ADESINA ALAMEEN B
18/SCI01/099
CSC 310

A translator or programming language processor is a generic term that can refer to anything that converts code from one computer language into another. These include translations between high-level and human-readable computer languages such as C++ and Java, intermediate-level languages such as Java bytecode, low-level languages such as the assembly language and machine code, and between similar levels of language on different computing platforms, as well as from any of the above to another.

There are 3 different types of translators as follows:

i)Compiler

A compiler is a translator used to convert high-level programming language to low-level programming language. It converts the whole program in one session and reports errors detected after the conversion. The compiler takes time to do its work as it translates high-level code to lower-level code all at once and then saves it to memory. A compiler is processor-dependent and platform-dependent. It has been addressed by alternate names as the following: special compiler, cross-compiler and, source-to-source compiler.

ii)Interpreter

The interpreter is similar to a compiler, it is a translator used to convert high-level programming language to low-level programming language. The difference is that it converts the program one line of code at a time and reports errors when detected, while also doing the conversion. An interpreter is faster than a compiler as it immediately executes the code upon

reading the code. It is often used as a debugging tool for software development as it can execute a single line of code at a time. An interpreter is also more portable than a compiler as it is processor-independent, you can work between different hardware architectures.

iii)Assembler

An assembler is a translator used to translate assembly language into machine language. It has the same function as a compiler for the assembly language but works like an interpreter. Assembly language is difficult to understand as it is a low-level programming language. An assembler translates a low-level language, such as an assembly language to an even lower-level language, such as the machine code.

Comparative analysis of Assembler, Compiler and Interpreter

Assembler	Compiler	Interpreter
Assembler converts source code written in assembly language into machine code and then that machine code is executed by a computer.	A compiler converts high-level language program code into machine language and then executes it. High-level languages are C and C#	Interpreter converts source code into the intermediate form and then converts that intermediate code into machine language. The intermediate code looks same as assembler code.

Assembler converts assembly language to machine language at once.	Complier scans the entire program first before translating into machine code.	Interpreter scans and translates the program line by line to equivalent machine code.
It converts a source code to an object first then it converts the object code to machine language with the linker programs.	Compiler takes entire program as input.	Interpreter takes single instruction as input.
Input to the assembler is assembly language code.	Intermediate object code is generated in case of compiler.	In case of interpreter, No intermediate object code is generated.
Assembler takes the most amount of execution time of the three	Compiler takes less execution time when compared to interpreter.	Interpreter takes more execution time when compared to compiler.
GAS, GNU is an example of an assembler.	Examples include C, COBOL, C#, C++, etc	Examples include Python, Perl, VB, PostScript, LISP etc

IMPORTANCE OF HIGH LEVEL LANGUAGE

The main advantage of high-level languages over low-level languages is that they are easier to read, write, and maintain. Ultimately, programs written in a high-level language must be translated into machine language by a compiler or interpreter. The first high-level programming languages were designed in the 1950s. Now there are dozens of different languages, including Ada, Algol, BASIC, COBOL, C, C++, FORTRAN, LISP, Pascal, and Prolog.