Learning Objectives

In this chapter you will learn about:

- Term “Software” and its relationship with “Hardware”
- Various types of software and their examples
- Relationship among hardware, system software, application software, and users of a computer system
- Different ways of acquiring software
- Various steps involved in software development
- Firmware
- BIOS
- Malware

Software

- ***Hardware*** refers to the physical devices of a computer system.
- ***Software*** refers to a collection of programs
- ***Program*** is a sequence of instructions written in a language that can be understood by a computer
- ***Software package*** is a group of programs that solve a specific problem or perform a specific type of job

Relationship Between Hardware and Software

- Both hardware and software are necessary for a computer to do useful job. They are complementary to each other
- Same hardware can be loaded with different software to make a computer system perform different types of jobs
- Except for *upgrades*, hardware is normally a one-time expense, whereas software is a continuing expense
- Upgrades refer to renewing or changing components like increasing the main memory, or hard disk capacities, or adding speakers, modems, etc.

Types of Software

Most software can be divided into two major categories:

- ***System software*** are designed to control the operation and extend the processing capability of a computer system
- ***Application software*** are designed to solve a specific problem or to do a specific task

System Software

- Make the operation of a computer system more effective and efficient

- Help hardware components work together and provide support for the development and execution of application software

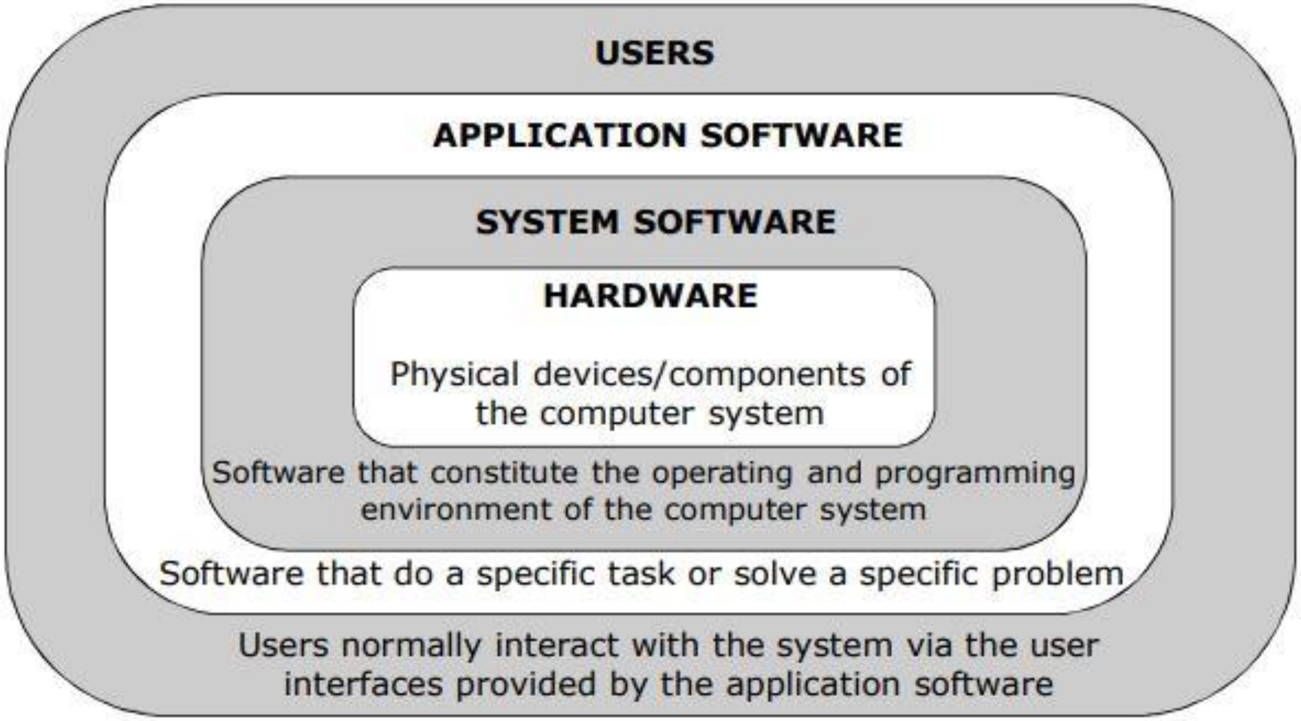
- Programs included in a system software package are called ***system programs*** and programmers who prepare them are called ***system programmers***

- Examples of system software are operating systems, programming language translators, utility programs, and communications software

Application Software

- Solve a specific problem or do a specific task
- Programs included in an application software package are called ***application programs*** and the programmers who prepare them are called ***application programmers***
- Examples of application software are word processing, inventory management, preparation of tax returns, banking, etc.

Logical System Architecture


USERS

APPLICATION SOFTWARE

SYSTEM SOFTWARE

HARDWARE

Physical devices/components of the computer system

Software that constitute the operating and programming environment of the computer system

Software that do a specific task or solve a specific problem

Users normally interact with the system via the user interfaces provided by the application software

Relationship among hardware, system software, application software, and users of a computer system.

Ways of Acquiring Software

- Buying pre-written software
- Ordering customized software
- Developing customized software
- Downloading public-domain software

Each of these ways of acquiring software has its own advantages and limitations

Software Development Steps

Developing a software and putting it to use is a complex process and involves following steps:

- Analyzing the problem at hand and planning the program(s) to solve the problem
- Coding the program(s)
- Testing, debugging, and documenting the program(s)
- Implementing the program(s)
- Evaluating and maintaining the program(s)

Firmware

- **Firmware** is data that is stored on a computer or other hardware device's ROM (read-only memory) that provides instruction on how that device should operate.
- Unlike normal software, **firmware** cannot be changed or deleted by an end-user without using special programs, and remains on that device whether it's on or off.
- Firmware technology has enabled production of various types of smart machines having microprocessor chips with embedded software.
- **Firmware** - Software that is absolutely essential to use hardware.

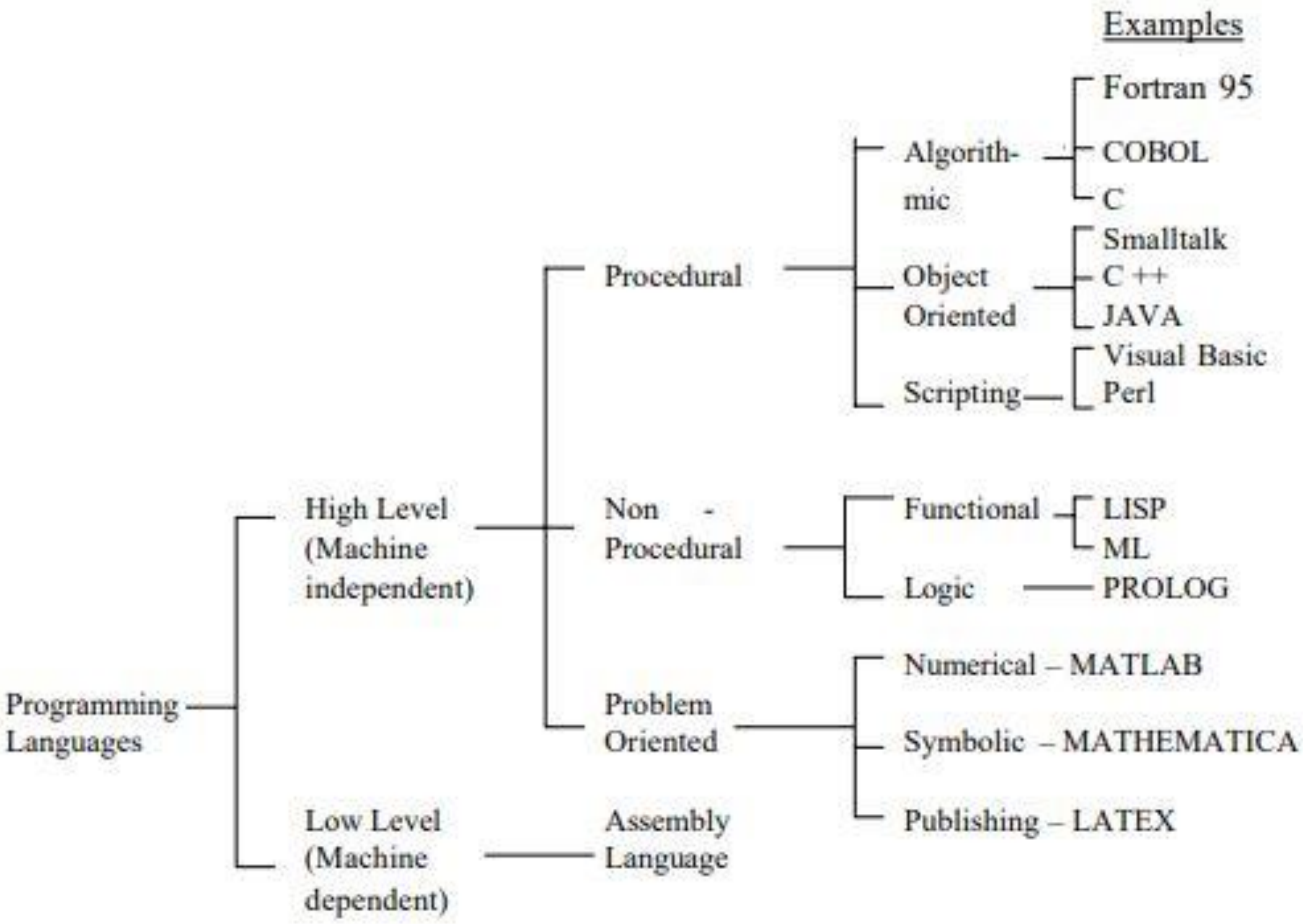
BIOS

- BIOS identifies, configures, tests and connects computer hardware to the OS immediately after a computer is turned on.
- The combination of these steps is called the *boot process*.
- These tasks are each carried out by BIOS' four main functions:
  1. Power-on self-test (POST): This tests the hardware of the computer before loading the OS.
  2. Bootstrap loader: This locates the OS.
  3. Software/drivers: This locates the software and drivers that interface with the OS once running.
  4. Complementary metal-oxide semiconductor (CMOS) setup: This is a configuration program that enable users to alter hardware and system settings. CMOS is the name of BIOS' non-volatile memory.

Malware

- **Malware** - Software which is specifically designed to damage computer.
- Malware is any software intentionally designed to cause damage to a computer, server, client, or computer network.
- A wide variety of malware types exist, including computer viruses, worms, Trojan horses, ransomware, spyware, adware, rogue software, wiper and scareware.
- For more details: https://blog.totalprosource.com/5-common-malware-types

Programming Language



Examples

Fortran 95

Algorith-
mic

COBOL

C

Smalltalk

Procedural

Object
Oriented

C ++

JAVA

Visual Basic

Scripting

Perl

High Level
(Machine
independent)

Non -
Procedural

Functional

LISP

ML

Logic

PROLOG

Numerical – MATLAB

Programming
Languages

Problem
Oriented

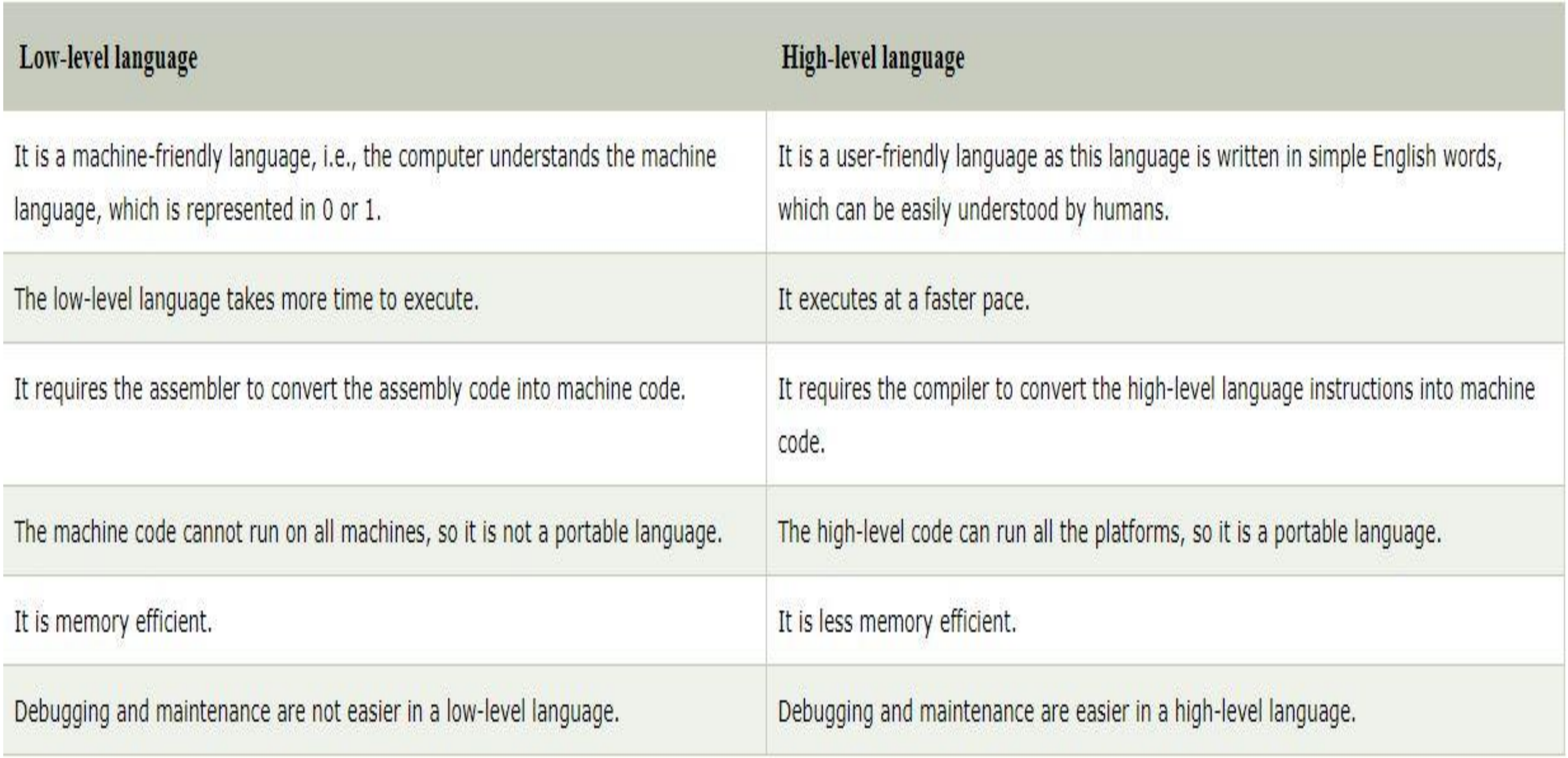
Symbolic – MATHEMATICA

Publishing – LATEX

Low Level
(Machine
dependent)

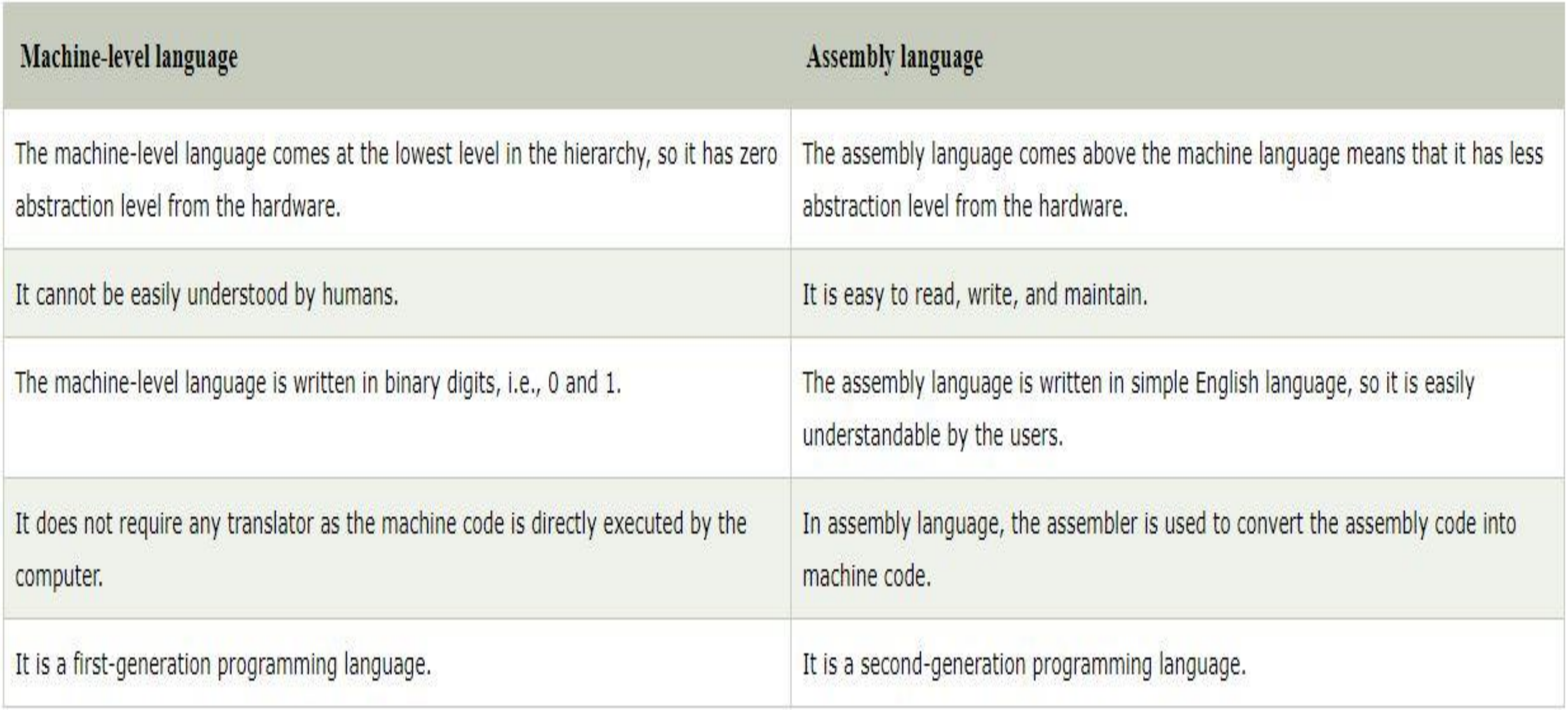
Assembly
Language

Programming 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.

Programming 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.