:** This comprises the collection of electronic and electrical equipment within the computer; it is the visible part for the user. This part is also referred to as the real machine.
- **The software:** This encompasses the set of programs (sequences of instructions) that run (execute) on the physical machine. This part is also called the virtual machine (imaginary).
  - **Program:** A set of instructions that directs a computer on what to do. There are three types of software:
    - **The basic software (operating system):** This is the collection of programs that manage the hardware part of the computer. ***Examples*** : DOS, Windows 7, UNIX, etc. Its presence is essential for the computer's operation. These programs are used to increase the level of abstraction between the user and the hardware part, aiming to hide the details and present to the user only a virtual machine with a clear and understandable language.
    - **Service software (system software):** These programs are not necessary for the machine's operation and functioning, but they provide services to users and programmers for better utilization of the machine. Examples: text editors, compilers, etc.
    - **Application software:** This category encompasses all programs that run on the machine under the control of the operating system, such as:
      - Word processing software: Word, Kword, LaTex, etc.
      - Image processing and synthesis software (Photoshop, etc.).
      - Computer-aided design software (CAD: AUTOCAD, etc.).
      - Video games.
      - Management software.
      - Software for calculation, statistics, simulation, mathematics, etc.
      - ...

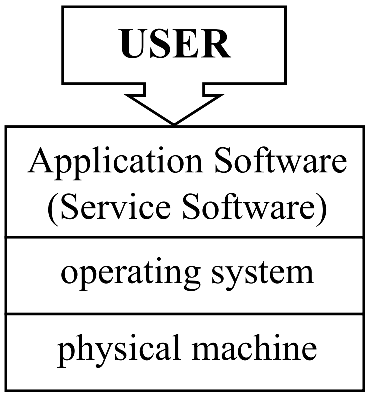
2.2. The Role of the Software Component

The human thinks at a high level of abstraction. To use a computer, he has to transform his concepts (ideas) into a form that can be executed on the computer (algorithm and program). The distance between a problem and a computer is illustrated in the following diagram:

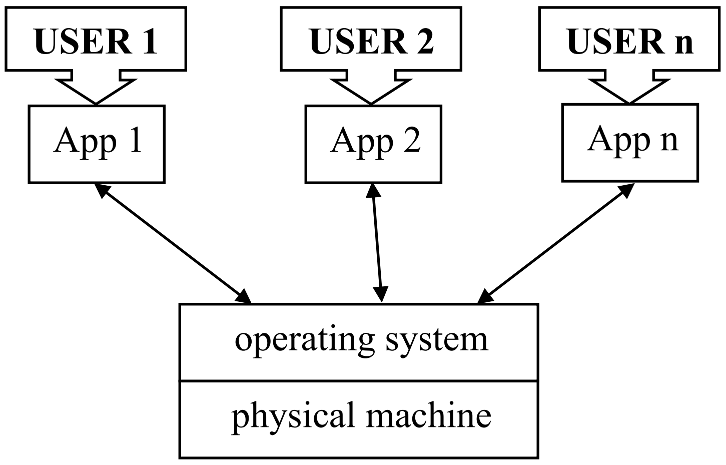


Distance between a problem (a) the old computers and (b) the new computers

Conceptually, if we represent the functions of a computer system in the form of layers, we could describe them as follows:



Single-User Model



Multi-User Model

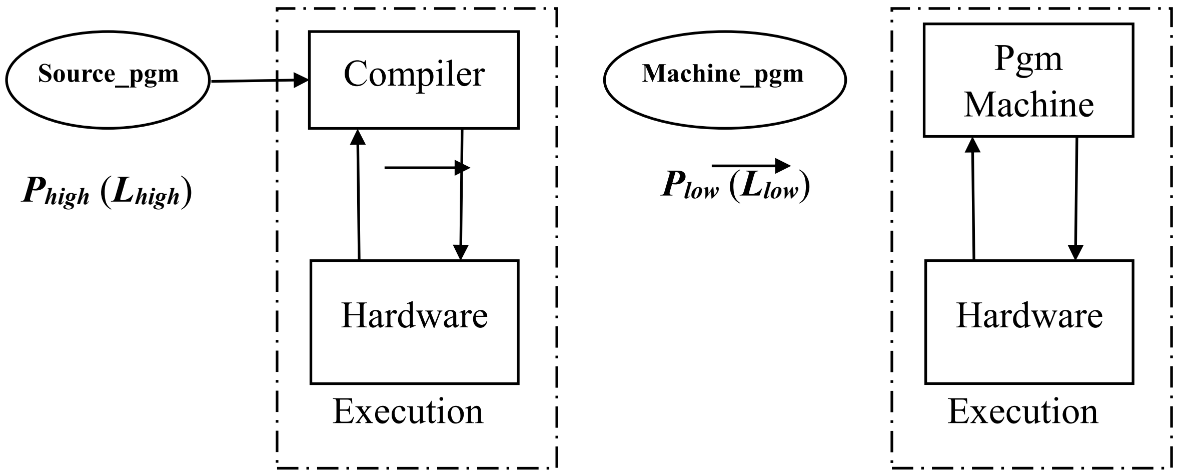
2.3. Transition mechanisms between levels

2.3.1. Compilation phase

During the compilation phase of a source program P_{high} (written in a high-level language L_{high}) to a program written in a low-level language L_{low} (understandable and executable by the machine), each expression in the P_{high} program is replaced by a series of instructions in the L_{low} language in such a way that they maintain the same semantics (execution meaning) as the original expressions in P_{high}. This translation process is carried out by a specialized program called a 'Compiler' (usually written in the L_{low} language). It takes the original P_{high} program as input and generates an executable P_{low} program as output.

After the compilation phase, the P_{low} program is loaded into the machine's memory for execution. During this execution, the source program P_{high} is no longer necessary and can be deleted at any time. Nevertheless, it is advisable to keep P_{high} for potential modifications.

The development of compilers is a highly complex task.

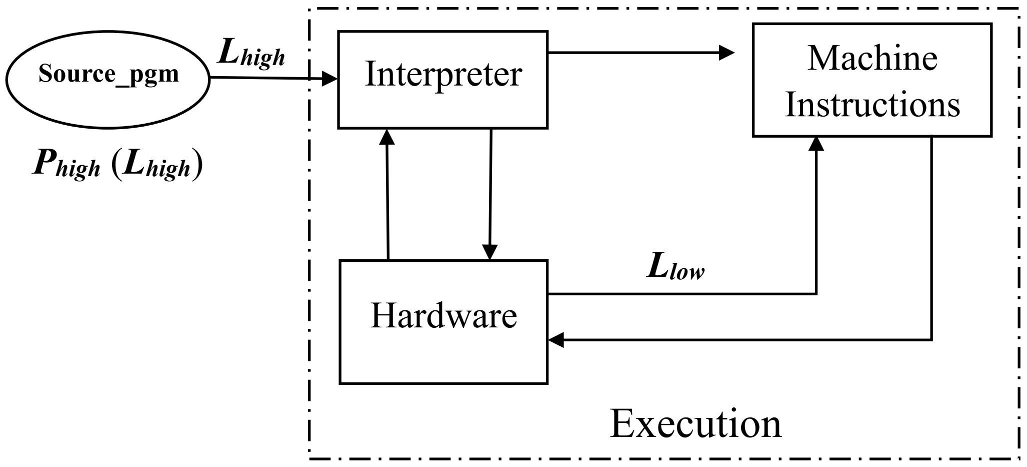


2.3.2. Interpretation phase

When the program P_{high} is written in the L_{high} language, it is interpreted at the L_{low} level, requiring a specialized program called an 'Interpreter'.

The task of interpreting an instruction from P_{high} involves analyzing and provoking the execution of a series of L_{low}-level instructions with the same semantics. This process continues for the remaining instructions in P_{high} until the last instruction (end of the program).

The advantage of this transition mechanism is that the interpreter is relatively small, making it easier to create a new virtual machine instead of a compiler.


Source_pgm
L_{high}
P_{high} (L_{high})
Interpreter
Machine Instructions
Hardware
L_{low}
Execution