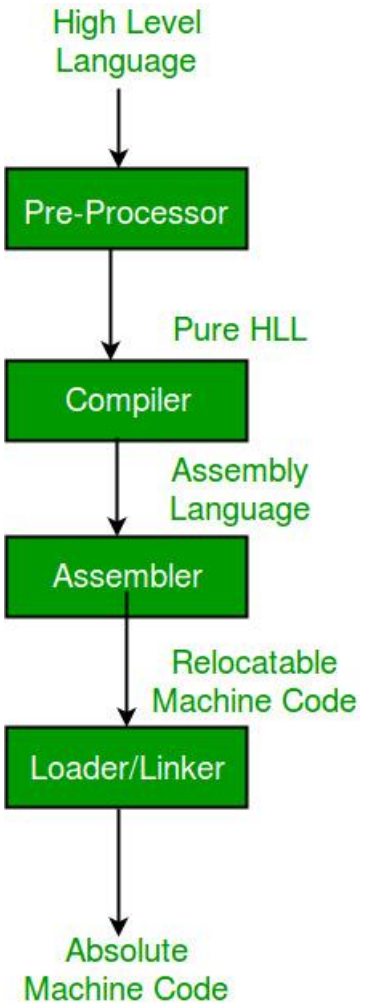
❑ Introduction to Compiler Design Course

Language Processing Systems

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

- **High-Level Language:** If a program contains pre-processor directives such as #include or #define it is called HLL.
- **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:** It's 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:** The output of the assembler is called an object file. It translates assembly language to machine code.
- **Compiler:** The compiler is an intelligent program as compared to an assembler. The compiler verifies all types of limits, ranges, errors, etc.
- **Interpreter:** An interpreter converts high-level language into low-level machine language, just like a compiler.
- **Relocatable Machine Code:** It can be loaded at any point and can be run.
- **Loader/Linker:** Loader/Linker converts the relocatable code into absolute code and tries to run the program resulting in a running program or an error message. Linker loads a variety of object files into a single file to make it executable. Then loader loads it in memory and executes it.



❑ What Is a Compiler?

- ✓ A compiler is a translator that converts the high-level language into the machine language.
- ✓ High-level language is written by a developer and machine language can be understood by the processor.
- ✓ Compiler is used to show errors to the programmer.
- ✓ The main purpose of compiler is to change the code written in one language without changing the meaning of the program.
- ✓ When you execute a program which is written in HLL programming language then it executes into two parts.
- ✓ In the first part, the source program compiled and translated into the object program (low level language).
- ✓ In the second part, object program translated into the target program through the assembler.

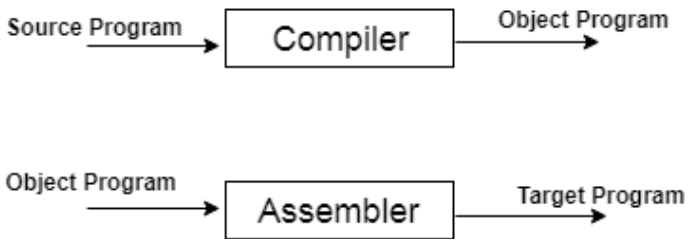
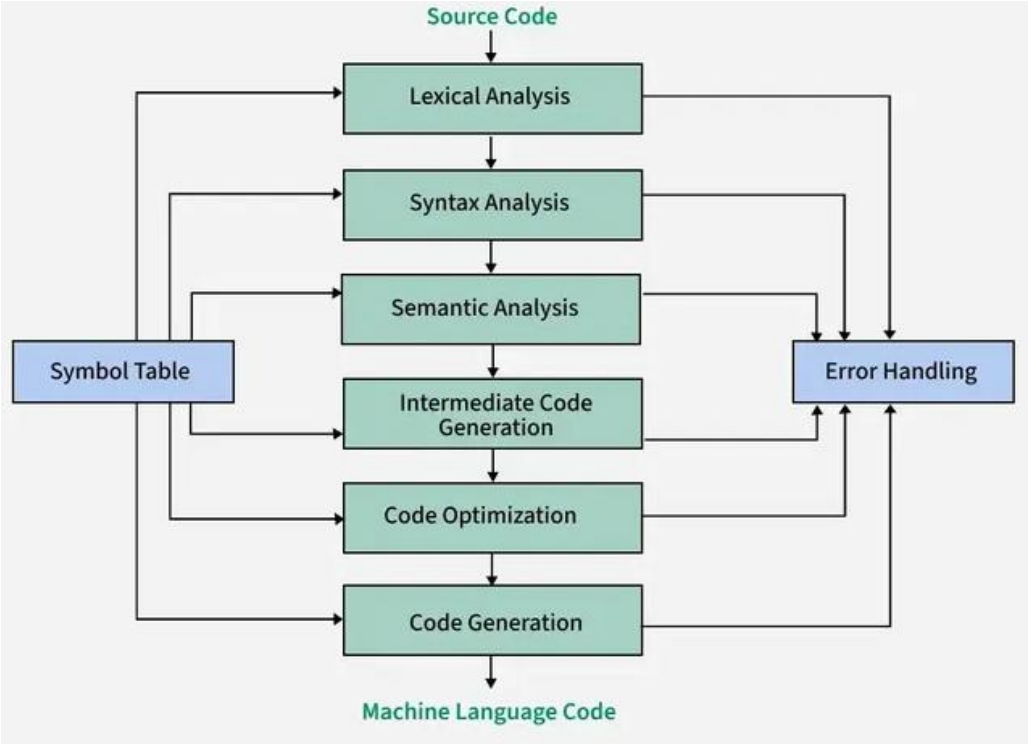


Fig: Execution process of source program in Compiler

❑ What Are the Phases of a Compiler?

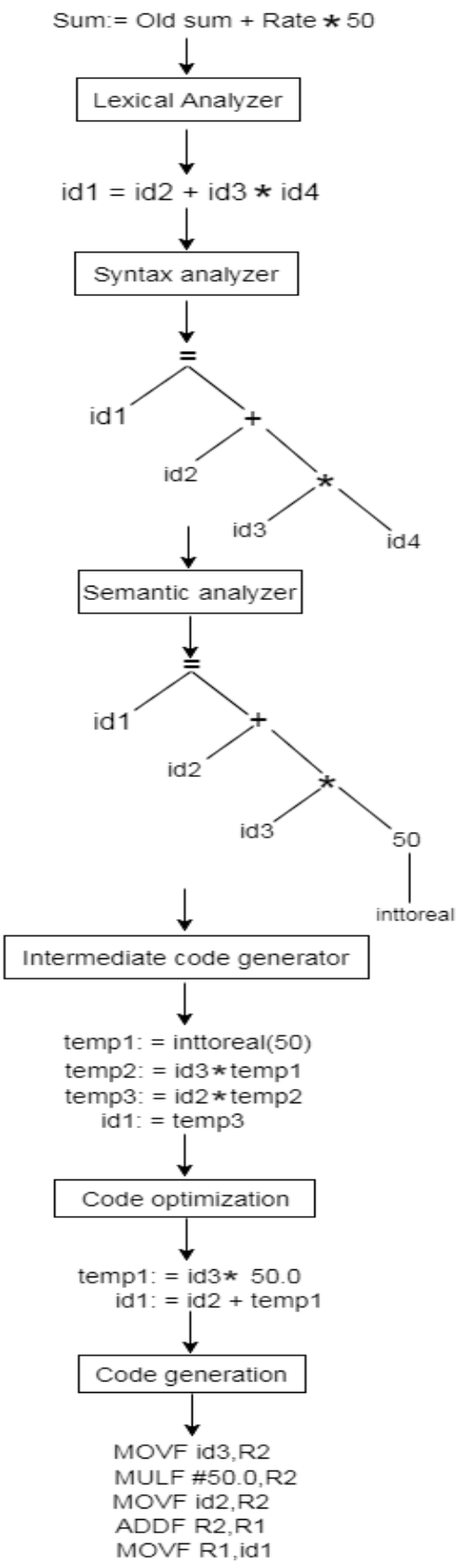
A compiler works in multiple stages, each performing a specific task to transform source code into machine code. Here are the six main phases of a compiler:

- Lexical Analysis– Breaks the source code into meaningful tokens.
- Syntax Analysis (Parsing)– Checks the structure of the code according to grammar rules.
- Semantic Analysis– Ensures logical correctness and detects type errors.
- Intermediate Code Generation– Converts code into an intermediate representation (IR).
- Code Optimization– Improves efficiency by refining the IR.
- Code Generation– Translates the optimized IR into final machine code.

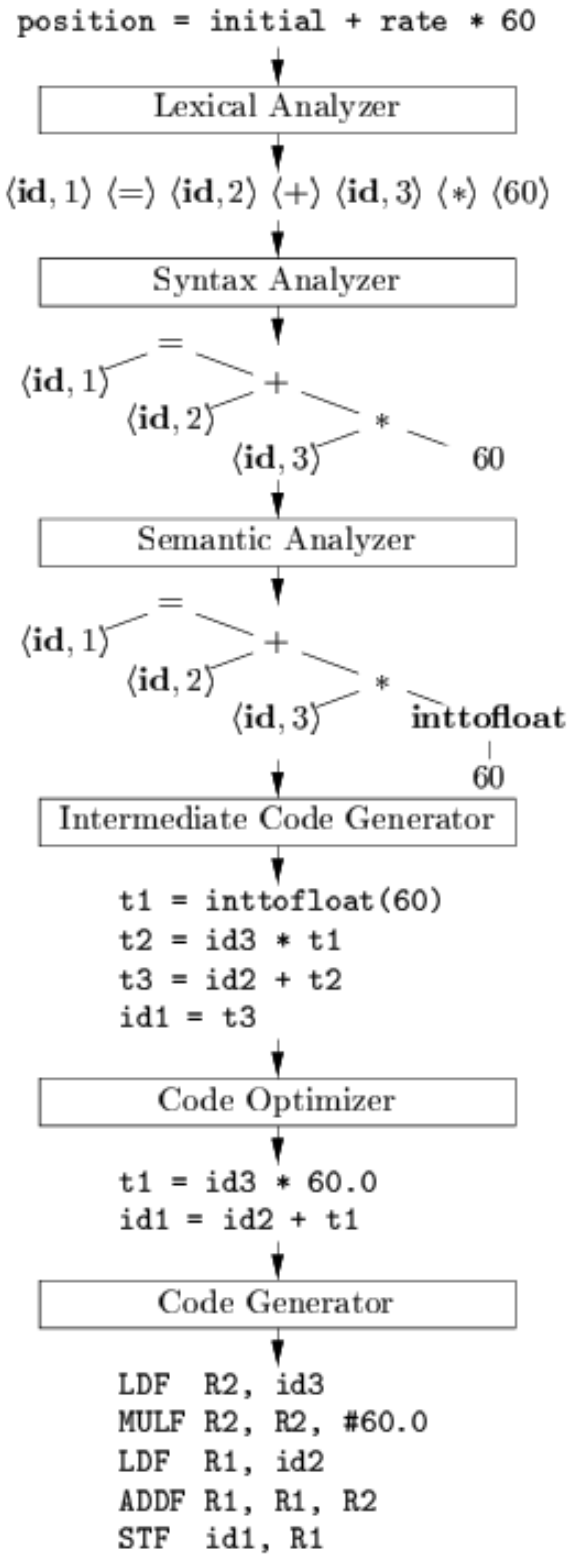


❑ Working of Compiler Phases with Example

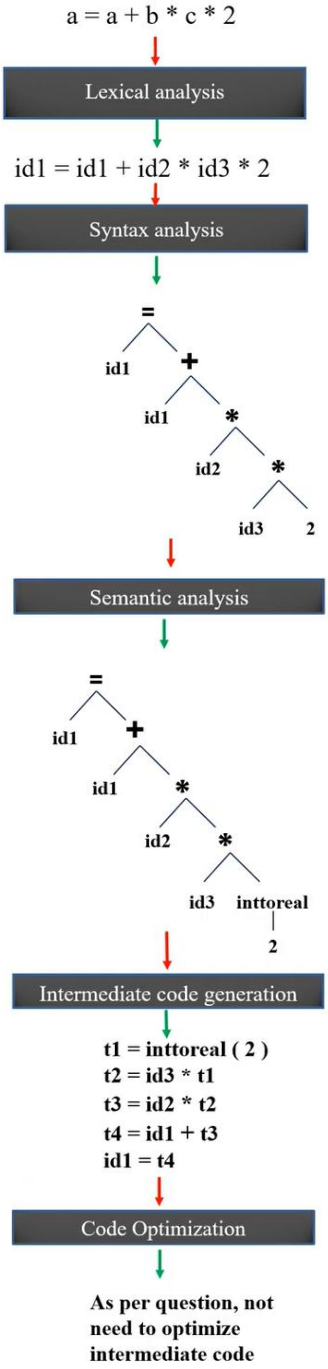
Example - 1:



Example - 2:



Example -3:



Code Generation

MOVF R1, #2
MOVF R2, id3
MULF R2, R1
MOVF R3, id2
MULF R3, R2
MOVF R4, id1
ADDF R4, R3
MOVF id1, R4

Example-4:

x = a+b*50

Solution: **Solve By yourself!**

Example – 5:

data = data + total/50

Solution: **Solve By yourself!**

Questions for Exam: -

1. what are the phases of language processor?
2. what are the phases of compiler?
3. write down the Working of Compiler Phases for this following example.