Introduction to Compiler

Course Name: Compiler Design
Course Code: CSE331
Level:3, Term:3
Department of Computer Science and Engineering
Daffodil International University

Compiler

- Programming languages are notations for describing computations to people and to machines. The world as we know it depends on programming languages, because all the software running on all the computers was written in some programming language. But, before a program can be run, it first must be translated into a form in which it can be executed by a computer. The software systems that do this translation are called compilers.

- **A compiler is a program that can read a program in one language - the source language - and translate it into an equivalent program in another language - the target language; An important role of the compiler is to report any errors in the source program that it detects during the translation process.**

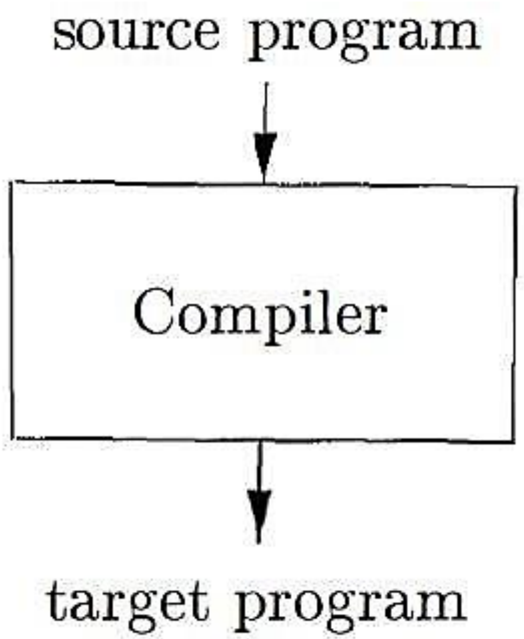


Figure 1.1: A compiler

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

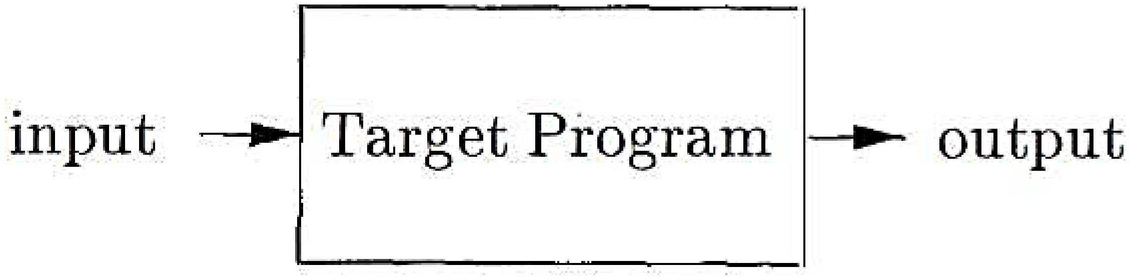


Figure 1.2: Running the target program

Interpreter

An interpreter is another common kind of language processor. Instead of producing a target program as a translation, an interpreter appears to directly execute the operations specified in the source program on inputs supplied by the user.

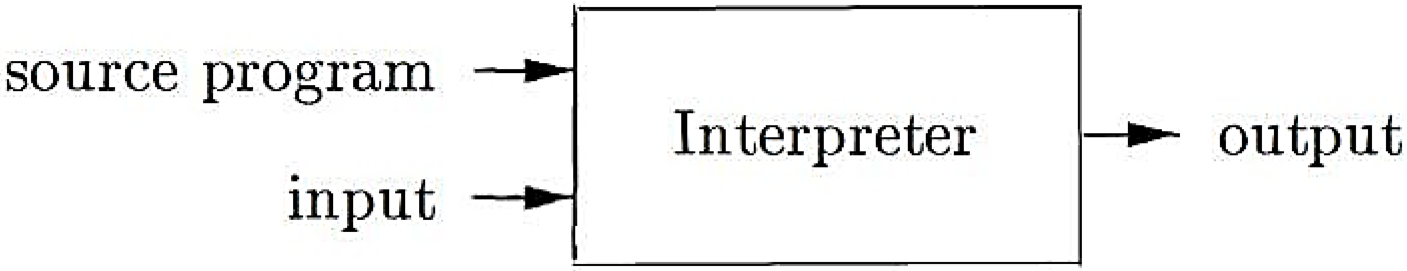


Figure 1.3: An interpreter

- The machine-language target program produced by a compiler is usually much faster than an interpreter at mapping inputs to outputs . An interpreter, however, can usually give better error diagnostics than a compiler, because it executes the source program statement by statement.

- Java language processors combine compilation and interpretation, as shown in Fig. 1.4. A Java source program may first be compiled into an intermediate form called bytecodes. The bytecodes are then interpreted by a virtual machine. A benefit of this arrangement is that bytecodes compiled on one machine can be interpreted on another machine , perhaps across a network.

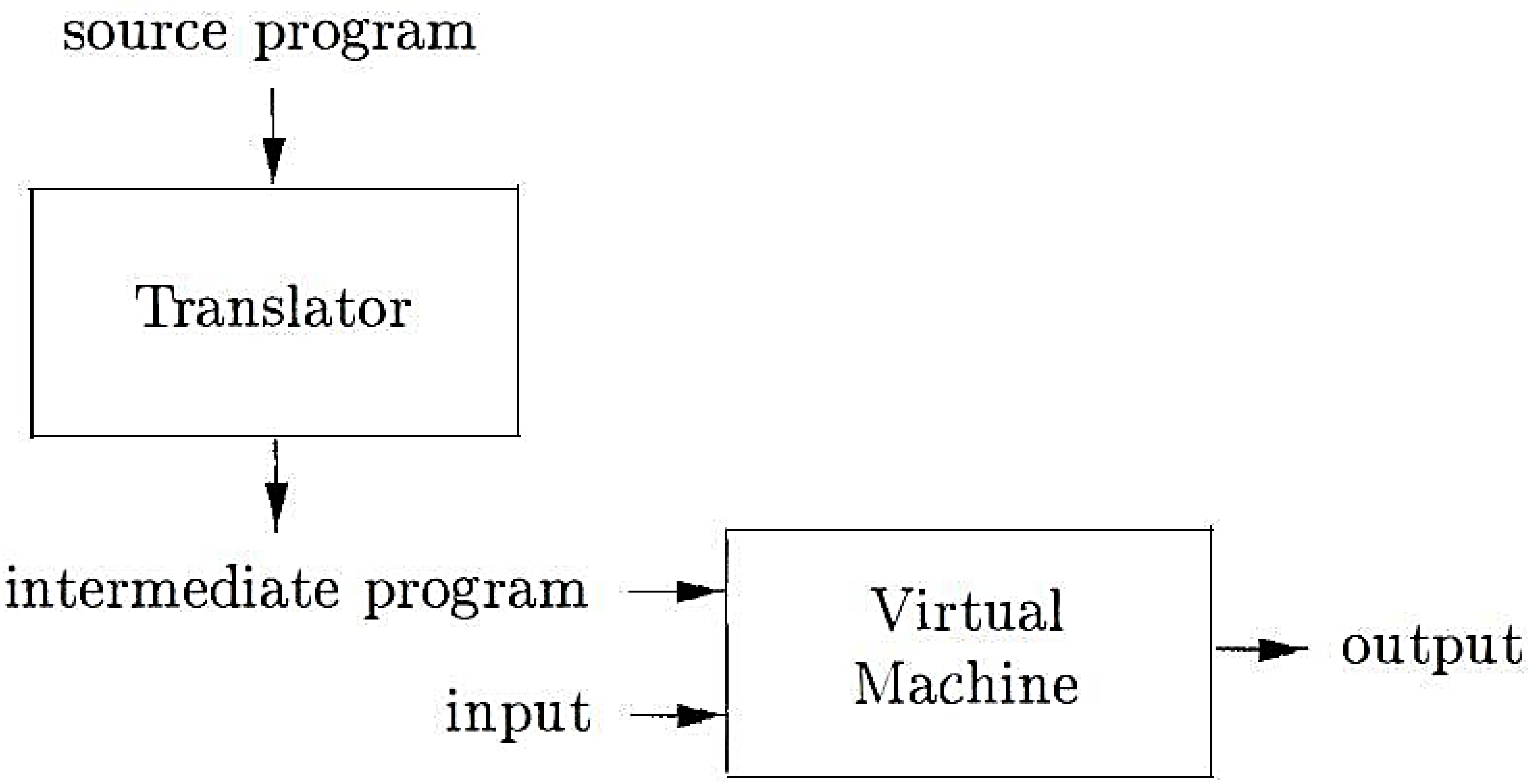


Figure 1.4: A hybrid compiler

Compiler & Interpreter

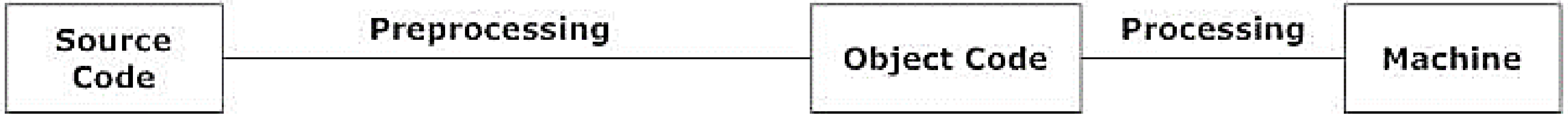


Figure: Compiler

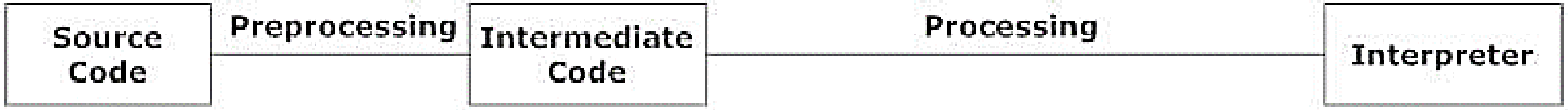


Figure: Interpreter

Difference between Interpreter and Compiler

Interpreter	Compiler
Translates program 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 takes a large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object 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 hard.
Programming languages like Python, Ruby use interpreters.	Programming languages like C, C++, Java use compilers.

Key Differences Between Compiler and Interpreter

Let’s look at major differences between Compiler and Interpreter:

1. The compiler takes a program as a whole and translates it, but interpreter translates a program statement by statement.
2. Intermediate code or target code is generated in case of a compiler. As against interpreter doesn’t create intermediate code.
3. A compiler is comparatively faster than Interpreter as the compiler take the whole program at one go whereas interpreters compile each line of code after the other.
4. The compiler requires more memory than interpreter because of the generation of object code.
5. Compiler presents all errors concurrently, and it’s difficult to detect the errors in contrast interpreter display errors of each statement one by one, and it’s easier to detect errors.
6. In compiler when an error occurs in the program, it stops its translation and after removing error whole program is translated again. On the contrary, when an error takes place in the interpreter, it prevents its translation and after removing the error, translation resumes.

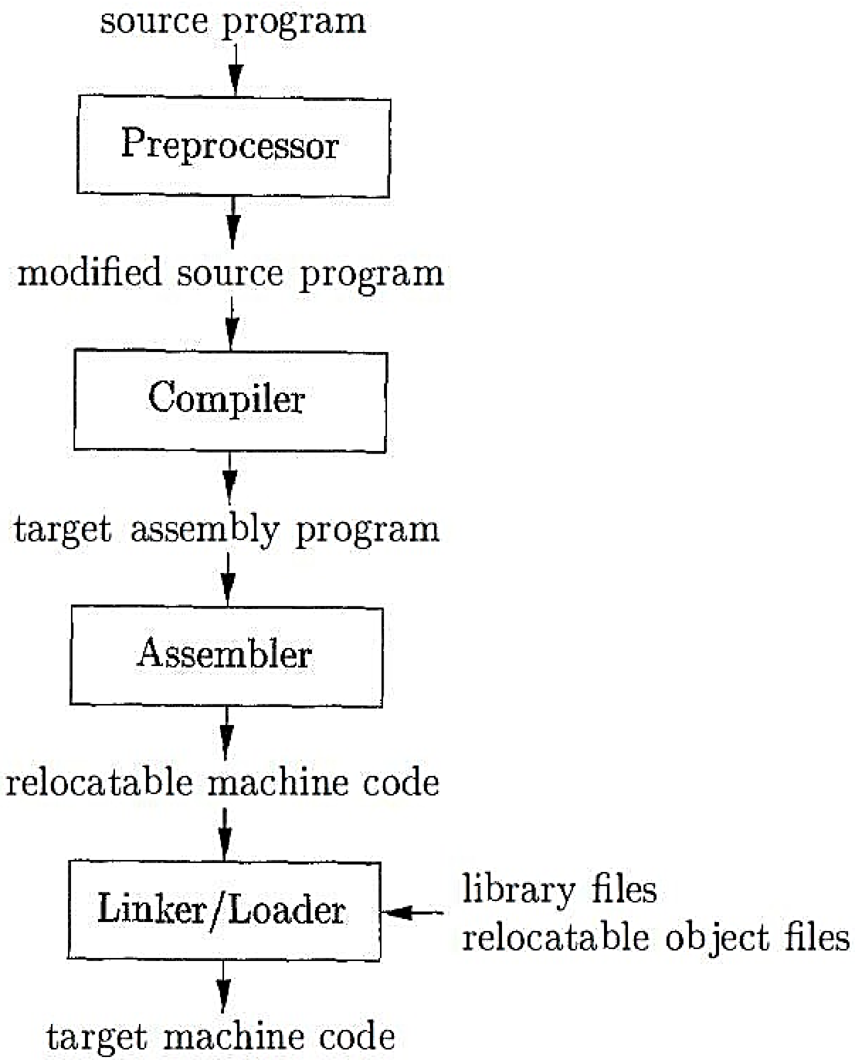
7. In a compiler, the process requires two steps in which firstly source code is translated to target program then executed. While in Interpreter It’s a one-step process in which Source code is compiled and executed at the same time.

8. The compiler is used in programming languages like C, C++, C#, Scala, etc. On the other Interpreter is employed in languages like PHP, Ruby, Python, etc.

Language Processing System

We know a computer is a logical assembly of Software and Hardware. The hardware knows a language, that is hard for us to understand, consequently we tend to write programs in high-level language, that is much less complicated for us to comprehend and maintain in thoughts. Now these programs go through a series of transformation so that they can readily be used in machines. This is where language procedure systems come handy.

Language Processing System



source program

Preprocessor

modified source program

Compiler

target assembly program

Assembler

relocatable machine code

Linker/Loader

library files
relocatable object files

target machine code

Figure 1.5: A language-processing system

- **High Level Language –** If a program contains #define or #include directives such as #include or #define it is called HLL. They are closer to humans but far from machines. These (#) tags are called pre-processor directives. They direct the pre-processor about what to do.
- **Pre-Processor –** The pre-processor removes all the #include directives by including the files called file inclusion and all the #define directives using macro expansion. It performs file inclusion, augmentation, macro-processing etc.
- **Assembly Language –** Its neither in binary form nor high level. It is an intermediate state that is a combination of machine instructions and some other useful data needed for execution.
- **Assembler –** For every platform (Hardware + OS) we will have a assembler. They are not universal since for each platform we have one. The output of assembler is called object file. Its translates assembly language to machine code.

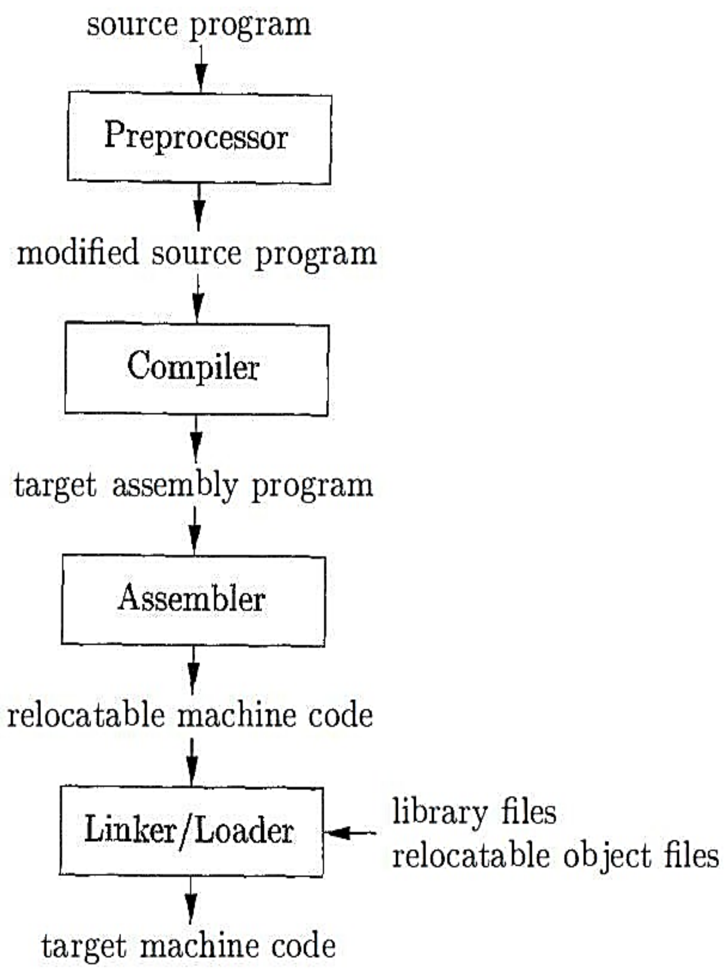


Figure 1.5: A language-processing system

- **Interpreter –** An interpreter converts high level language into low level machine language, just like a compiler. But they are different in the way they read the input. The Compiler in one go reads the inputs, does the processing and executes the source code whereas the interpreter does the same line by line. Compiler scans the entire program and translates it as a whole into machine code whereas an interpreter translates the program one statement at a time. Interpreted programs are usually slower with respect to compiled ones.
- **Relocatable Machine Code –** It can be loaded at any point and can be run. The address within the program will be in such a way that it will cooperate for the program movement.
- **Loader/Linker –** It converts the relocatable code into absolute code and tries to run the program resulting in a running program or an error message (or sometimes both can happen). Linker loads a variety of object files into a single file to make it executable. Then loader loads it in memory and executes it.

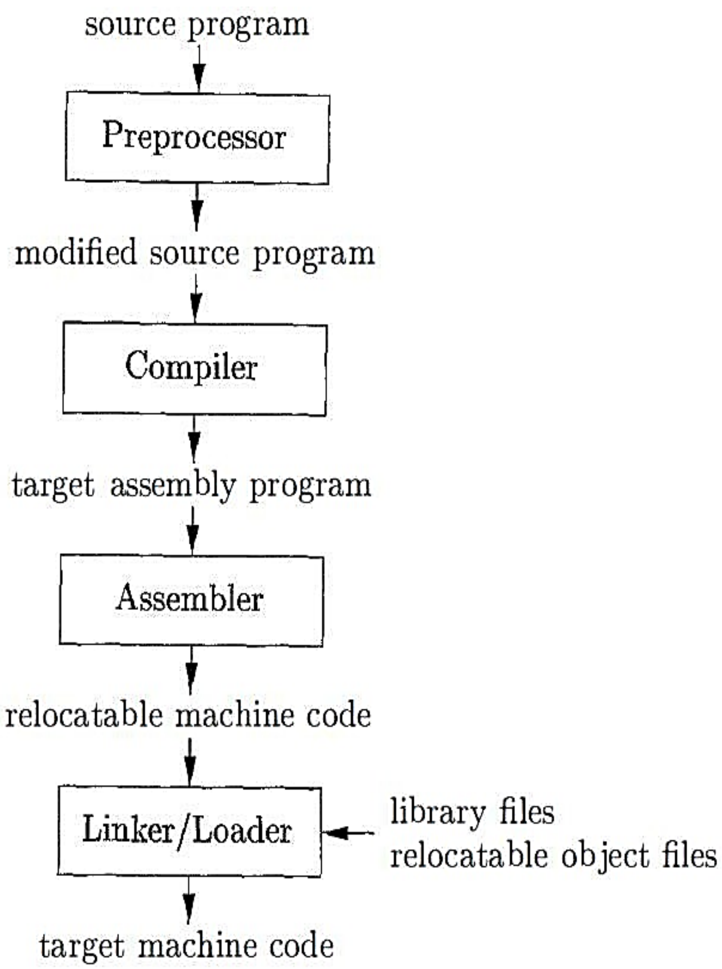


Figure 1.5: A language-processing system

THANK YOU