1.2.2 Translators

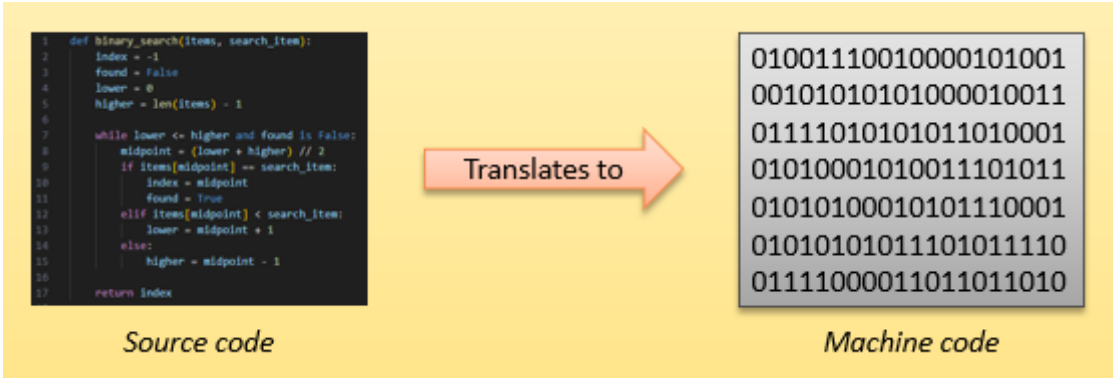
Translators

Translators turn high level programming languages (or low level languages) into binary machine code (or intermediate code).

Humans write the source code for a program, and the translator converts that into machine code that the computer can execute

Three types of translators are:

- **Compilers** – convert high-level languages to machine code
- **Interpreters** – convert high-level languages to machine code
- **Assemblers** – convert assembly code to machine code



Interpreters

An interpreter translates and executes high-level language source code, typically one statement/instruction at a time.

It translates and executes the source code sequentially, rather than producing a separate executable file containing the translated program

The program has to be translated every time the code is run.

The source code is required by the interpreter, so it must be available on the system running the program.

As the code must be translated as it runs, interpreted programs generally run more slowly than compiled programs.

The source code is portable because it can be run on different systems that have a suitable interpreter.

The program can execute until it encounters an error. Once the error is corrected, the program can be run again.

Compilers

A compiler translates a high-level language program into machine code or object code before the program is run.

The compiler translates the whole program and produces object code, which can be stored in an executable file.

The program can then be run without needing to be translated again. However, changes to the source code will require the code to be recompiled.

The source code does not need to be distributed with the executable, so users can run the program without having access to the source code.

Once compiled, the program can execute quickly because it is already in machine/object code and does not need to be translated each time it runs.

Once compiled, the object code is specific to the target processor architecture and operating environment, so the same executable may not run on a different architecture.

If errors are detected during compilation, the compiler reports the errors and does not produce a usable executable until the errors have been corrected.

Assemblers

Assemblers take the low-level language, assembly code, and translates it into machine code.

Assembly code has a one to one relationship with machine code. Each assembly instruction is equivalent to one machine code instruction.

(Assembly code is explained in Topic 1.2.4 Assembly Language)

Comparison Table

Compiler	Interpreters
Translate a high-level language into machine code by translating all the code and storing the resulting binary in an executable.	Translate a high-level language into machine code by translating and executing the code line by line.
Once compiled the code does not need translating again before execution.	The program must be translated each time it is executed so is slower than compiled code.
When the program is distributed only the executable code is distributed so the source code is not seen.	Source code must be distributed to users.
If even a small change is made to part of a program then the whole program has to be recompiled. This increases development time.	The program can run on any processor as long as that processor has an interpreter for that language. This makes development easier as code can be tested for multiple platforms.
A compiler will not compile the code if there is an error. Instead, it will attempt to list all the errors in the code. However, only the first listed error is reliable, as the other errors may also have been caused by this error.	An interpreter stops when it encounters an error, making it easier to identify where the error occurred.
The executable is generally platform-specific and must be compiled for each target platform. Every platform that needs to run the code must have a compiler written for it.	

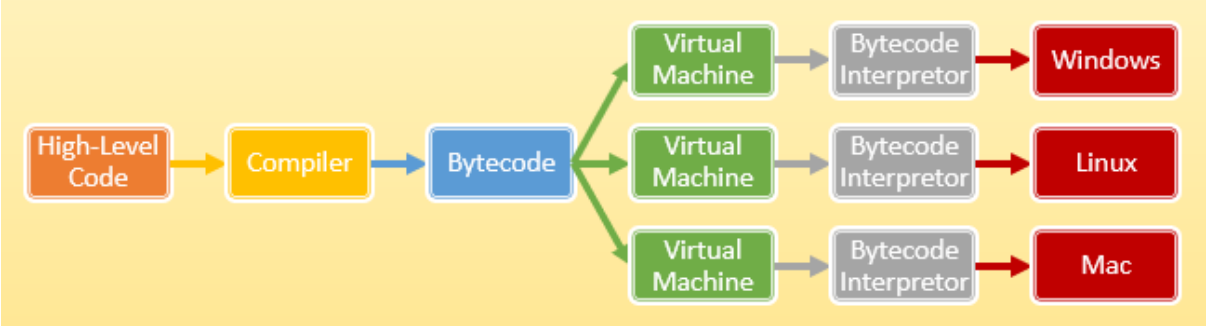
Intermediate Code

Some compilers translate source code into an intermediate language rather than producing an executable file.

Bytecode is intermediate code that can be executed using a virtual machine.

A virtual machine emulates the architecture of a computer by interpreting the bytecode for the target machine. This means that code translated into bytecode can be executed on any platform that has a suitable virtual machine.

This allows the source code to be hidden but keeps the program portable.



A virtual machine can also perform security checks on the bytecode without execution making it more secure.