Language Processors –

Compilers, interpreters, translate programs written in high-level languages into machine code that a computer understands. And assemblers translate programs written in low-level or assembly language into machine code.

The language processors can be any of the following three types:

1. Compiler :
The language processor that reads the complete source program written in high-level language as a whole in one go and translates it into an equivalent program in machine language is called a Compiler.
Example: C, C++, C#, Java.

In a compiler, the source code is translated

to object code successfully if it is free of errors. The compiler specifies the errors at the end of the compilation with line numbers when there are any errors in the source code. The errors must be removed before the compiler can successfully recompile the source code again

2. Assembler :

The Assembler is used to translate the program written in Assembly language into machine code. The source program is an input of an assembler that contains assembly language instructions. The output generated by the assembler is the object code or machine code understandable by the computer. Assembler is basically the 1st interface that is able to communicate humans with the machine. We need an Assembler to fill the gap between human and machine so

that they can communicate with each other. code written in assembly language is some sort of mnemonics(instructions) like ADD, MUL, MUX, SUB, DIV, MOV and so on. and the assembler is basically able to convert these mnemonics in Binary code. Here, these mnemonics also depend upon the architecture of the machine.

For example, the architecture of intel 8085 and intel 8086 are different.

3. Interpreter :
The translation of a single statement of the source program into machine code is done by a language processor and executes immediately before moving on to the next line is called an interpreter. If there is an error in the statement, the interpreter terminates its translating

process at that statement and displays an error message. The interpreter moves on to the next line for execution only after the removal of the error. An Interpreter directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code.

Example: Perl, Python and Matlab.

Difference between Compiler and Interpreter –

1.A compiler is a program that converts the entire source code of a programming language into executable machine code for a CPU.
An interpreter takes a source program and runs it line by line, translating each line as

it comes to it
2.The compiler takes a large amount of time to analyze the entire source code but the overall execution time of the program is comparatively faster.
An interpreter takes less amount of time to analyze the source code but the overall execution time of the program is slower.

3.The compiler generates the error message only after scanning the whole program, so debugging is comparatively hard as the error can be present anywhere in the program.
Interpreter's debugging is easier as it continues translating the program until the error is met.
4. The compiler requires a lot of memory for generating object codes.
Interpreter requires less memory than a compiler because no object code is

generated.
5. Compiler generates intermediate object code.
No intermediate object code is generated.
6. For Security purpose compiler is more useful.
The interpreter is a little vulnerable in case of security.
7.Examples of programming languages that uses compiler : C, C++,
Java Examples of programming languages that uses interpreter: Python, Perl, JavaScript, Ruby

Difference between Assembler and Interpreter :

1.Assembler converts low-level language to the machine language.
Interpreter converts high-level language to the machine language.

2.The program for an Assembler is written for particular hardware.
The program for an Interpreter is written for particular language.
3.Assembler is one to one i.e. one instruction translates to only one instruction.
Interpreter is one to many i.e. one instruction translates to many instruction.
4.Assembler translates entire program before running.
Interpreter translates program instructions line by line.
5.Errors are displayed before program is running by assembler.
Errors are displayed for the every interpreted instruction (if any).
6.Assembler is used only one time to create an executable file.
Interpreter is used everytime when the program is running.

7.Requirement of memory is less for assembler.
Requirement of memory is more for interpreter.
8.Programming language that assembler convert is Assembly language.
Programming language that interpreter convert are PHP, Python, Perl, Ruby.

Difference between Compiler and Assembler:

Compiler converts the source code written by the programmer to a machine level language.
Assembler converts the assembly code into the machine code.
Compiler input source code.
Assembler input assembly language code.
Compiler converts the whole code into machine language at a time.

But the Assembler can’t do this at once.

Compiler takes less execution time compared to an assembler.
Assemblet takes more time than a compiler.
Compiler shows the whole program error after the whole program is scanned.
Assemblet detects errors in the first phase, fixes them, and then the second phase is start to execute.
A Compiler is more intelligent than an Assembler.
But, an Assembler is less intelligent than a Compiler.
The compilation phases are lexical analyzer, syntax analyzer, semantic analyzer, intermediate code generated, a code optimizer, code generator, and error handler
Assembler makes two phases over the

given input, first phase and the second phase.
The output of compiler is a mnemonic version of machine code.
The output of assembler is binary code.
C, C++, Java, and C# are examples of compiled languages.
GAS, GNU is an example of an assembler.