MCA 18 302
PRINCIPLES OF COMPILERS

MODULE 1

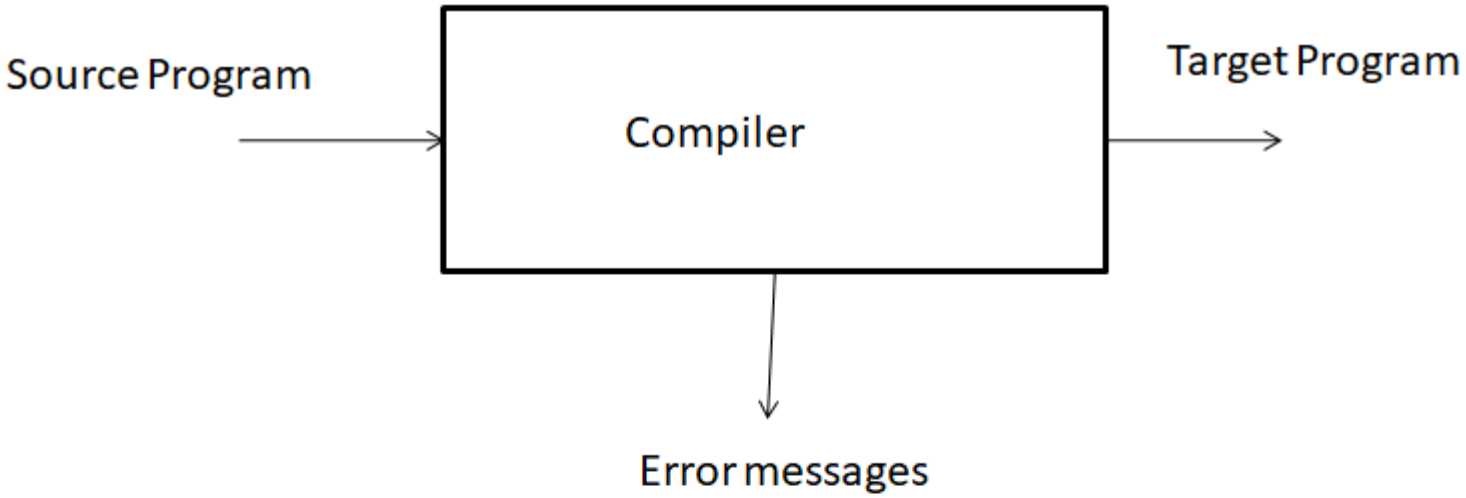
1. **Introduction to compiling**
    1. Definition of compiler, translator, interpreter
    2. Analysis of the source program
    3. The phases of a compiler
    4. Compiler construction tools
2. **Programming language basics**
3. **Lexical analysis**
    1. Role of lexical analyzer
    2. Input buffering
    3. Specification of tokens
    4. Recognition of tokens using finite automata
    5. Regular expressions and finite automata
    6. From NFA to DFA
    7. Regular expression to an NFA

Introduction to compiling

1. Definition Of Compiler, Translator, Interpreter:

a) Definition Of Compiler:

- Compiler is a software(translator/program) which converts a program written in high level language(source language) to low level language (object/target/machine language).
- It converts the whole program in one session and reports errors detected after the conversion.
- A compiler is processor dependent and platform dependent.
- Example: GCC(GNU compiler collection)


Source Program
Compiler
Target Program
Error messages

- **Features of Compilers:**

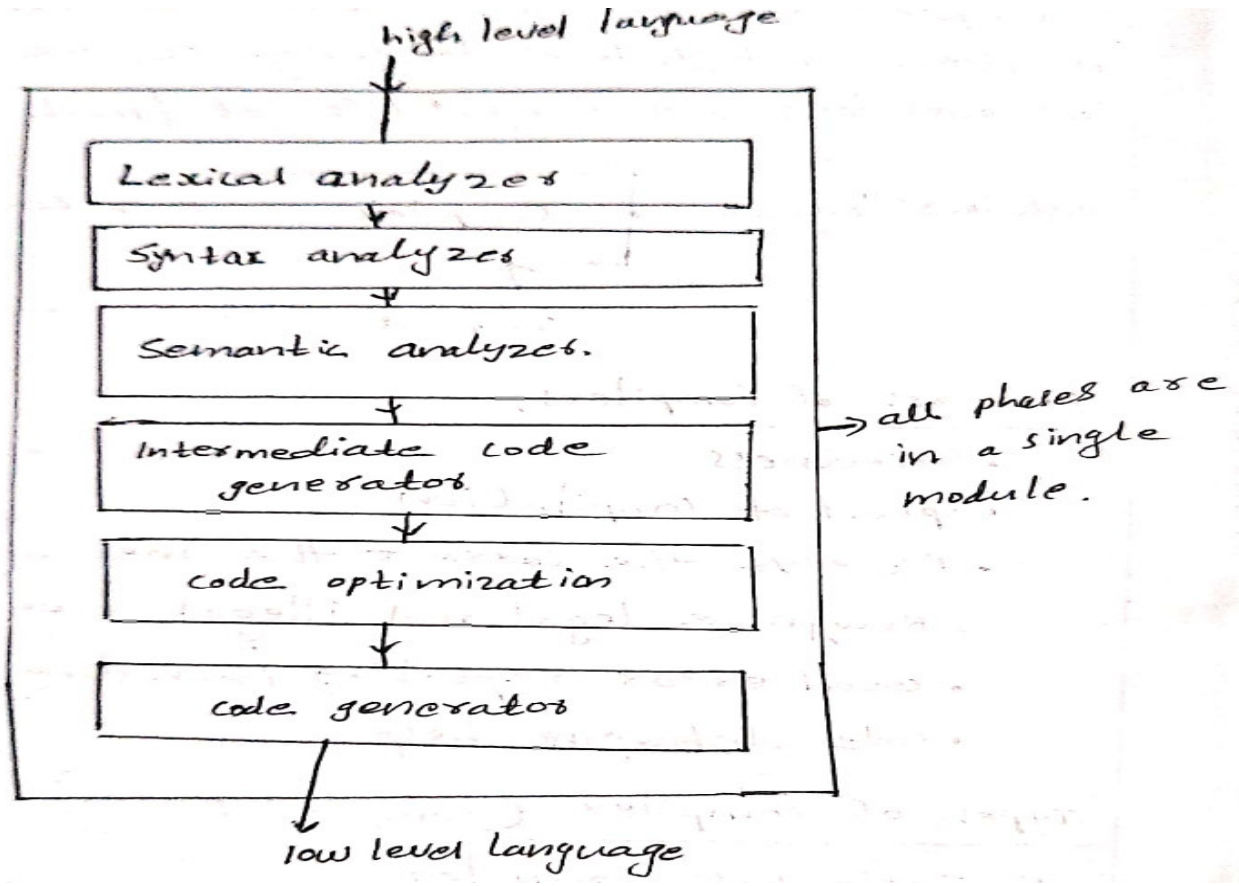
1. Correctness
2. Speed of compilation
3. Preserve the correct the meaning of the code.
4. Recognize legal and illegal program constructs
5. Good error reporting/handling
6. Code debugging help

➢ Types of compiler(Based on compiler passes):

- A compiler pass refers to the traversal of a compiler through the entire program .
- Compiler pass are two types:
    1. Single pass compiler
    2. Two pass compiler or Multi pass Compiler

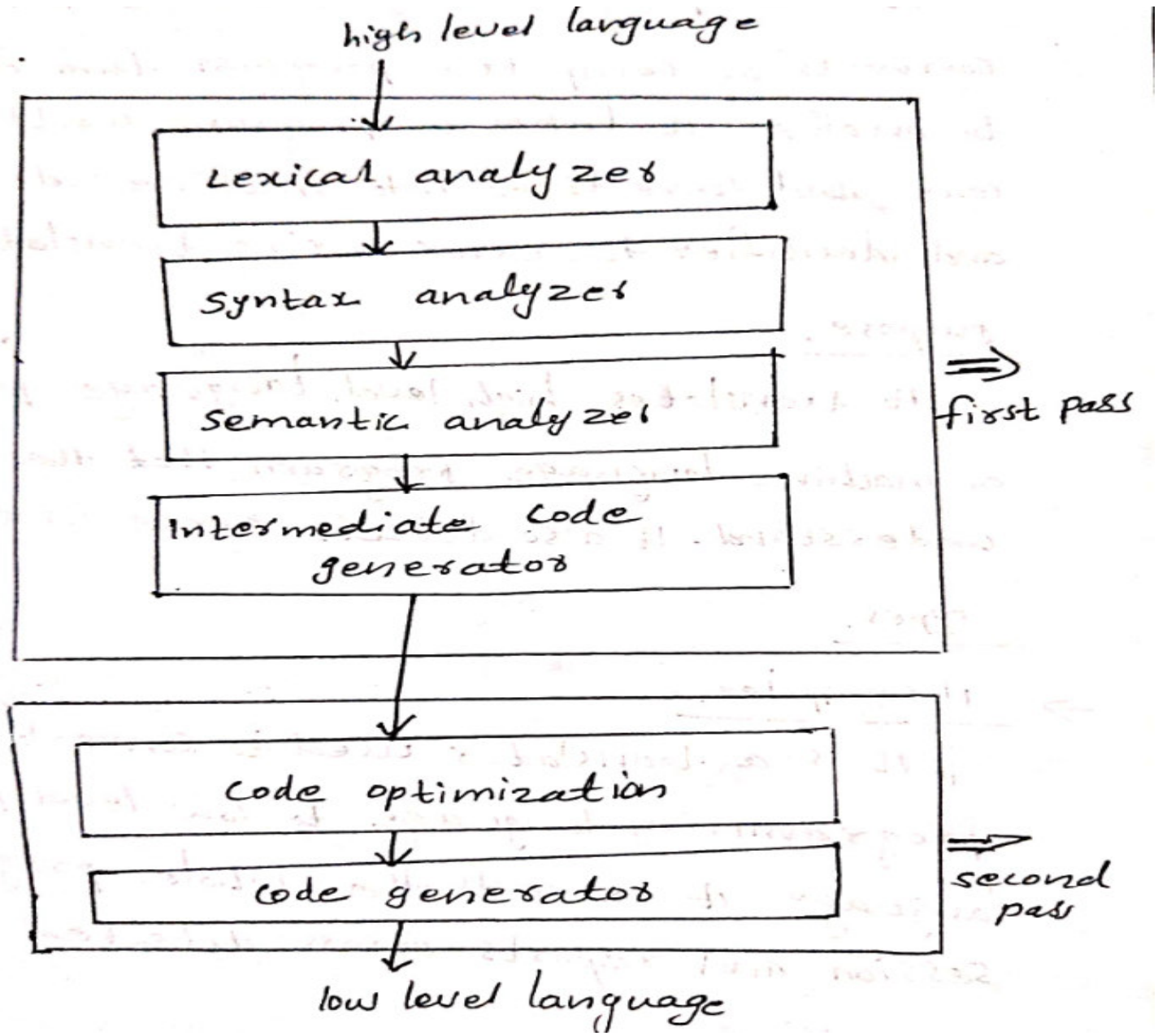
1. Single pass compiler:

- If we combine or group all the phases of compiler design in a single module known as single pass compiler.
- Example : Pascal ,C



2. Two pass compiler or multipass compiler:

- It is a type of compiler that processes the source code of a program multiple times.
- In multipass compiler we divide phases in two pass as:
  - First pass is refer as front end, analytic part and platform independent.
  - Second pass is refer as back end, synthesis part and platform dependent.
- Example Java ,C#


high level language
Lexical analyzer
Syntax analyzer
Semantic analyzer
Intermediate code generator
first pass
Code optimization
Code generator
second pass
low level language

- There are two parts to compilation:

1. **Analysis:**

   - The analysis part breaks up the source program into constituent pieces and creates an intermediate representation of the source program.

2. **Synthesis:**

   - The synthesis part constructs the desired target program from the intermediate representation.

- **Advantage:**

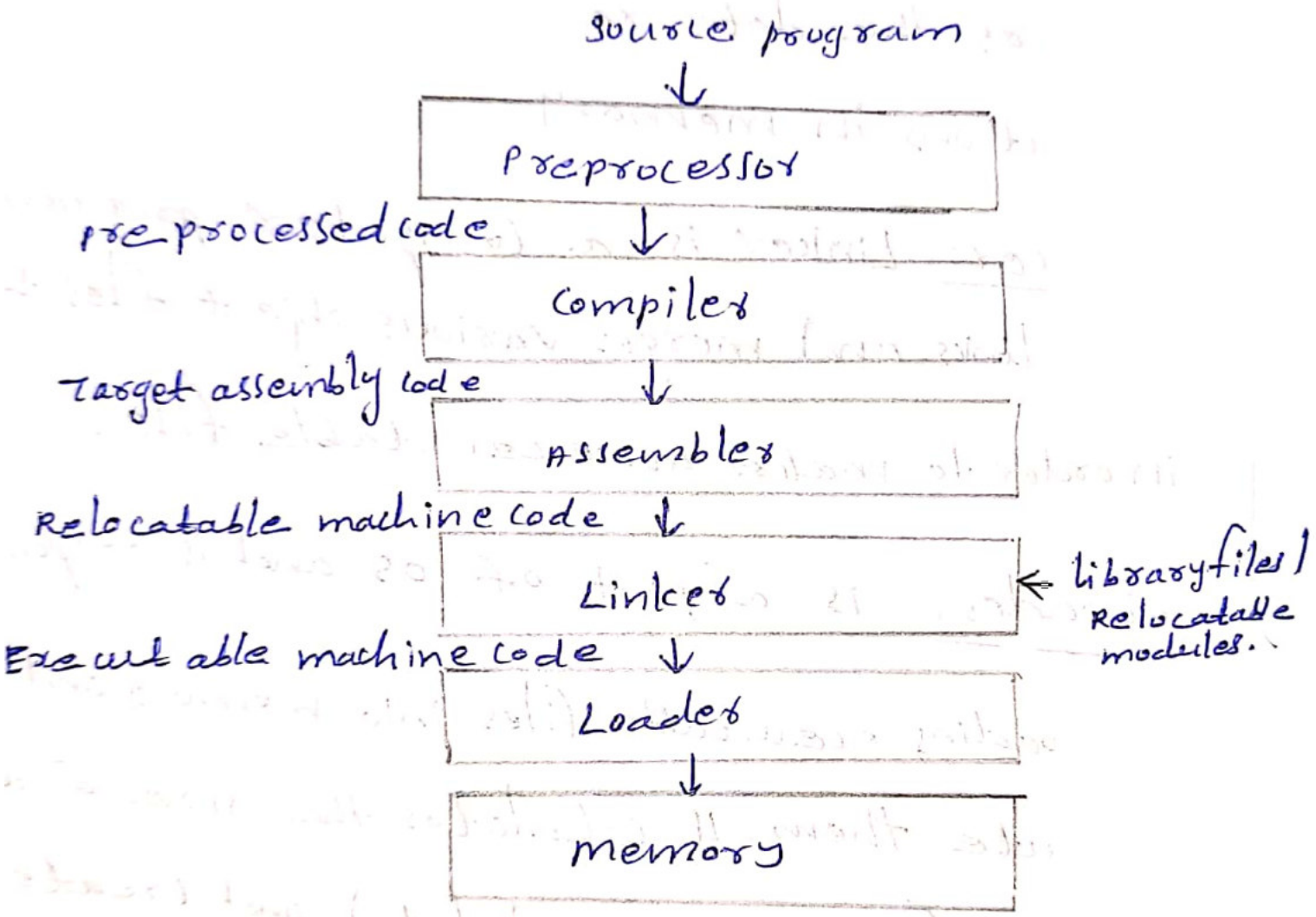
1. The whole program is validated so there are no system errors.
2. The executable file is enhanced by the compiler ,so it runs faster.

- **Disadvantage:**

1. It is slow to execute as you have to finish the whole program.
2. Hardware specific, it works on specific machine language and architecture.
3. It is not easy to debug as errors are shown at the end of the execution.

- **The context of a compiler: A language processing system:**
  - A language processor is a software program designed or used to perform tasks such as processing program code to machine code.
  - There are two main types of language processors: Interpreter and translator(Compiler and assembler).
  - The difference between an interpreter and a translator is that an interpreter is telling the computer what to do. A translator takes the program's code and converts it to machine code, allowing the computer to read it.

- We have learnt that any computer system is made of hardware and software. The hardware understands a language, which humans cannot understand. So we write programs in high level language, which is easier for us to understand and remember. These programs are then fed into a series of tools and OS component to get the desired code that can be used by the machine. This is known as **language processing system.**


Source program
Preprocessor
preprocessed code
Compiler
Target assembly code
Assembler
Relocatable machine code
Linker
library files / Relocatable modules.
Executable machine code
Loader
memory

- **Preprocessor:** A preprocessor, generally considered as a part of compiler, is a tool that produces input for compilers. It deals with macro processing,augmentation,file inclusion, language extension etc.
- **Compiler:** Compile the preprocessed code and translates it to assembly code.
- **Assembler:** An assembler translates assembly language into machine code. The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.

- **Linker:** Linker is a computer program that links and merges various object files together in order to make an executable file.
- **Loader:** It is a part of OS and is responsible for loading executable files into memory and execute them. It calculates the size of a program(instructions and data) and creates memory space for it. It initializes various register to initiate execution.

❖ Example : C compiler

- User writes a program in C language(high level language).
- The C compiler, compiles the program and translates it to assembly program(low level language).
- An assembler then translates the assembly program into machine code(object)
- A linker tool is used to link all the parts of the program together for execution(executable machine code).
- A loader loads all of them into memory and then the program is executed.

b) Translator:

- It is a programming language processor that converts a computer program from one language to another. It takes a program written in source code and converts it into machine code. It discovers and identifies the error during translation.
- **Purpose:**

  It translates high level language program into a machine language program that the CPU can understand. It also detects errors in the program.
- **Types**

  1. **Compiler**
  2. **Interpreter**
  3. **Assembler**

c) Interpreter:

- It is a translator used to convert high level programming language to low level programming language.
- It converts the program one at a time and reports errors detected at once, while doing the conversion.
- An interpreter is faster than a compiler as it immediately executes the code upon reading the code.
- It used as a debugging tool for software development as it can execute a single line of code at a time.
- An interpreter is also more portable than a compiler as it is not processor dependent.
- Example : Python

- **Advantage:**

1. You discover errors before you complete the program, so you learn from your mistakes.
2. Program can be run before it is completed so you get partial results immediately.
3. You can work on small parts of the program and link them later into a whole program.

- **Disadvantage:**

1. It may be slow because of the interpretation in every execution.
2. Program is not enhanced and may encounter data errors.
3. There's a possibility of syntax errors on unverified scripts.

d) Assembler:

- It is a translator used to translate assembly language to machine language.
- Example : Macro Assembly Program(MAP)