
HARDWARE AND SOFTWARE

Hardware: generic term given to the physical parts of a computer, both internal and external.





Software: generic term used to describe programs which can be run on hardware. The idea of software was that you could update it without having to replace the hardware. There are two main types of software, system software and application software:

Application Software

Application software refers to all the programs which a user would use on a system allowing users to complete a particular task.

System Software

System software is software which is essential to running a computer system and support the applications software. This category of software is often subdivided into four different categories:

-  Utility Programs: perform specific common tasks related to running the computer, e.g. zipping files and disk defragmenter.
-  Libraries: code, data and resources that can be called by other programs.
-  Translators: software which converts programming language instructions into machine code. There are three types of translators: compilers, assemblers and interpreters.
-  Operating Systems: collection of software designed to act as an interface between the user and the computer and manages the overall operation of the computer. Provides a link between the hardware, applications and user, hiding the complexity of the computer. It creates a so-called virtual machine.

Resource management: how an operating system manages the access of different programs running to the hardware (also called scheduling):

- *Processor*: e.g. allocate to each task a time slice
- *Memory management*: uses the heap (details of all the unallocated locations in a section of memory) and memory map (shows which blocks of memory have been allocated to each task. When programs are too big to fit in available RAM:
 - *Virtual memory*: part of the secondary storage (hard disk) is used to store code or files that would normally be stored in the RAM.
 - *Paging*: the code of large applications is loaded in pages storing on the RAM only a kernel (central block) that is used.

PROGRAMMING LANGUAGES

Low-level Languages

Low level programming languages are languages where each instruction in the language equates to one action of the CPU. The characteristics of low level languages are: usually optimised by the programmer to run as fast as possible, compact and allows direct manipulation of registers for more control. Machine code and assembly language are low-level languages:

- ✚ Machine code: lowest level code made up of 0s and 1s
- ✚ Assembly language: way of programming machine instruction using mnemonics. Assembler takes the assembly source code and converts it in machine code.

High-level languages

High level programming languages are languages which allows programs to be written using English keywords and that are platform independent. The high-level languages are: easy to understand and maintain, portable and must be translated.

High level languages can be split into groups, imperative languages, object-oriented languages, declarative languages and functional languages:

- ✚ Imperative languages: giving the computer commands or procedures to follow.
- ✚ Object-oriented languages: entered around objects, where the data and methods for the program are contained. Objects which can be grouped together into classes.
- ✚ Declarative languages: work by defining a problem to accomplish instead of how to accomplish it. E.g. logic programming whereby a programmer programs in facts and rules to compare with the data and provide an output.
- ✚ Functional programming: uses mathematical functions as the building blocks to create programs instead of sets of instructions.

Translators

Compiler

A compiler is a program that translates high-level languages into machine code by translating all the code at 'compile time'. Compilers often can optimise code so that it runs faster, however this takes time and a program must be compiled before it can be run. It is used for deployment of programs.

Assemblers

Assemblers translate assembly code into machine code, similar to a compiler. As with a compiler, code must be translated before it can be run.

Interpreter

Interpreters are a type of program which translates high-level languages into machine code, however once each statement has been translated from the high-level code, it is then run on the computer immediately. Generally interpreters are less efficient and slower than compilers however they are suitable for developers when debugging and testing applications because the code can be run step by step for debugging purposes, as well as requiring no time to compile the application in the first place.

Source Code: the original programming code that has not yet been compiled into an executable file.

Object (Executable) Code: compiled code which can be run as an executable on any computer.

Intermediate Languages

Some compilers will not compile directly into machine-code, but instead compile into an intermediate language such as bytecode, which can then subsequently be executed on any platform using a virtual machine (e.g. Java Virtual Machine). This allows source code to be translated into a format which can be run on any platform.