

Department of Cybersecurity Technologies Engineering

PROGRAMMING ESSENTIALS

Lecture 1: Introduction



Dr. Hayder Kareem Algabri

hayder_kareem_algabri@hilla-unc.edu.iq



2024-2025

Lecture 1: Introduction

Contents

1. Introduction	2
[1] Computer:	2
Functionalities of a Computer	2
[2] Types of Programs	3
[3] Computer Components:.....	4
[4] How do computers store data?	4
2. What is a programming language?	6
[1] Low-level language	7
[2] Machine-level language	7
[3] Assembly Language	7
[4] High-Level Language	7
[5] Differences between Machine-Level language and Assembly language: .	8
[6] Differences between Low-Level language and High-Level language:	9
3. Introduction To Compilers	10
4. Introduction to Interpreters:	11
5. Object file.....	12
6. Executable file	12

Lecture 1: Introduction**1. Introduction****[1] Computer:**

Computer is an advanced electronic device that takes raw data as an input from the user and processes it under the control of a set of instructions (called a program), produces a result (output), and saves it for future use. This tutorial explains the foundational concepts of computer hardware, software, operating systems, peripherals, etc., along with how to get the most value and impact from computer technology.

Functionalities of a Computer

If we look at it in a very broad sense, any digital computer carries out the following five functions –

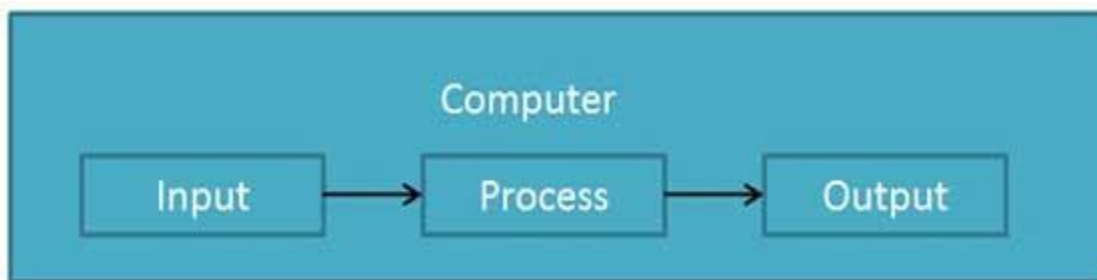
Step 1 – Takes data as input.

Step 2 – Stores the data/instructions in its memory and uses them as required.

Step 3 – Processes the data and converts it into useful information.

Step 4 – Generates the output.

Step 5 – Controls all the above four steps.



Lecture 1: Introduction

[2] Types of Programs

There are two categories of programs. **Application** programs (usually called just "applications") are programs that people use to get their work done. Computers exist because people want to run these programs. **Systems** programs keep all the hardware and software running together smoothly. The difference between "application program" and "system program" is fuzzy. Often it is more a matter of marketing than of logic.

Application Programs	Systems Programs
• Word processors • Game programs • Spreadsheets • Data base systems • Graphics programs • Web browsers	• Operating system. • Networking system. • Database system. • Programming language software. • Web site server. • Data backup.

The most important systems program is the **operating system**. The operating system is always present when the computer is running. It coordinates the operation of all the hardware and software components of the computer system. The operating system is responsible for starting application programs running and finding the resources that they need. When an application program is running, the operating system manages the details of the hardware for it. For example, when you type characters on the keyboard, the operating system determines which application program they are intended for and does the work of getting them there.

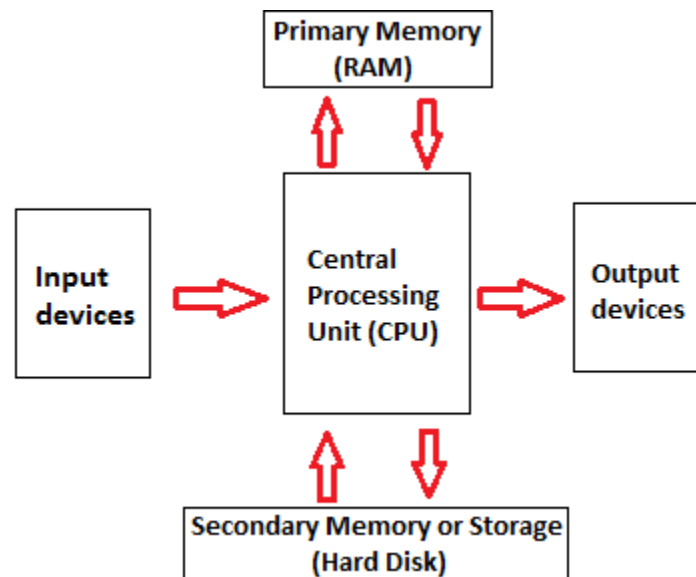
Modern operating systems usually come with a **user interface** that enables users to easily interact with application programs (and with the operating system itself) by

Lecture 1: Introduction

using windows, buttons, menus, icons, the mouse, and the keyboard. Examples of operating systems are Unix, Windows 98, Windows NT, Linux, Solaris, and System 7.

[3] Computer Components:

Computer systems consist of three components: Central Processing Unit, Input devices and Output devices. Input devices provide data to the processor (processing unit), which processes it and generates useful information that's displayed to the user through output devices. Output devices are used to show the processed data to the user on computer screen or in print form. Below figure depicts the major components of computer systems.

**[4] How do computers store data?**

According to the program in hand, the CPU stores the data in RAM or the storage drive. If it has to be used frequently, it will stay in the RAM. Otherwise, it will go to the long-term storage.

If I am editing a text file on a notepad app, the important files of both the video data and the software will be stored in the RAM if the system has enough of it. If the

Lecture 1: Introduction

RAM is low, the data may be fetched from the secondary storage (SSD or HDD) and then pushed into the RAM. If I have to save the file, the CPU will send everything to the secondary storage but keep the files in the RAM until the software is closed.

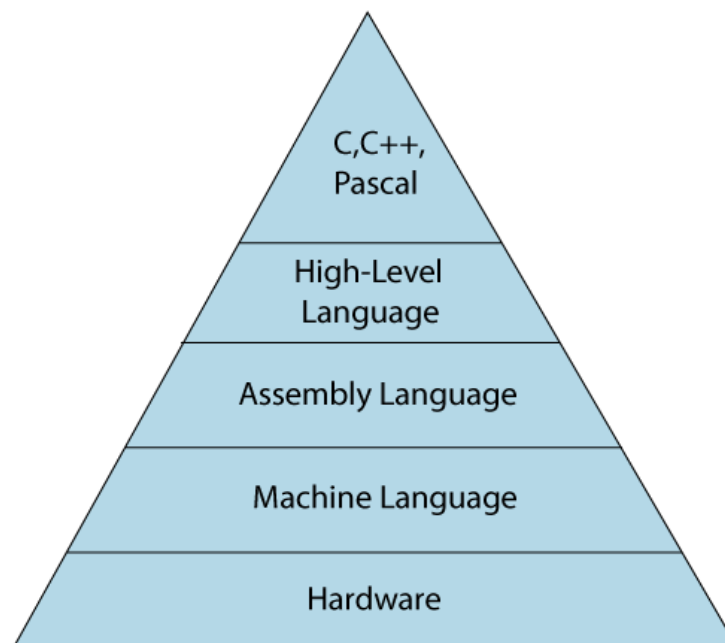
In terms of operating system. The BIOS loads up the required file to first initiate the hardware and then do things like loading up settings, displaying the welcome menu, etc. After the BIOS, the OS is the first layer of software on our computer. It provides us with a platform to do things like running programs and games.

*Lecture 1: Introduction***2. What is a programming language?**

A programming language defines a set of instructions that are compiled together to perform a specific task by the CPU (Central Processing Unit). The programming language mainly refers to high-level languages such as C, C++, Pascal, Ada, COBOL, etc.

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. Thousands of programming languages have been developed till now, but each language has its specific purpose. These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction. Based on the levels of abstraction, they can be classified into two categories:

- Low-level language
- High-level language.



Lecture 1: Introduction**[1] Low-level language**

The low-level language is a programming language that provides no abstraction from the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.

[2] Machine-level language

The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1. As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes. Creating a program in a machine-level language is a very difficult task as it is not easy for the programmers to write the program in machine instructions.

[3] Assembly Language

The assembly language contains some human-readable commands such as mov, add, sub, etc. The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language.

[4] High-Level Language

The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.

When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.

A compiler is required to translate a high-level language into a low-level language.

Lecture 1: Introduction**[5] Differences between Machine-Level language and Assembly language:**

Machine-level language	Assembly language
The machine-level language comes at the lowest level in the hierarchy, so it has zero abstraction level from the hardware.	The assembly language comes above the machine language means that it has less abstraction level from the hardware.
It cannot be easily understood by humans.	It is easy to read, write, and maintain.
The machine-level language is written in binary digits, i.e., 0 and 1.	The assembly language is written in simple English language, so it is easily understandable by the users.
It does not require any translator as the machine code is directly executed by the computer.	In assembly language, the assembler is used to convert the assembly code into machine code.
It is a first-generation programming language.	It is a second-generation programming language.

Lecture 1: Introduction**[6] Differences between Low-Level language and High-Level language:**

Low-level language	High-level language
It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1.	It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans.
The low-level language takes more time to execute.	It executes at a faster pace.
It requires the assembler to convert the assembly code into machine code.	It requires the compiler to convert the high-level language instructions into machine code.
The machine code cannot run on all machines, so it is not a portable language.	The high-level code can run all the platforms, so it is a portable language.
It is memory efficient.	It is less memory efficient.
Debugging and maintenance are not easier in a low-level language.	Debugging and maintenance are easier in a high-level language.

Most Popular Programming Languages –

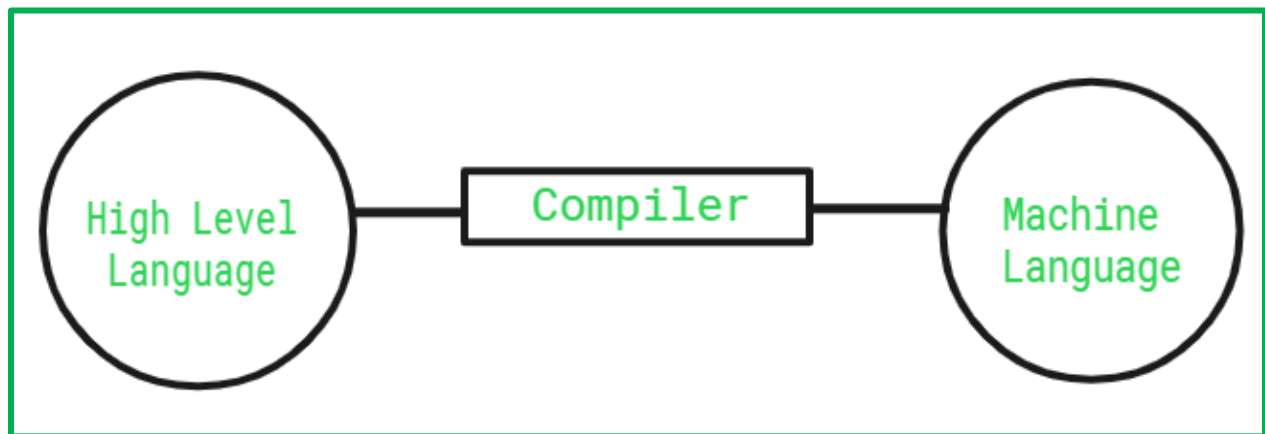
- ❖ C
- ❖ Python
- ❖ C++
- ❖ Java
- ❖ SCALA
- ❖ C#
- ❖ R
- ❖ Ruby
- ❖ Go
- ❖ Swift
- ❖ JavaScript.

Lecture 1: Introduction

3. Introduction To Compilers

A Compiler is a software that typically takes a high level language (Like C++ and Java) code as input and converts the input to a lower level language at once. It lists all the errors if the input code does not follow the rules of its language. This process is much faster than interpreter but it becomes difficult to debug all the errors together in a program .

A compiler is a translating program that translates the instructions of high level language to machine level language. A program which is input to the compiler is called a Source program. This program is now converted to a machine level language by a compiler is known as the Object code.



There are different Compilers :

- **Cross-Compiler** – The compiled program can run on a computer whose CPU or Operating System is different from the one on which the compiler runs.
- **Bootstrap Compiler** – The compiler written in the language that it intends to compile.
- **Decompiler** – The compiler that translates from a low-level language to a higher level one.
- **Transcompiler** – The compiler that translates high level languages.

Lecture 1: Introduction

Top most Compilers used according to the Computer Languages –

- ❖ C– Turbo C, Tiny C Compiler, GCC, Clang, Portable C Compiler .
- ❖ C++ -GCC, Clang, Dev C++, Intel C++, Code Block.
- ❖ JAVA– IntelliJ IDEA, Eclipse IDE, NetBeans, BlueJ, JDeveloper .
- ❖ Kotlin– IntelliJ IDEA, Eclipse IDE
- ❖ Python– CPython, JPython, Wing, Spyder
- ❖ JavaScript– WebStorm, Atom IDE, Visual Studio Code, Komodo Edit.

4. Introduction to Interpreters:

All high-level languages need to be converted to machine code so that the computer can understand the program after taking the required inputs. The software by which the conversion of the high-level instructions is performed line-by-line to machine-level language, other than compiler and assembler, is known as INTERPRETER .

The interpreter in the compiler checks the source code line-by-line and if an error is found on any line, it stops the execution until the error is resolved. Error correction is quite easy for the interpreter as the interpreter provides a line-by-line error. But the program takes more time to complete the execution successfully. Interpreters were first used in 1952 to ease programming within the limitations of computers at the time. It translates source code into some efficient intermediate representation and executes them immediately.



Lecture 1: Introduction**Top Interpreters According to the Computer Languages**

- Python- CPython, PyPy, Stackless Python, IronPython
- Ruby- YARV, Ruby MRI (CRuby)
- JAVA- HotSpot, OpenJ9, JRockIt
- Kotlin- JariKo.

5. Object file

An object file is a file that contains machine code or bytecode, as well as other data and metadata, generated by a compiler or assembler from source code during the compilation or assembly process. The machine code that is generated is known as object code.

The object code is usually relocatable, and not usually directly executable. There are various formats for object files, and the same machine code can be packaged in different object file formats. An object file may also work like a shared library.

6. Executable file

An executable file (EXE file) is a computer file that contains an encoded sequence of instructions that the system can execute directly when the user clicks the file icon. Executable files commonly have an EXE file extension, but there are hundreds of other executable file formats.

Some executable file types can run on any compatible system without requiring the existence of another program. These files are considered to pose a high security risk. They include EXE, BAT, COM, CMD, INF, IPA, OSX, PIF, RUN and WSH.