1.4 Machine Languages, Assembly Languages and High-Level Languages

Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation* steps. Hundreds of such languages are in use today. These may be divided into three general types:

1. Machine languages
2. Assembly languages
3. High-level languages

Machine Languages

Any computer can directly understand only its own **machine language**, defined by its hardware design. Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time. Machine languages are *machine dependent* (a particular machine language can be used on only one type of computer). Such languages are cumbersome for humans. For example, here's a section of an early machine-language payroll program that adds overtime pay to base pay and stores the result in gross pay:

```
+1300042774
+1400593419
+1200274027
```

Assembly Languages and Assemblers

Programming in machine language was simply too slow and tedious for most programmers. Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations. These abbreviations formed the basis of **assembly languages**. *Translator programs* called **assemblers** were developed to convert early assembly-language programs to machine language at computer speeds. The following section of an assembly-language payroll program also adds overtime pay to base pay and stores the result in gross pay:

`load`	`basepay`
`add`	`overpay`
`store`	`grosspay`

Although such code is clearer to humans, it’s incomprehensible to computers until translated to machine language.

High-Level Languages and Compilers

With the advent of assembly languages, computer usage increased rapidly, but programmers still had to use numerous instructions to accomplish even the simplest tasks. To speed the programming process, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks. Translator programs called **compilers** convert high-level language programs into machine language. High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical notations. A payroll program written in a high-level language might contain a *single* statement such as

```
grossPay = basePay + overTimePay
```

From the programmer's standpoint, high-level languages are preferable to machine and assembly languages. Java is the world's most widely used high-level programming language.

Interpreters

Compiling a large high-level language program into machine language can take considerable computer time. *Interpreter* programs, developed to execute high-level language programs directly, avoid the delay of compilation, although they run slower than compiled programs. We’ll say more about interpreters in Section 1.9, where you’ll learn that Java uses a clever performance-tuned mixture of compilation and interpretation to run programs.