Compilation & Execution

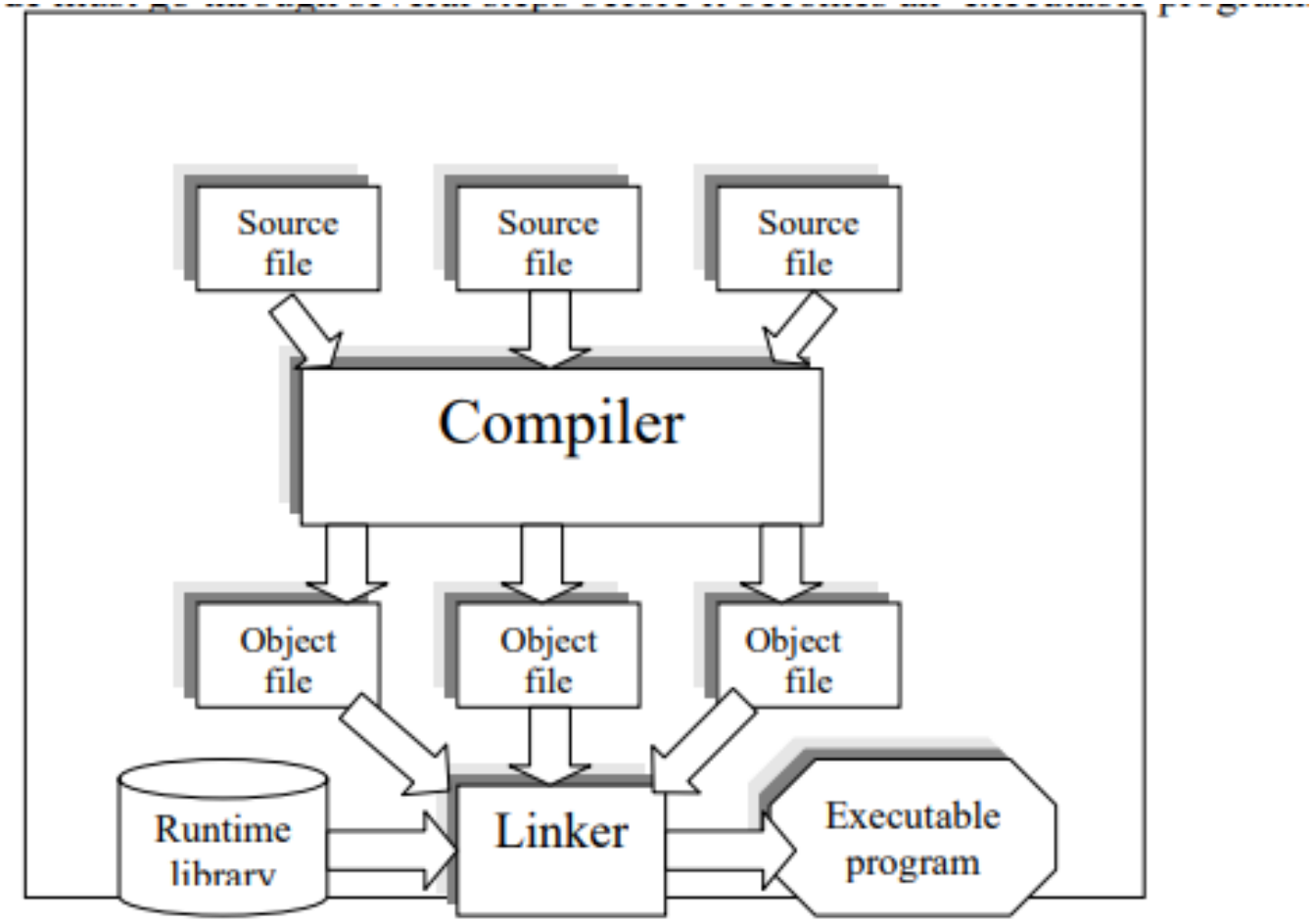
What is Compilation and Execution?

- **Compilation** is the process the computer takes to convert a **high-level programming language** into **a machine language** that the **computer** can understand.
- The software which performs this conversion is called a **compiler**.
- After compilation if everything is ok, the code is going under other process that is know.n as execution

Tasks of a Compiler

- To translate **High level language source program** to **machine codes**.
- To **trace variables** in the program.
- To **include linkage** for **subroutines**.
- To **allocate memory** for **storage** of program and variables.
- To **generate error messages**, if there are errors in the program.

Compilation Process



The process of translation from high level language (source code) to low level language (object code) is called compilation.

- The first step is to pass the source code through a compiler, which translates the high level language instructions into object code.
- The final step is producing an executable program is to pass the object code through a linker. The linker combines modules and gives real values to all symbolic addresses, there by producing machine code.
- Compilation process ends producing an executable program.
- The compiler stores the object and executable files in secondary storage.
- If there is any illegal instruction in the source code, compiler lists all the errors during compilation.

Interpreter

- The basic purpose of interpreter is same as that of complier. In compiler, the program is translated completely and directly executable version is generated. Whereas interpreter translates each instruction, executes it and then the next instruction is translated and this goes on until end of the program

No	Compiler	Interpreter
1.	Compiler takes entire program as an input.	Interpreter takes single instruction as a input.
2	Intermediate object code is generated.	No intermediate object code is generated.
3	It takes less execution time	It takes more execution time.
4	Since object code is generated more memory is required.	Memory requirement is less.
5	Program need not to be compiled every time.	Every time higher level program is converted to lower level program.
6	Errors are displayed after entire program is checked.	Errors are displayed for every instruction interpreted (if any).
7	E.g.:- C Compiler	E.g.:- BASIC

Sr. No.	Key	Compiler	Assembler
1	Operation	Compiler translates high level programming language code to machine level code.	Assembler converts the assembly level language to machine level code.
2	Input	Source code in high level programming language.	Assembly level code as input.
3	Conversion type	Compiler checks and converts the complete code at one time.	Assembler generally does not convert complete code at one time.

Debugging and Testing

- It is the process of detecting and removing errors in a program, so that the program produces the desired results on all occasions

Errors

- 1. Runtime Errors
- 2. Compile Errors : Syntax Errors, Semantic Errors
- 3. Logical Errors
- 4. Latent Errors

Debugging: It is the process of isolating and correcting different type of errors. Different debugging techniques are given below.

- 1. Error Isolation
- 2. Tracing
- 3. Watch Values
- 4. Breakpoints
- 5. Stepping

Testing

- It is the process of executing a program or system with the intent of finding errors. It involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.
- To improve quality.
- For verification and validation.
- For reliability estimation.

Difference between Testing and Debugging

Testing	Debugging
The purpose of testing is to find bugs and errors.	The purpose of debugging is to correct those bugs found during testing.
Testing is done by tester.	Debugging is done by programmer or developer.
It can be automated.	It can't be automated.
It can be done by outsider like client.	It must be done only by insider i.e. programmer.
Most of the testing can be done without design knowledge.	Debugging can't be done without proper design knowledge.