INTRODUCTION TO COMPILERS

INTRODUCTION TO LANGUAGE PROCESSING:

As Computers became inevitable and indigenous part of human life, and several languages with different and more advanced features are evolved into this stream to satisfy or comfort the user in communicating with the machine , the development of the translators or mediator Software's have become essential to fill the huge gap between the human and machine understanding. This process is called Language Processing to reflect the goal and intent of the process. On the way to this process to understand it in a better way, we have to be familiar with some key terms and concepts explained in following lines.

LANGUAGE TRANSLATORS :

Is a computer program which translates a program written in one (Source) language to its equivalent program in other [Target]language. The Source program is a high level language where as the Target language can be any thing from the machine language of a target machine (between Microprocessor to Supercomputer) to another high level language program.

➢ Two commonly Used Translators are Compiler and Interpreter

1. **Compiler :** Compiler is a program, reads program in one language called Source Language and translates in to its equivalent program in another Language called Target Language, in addition to this its presents the error information to the User.
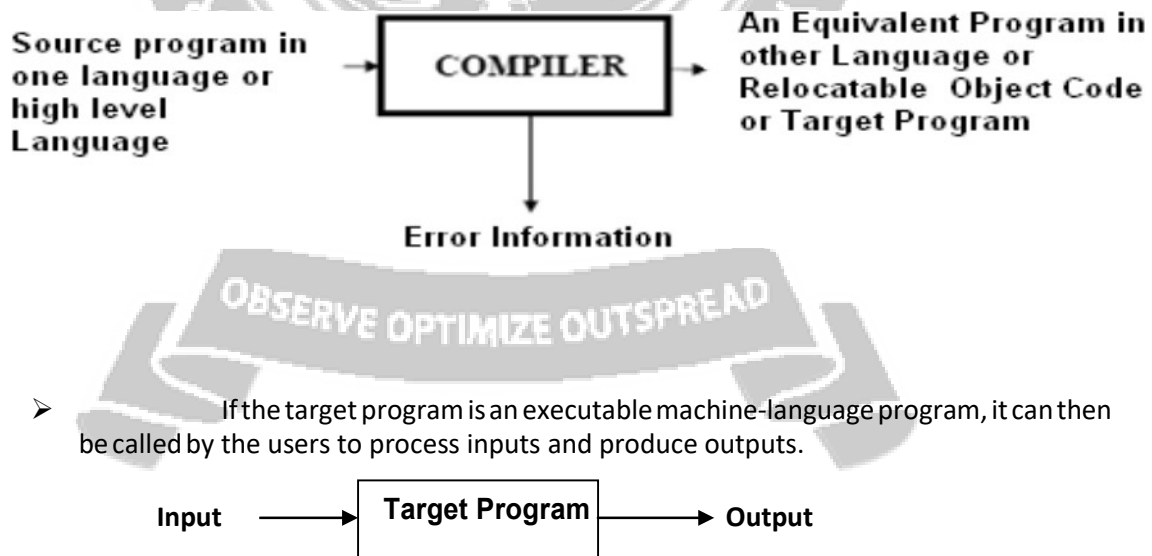

➢ If the target program is an executable machine-language program, it can then be called by the users to process inputs and produce outputs.

Figure: Running the target Program

2. **Interpreter:** An interpreter is another commonly used language processor. Instead of producing a target program as a single translation unit, an interpreter appears to directly execute the operations specified in the source program on inputs supplied by theuser.

Figure : Running the target Program

LANGUAGE PROCESSING SYSTEM:

Based on the input the translator takes and the output it produces, a language translator can be called as any one of the following.

Preprocessor: A preprocessor takes the skeletal source program as input and produces an extended version of it, which is the resultant of expanding the Macros, manifest constants if any, and including header files etc in the source file. For example, the C preprocessor is a macro processor that is used automatically by the C compiler to transform our source before actual compilation. Over and above a preprocessor performs the following activities:

- Collects all the modules, files in case if the source program is divided into different modules stored at different files.

- Expands short hands / macros into source languagestatements.

Compiler: Is a translator that takes as input a source program written in high level language and converts it into its equivalent target program in machine language. In addition to above the compiler also

- Reports to its user the presence of errors in the source program.

- Facilitates the user in rectifying the errors, and execute the code.

Assembler: Is a program that takes as input an assembly language program and converts it into its equivalent machine language code.

Loader / Linker: This is a program that takes as input a relocatable code and collects the library functions, relocatable object files, and produces its equivalent absolute machine code.

Specifically,

- **Loading** consists of taking the relocatable machine code, altering the relocatable addresses, and placing the altered instructions and data in memory at the proper locations.

- **Linking** allows us to make a single program from several files of relocatable machine code. These files may have been result of several different compilations, one or more may be library routines provided by the system available to any program that needs them.

In addition to these translators, programs like interpreters, text formatters etc., may be used in language processing system. To translate a program in a high level language program to an executable one, the Compiler performs by default the compile and linking functions.

Normally the steps in a language processing system includes Preprocessing the skeletal Source program which produces an extended or expanded source program or a ready to compile unit of the source program, followed by compiling the resultant, then linking / loading , and finally its equivalent executable code is produced. As I said earlier not all these steps are mandatory. In some cases, the Compiler only performs this linking and loading functions implicitly.

The steps involved in a typical language processing system can be understood with following diagram.
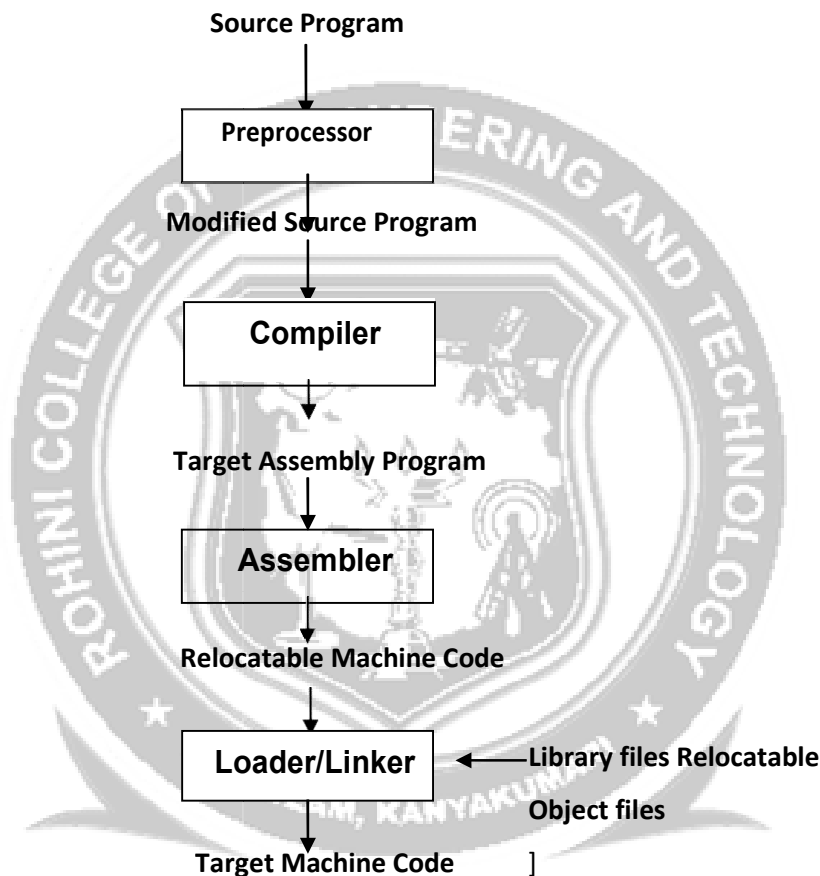
Source Program

↓

| Preprocessor |

Modified Source Program

↓

| **Compiler** |

Target Assembly Program

↓

| **Assembler** |

Relocatable Machine Code

↓

| **Loader/Linker** | ← —**Library files Relocatable**

Object files

↓

Target Machine Code   ]

Figure : Context of a Compiler in Language Processing System

TYPES OF COMPILERS:

Based on the specific input it takes and the output it produces, the Compilers can be classified into the following types;

Traditional Compilers(C, C++, Pascal): These Compilers convert a source program in a HLL into its equivalent in native machine code or object code.

Interpreters(LISP, SNOBOL, Java1.0): These Compilers first convert Source code into intermediate code, and then interprets (emulates) it to its equivalent machine code.

Cross-Compilers: These are the compilers that run on one machine and produce code for another machine.

Incremental Compilers: These compilers separate the source into user defined–steps; Compiling/recompiling step- by- step; interpreting steps in a given order

Converters (e.g. COBOL to C++): These Programs will be compiling from one high level language to another.

Just-In-Time (JIT) Compilers (Java, Micosoft.NET): These are the runtime compilers from intermediate language (byte code, MSIL) to executable code or native machine code. These perform type –based verification which makes the executable code more trustworthy

Ahead-of-Time (AOT) Compilers (e.g., .NET ngen): These are the pre-compilers to the native code for Java and .NET

Binary Compilation: These compilers will be compiling object code of one platform into object code of another platform**.**