Programming languages, Compilers and interpreters

Programming Languages

- A programming language is a set of rules that provides a way of telling a computer what operations to perform.
- A programming language is a set of rules for communicating an algorithm
- It provides a linguistic framework for describing computations
- A programming language is a notational system for describing computation in a machine-readable and human-readable form.

Programming Languages (Contd…)

- English is a natural language. It has words, symbols and grammatical rules.
- A programming language also has words, symbols and rules of grammar.
- The grammatical rules are called syntax.
- Each programming language has a different set of syntax rules.

Level of Programming languages

High-level program

```
class Triangle {
   ...
   float surface()
      return b*h/2;
   }
```

Low-level program

```
LOAD     r1,b
LOAD     r2,h
MUL     r1,r2
DIV     r1,#2
RET
```

Executable Machine code

```
0001001001000101
0010010011101100
10101101001...
```

Types of programming languages

- First Generation Languages
- Second Generation Languages
- Third Generation Languages
- Fourth Generation Languages
- Fifth Generation Languages

First Generation Languages

- Machine language
  - Operation code – such as addition or subtraction.
  - Operands – that identify the data to be processed.
  - Machine language is machine dependent as it is the only language the computer can understand.
  - Very efficient code but very difficult to write

Second Generation Languages

- Assembly languages is a low-level language for programming computers.
  - Symbolic operation codes replaced binary operation codes.
  - Each assembly language instruction is translated into one machine language instruction.
  - Very efficient code and easier to write.

Third Generation Languages

- Closer to English but included simple mathematical notation.
  - Programs written in source code which must be translated into machine language programs called object code.
  - The translation of source code to object code is accomplished by a machine language system program called a compiler.
- Alternative to compilation is interpretation which is accomplished by a system program called an interpreter.
- Common third generation languages
  - FORTRAN
  - COBOL
  - C and C++
  - Visual Basic

Fourth Generation Languages

- A fourth generation languages (4GL) require fewer instructions to accomplish a task than a third generation language.
- Used with databases
  - Query languages
  - Report generators
  - Forms designers
  - Application generators

Fifth Generation Languages

- Declarative languages
- Functional(?): Lisp, Scheme, SML (Standard Meta Language)
  - Also called applicative
  - Everything is a function
- Logic: Prolog
  - Based on mathematical logic
  - Rule- or Constraint-based

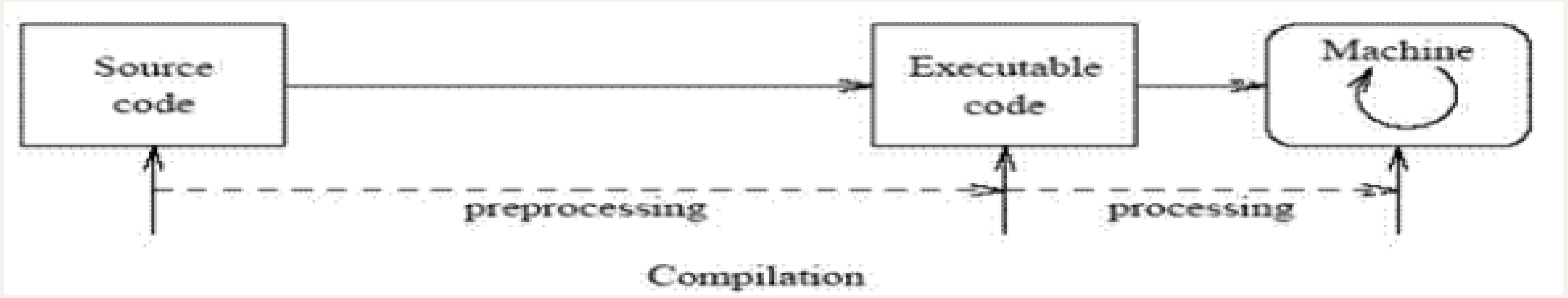
Types of Translators

- These programs convert High Level Language Source code into machine code (binary)

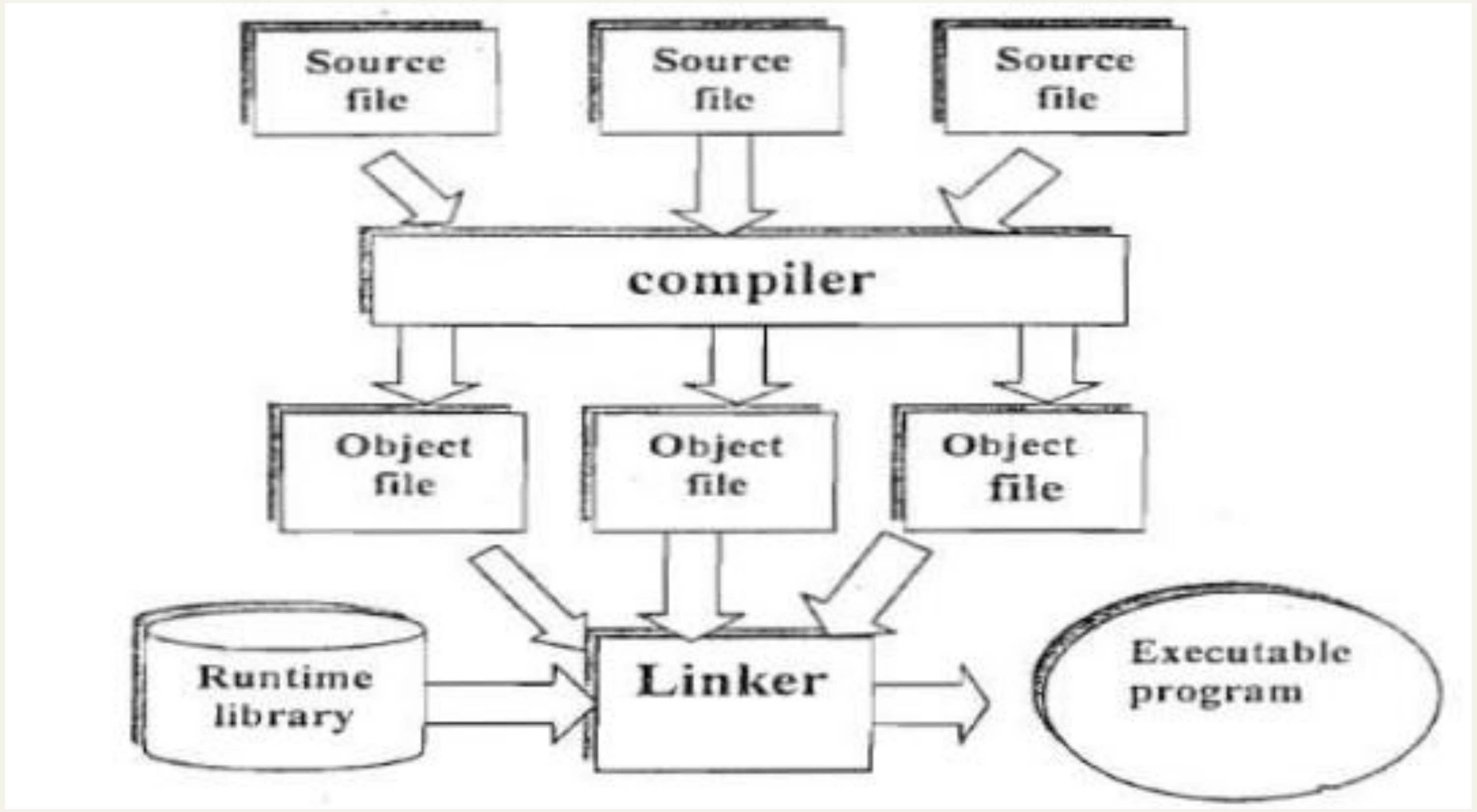
  - Interpreter
    - Translates and executes **one** line at a time
    - Always has to be re-interpreted
    - Spots errors line by line
  - Compiler
    - Translates and executes the **entire program** at once
    - This program can then be ran repeatedly

Compiler

- A compiler is a piece of code that translates the high level language into machine language.
- When a user writes a code in a high level language such as Java and wants it to execute, a specific compiler which is designed for Java is used before it will be executed.
- The compiler scans the entire program first and then translates it into machine code which will be executed by the computer processor and the corresponding tasks will be performed.

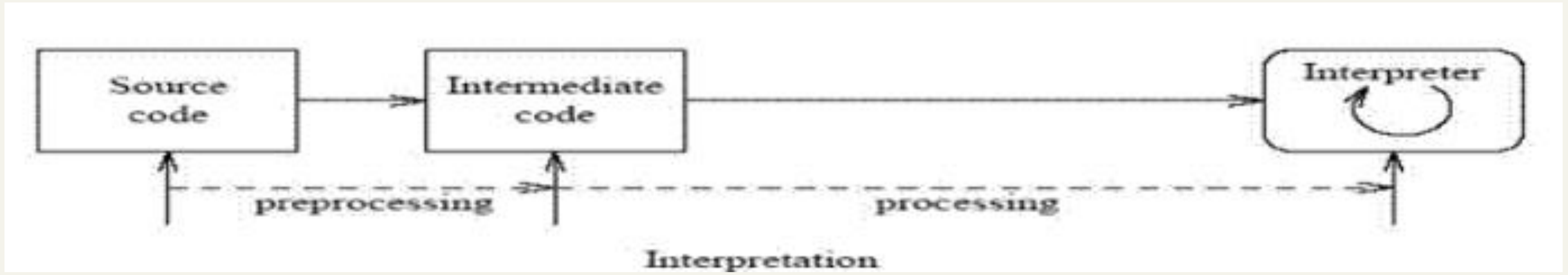


Compiler working Process



Interpreter

- Interpreters are not much different than compilers. They also convert the high level language into machine readable binary equivalents.
- Each time when an interpreter gets a high level language code to be executed, it converts the code into an intermediate code before converting it into the machine code.
- Each part of the code is interpreted and then execute separately in a sequence and if an error is found in a part of the code, it will stop the interpretation of the code without translating the next set of the codes.



Advantages of using compiler

- Since compiler converts the program to native code of the target machine (object code), faster performance can be expected.
- There is a scope for code optimization.

Advantages of using interpreter

- Process of execution can be done in a single stage. There is no need of a compilation stage.
- Alteration of codes possible during runtime.
- Facilitates interactive code development.

Compiler vs Interpreter

BASIS FOR COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, FORTRAN, Java, COBOL, Scala, etc.	PHP, Perl, Python, Ruby