Assembler

An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer.

An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components.

An assembler is sometimes referred to as the compiler of assembly language. It also provides the services of an interpreter.

An assembler primarily serves as the bridge between symbolically coded instructions written in assembly language and the computer processor, memory and other computational components. An assembler works by assembling and converting the source code of assembly language into object code or an object file that constitutes a stream of zeros and ones of machine code, which are directly executable by the processor.

Assemblers are classified based on the number of times it takes them to read the source code before translating it; there are both single-pass and multi-pass assemblers. Moreover, some high-end assemblers provide enhanced functionality by enabling the use of control statements, data abstraction services and providing support for object-oriented programming structures.

***Compiler***

A compiler is a software program that transforms high-level source code that is written by a developer in a high-level programming language into a low level object code (binary code) in machine language, which can be understood by the processor. The process of converting high-level programming into machine language is known as compilation.

The processor executes object code, which indicates when binary high and low signals are required in the arithmetic logic unit of the processor.

A compiler that converts machine language into high-level natural language is called a decompiler. Compilers that produce the object code meant to run on a system are called cross-compilers. Finally, a compiler that converts one programming language into another is called a language translator.  
  
A compiler executes four major steps:

* **Scanning**: The scanner reads one character at a time from the source code and keeps track of which character is present in which line.
* **Lexical Analysis**: The compiler converts the sequence of characters that appear in the source code into a series of strings of characters (known as tokens), which are associated by a specific rule by a program called a lexical analyzer. A symbol table is used by the lexical analyzer to store the words in the source code that correspond to the token generated.
* **Syntactic Analysis**: In this step, syntax analysis is performed, which involves preprocessing to determine whether the tokens created during lexical analysis are in proper order as per their usage. The correct order of a set of keywords, which can yield a desired result, is called syntax. The compiler has to check the source code to ensure syntactic accuracy.
* **Semantic Analysis**: This step is comprised of several intermediate steps. First, the structure of tokens is checked, along with their order with respect to the grammar in a given language. The meaning of the token structure is interpreted by the parser and analyzer to finally generate an intermediate code, called object code. The object code includes instructions that represent the processor action for a corresponding token when encountered in the program. Finally, the entire code is parsed and interpreted to check if any optimizations are possible. Once optimizations can be performed, the appropriate modified tokens are inserted in the object code to generate the final object code, which is saved inside a file.

Macro Porcessor

A macro processor is a program that reads a file (or files) and scans them for certain keywords. When a keyword is found, it is replaced by some text. The keyword/text combination is called a macro.