**SOLUTION**

1. A translator is a programming language processor that converts a [computer program from one language](https://teachcomputerscience.com/programming-languages/) to another.  It takes a program written in source code and converts it into machine code. It discovers and identifies the error during translation.There are 3 different types of translators as follows: Compiler, Assembler, Interpreter.
2.

|  |  |
| --- | --- |
| **INTERPRETER** | **COMPILER** |
| Translate programs one statement at a time. | Scans the entire program and translates it as a whole into machine code. |
| It takes less amount of time to analyze the source code but the overall execution time is slower. | It take a larger amount of time to analyze the source code but the overall execution time is comparatively faster. |
| No intermediate code is generated, hence are memory efficient. | Generates intermediate object code which further requires linking, hence requires more memory.  |
| Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy. | It generates the error message only after scanning the whole program.Hence debugging is comparatively harder. |
| Programming languages like Python, Ruby use interpreters. | Programming languages like C, C++, Java use compilers. |
| No saving of machine code at all. | Stores machine language as machine code on the disk. |
| It is based on interpretation method. | It is based on language translation linking-loading model. |
| Do not generate output program.So they evaluate the source program at every execution  | Generates output program as .exe files which can be run independently from the original program |
| It is best suited for the program and development environment | It is bested suited for the production environment |

|  |  |  |
| --- | --- | --- |
| **Basis of Comparison** | **COMPILER** | **ASSEMBLER** |
| Basic | Generates the assembly language code or directly the executable code. | Generates the relocatable machine code. |
| Phases/ Passes | The compilation phases are lexical analyzer, syntax analyzer, semantic analyzer, intermediate code generation, code optimization, code generation. | Assembler makes two passes over the given input. |
| Input | Preprocessed source code. | Assembly language code. |
| Output | The assembly code generated by the compiler is a mnemonic version of machine code. | The relocatable machine code generated by an assembler is represented by binary code. |
| Conversion type | Compiler checks and converts the complete code at one time. | Assembler generally does not convert complete code at one time. |
| Examples | C, C++ , Java compilers | GAS, GNU assemblers. |

|  |  |
| --- | --- |
| **INTERPRETER** | **ASSEMBLER** |
| It can be used in other architectures. | It is machine dependent, cannot be used in other architecture.  |
| Program can be run before it is completed so you get partial results immediately. So any changes made are easily integrated. | A small change in design can invalidate the whole program. |
| You can work on small parts of the program and link them later into a whole program.  | It is difficult to maintain.  |
| Less efficiency, due to the fact that the language translated is not close to machine code. | Efficiency in execution just like machine level language. |

SIMILARITIES:

1. They are all translators.
2. The all convert to machine language.
3. They all have a form of symbol table to understand the commands given and translate it.
4. High level language is needed because they make the creation and tailoring of computer programs easier for the program. The it also allows for re-usability of programs writing programs writing in low level languages are writing for one purpose only and cannot be used for solving different problems or similar problems, many a time when write a program in low-level language came with alot of bugs and sometimes it didn’t even work, in this case the programmer has to keep trying and trying until he gets close to the results he/she wanted. But with the help of high level language we can create our programs more easily with fewer bugs and we are able to programming that can mimic real scenarios and find problems to them with high level language we can do more complex arithmetics and computation with ease. And also it use english language making it easier to understand.