shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

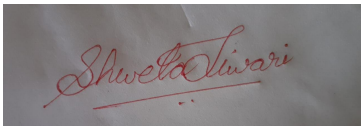
CD: UNIT-1 INTRODUCTION TO COMPILER DESIGN

FALL SEMESTER, YEAR (V/VI, 3rd)

Published: August, 2022

FALL SESSION: 2022-23

—



UNIT-1 CD
COMPILER DESIGN (KIT-052)

UNIT I – INTRODUCTION TO COMPILER DESIGN (PART-1)

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Guest Faculty

Rajkiya Engineering College | Ambedkar Nagar, UP, India

Faculty Name: Miss. Shweta Tiwari, Subject: CD-COMPILER DESIGN (KIT-052), Year- 3rd Year, Semester- 5th/6th Sem, Branch- CS and IT, Session- Odd/Even Semester (2022-23)

CD-COMPILER DESIGN (KIT-052)

This **(Question Bank Part-1 Easy to Advanced Level (Belong your Syllabus))** only attempts to discover some questions with proper explanation that can be generated in "CD-COMPILER DESIGN" with the answer to all these questions. There can be some errors to these answers. If you find any errors then please do write to us.

CD-COMPILER DESIGN (KIT-052)

UNIT I INTRODUCTION TO COMPILER (PART-1)

1. What is a Compiler?

A Compiler is a program that reads a program written in one language-the source language-and translates it into an equivalent program in another language-the target language . As an important part of this translation process, the compiler reports to its user the presence of errors in the source program

2.State some software tools that manipulate source programs?

I. Structure editors
II. Pretty printers
III. Static
IV. Checkers
V. Interpreters.

3.What are the cousins of the compiler? April/May 2004, April/May 2005

The following are the cousins of

I. Preprocessors
II. Assemblers
III. Loaders
IV. Link editors.

4.What are the main two parts of compilation? What are they performing?

The two main parts are

- **Analysis** part breaks up the source program into constituent pieces and creates

  an intermediate representation of the source program.
- **Synthesis** part constructs the desired target program from the intermediate representation

5.What is a Structure editor?

A structure editor takes as input a sequence of commands to build a source program .The structure editor not only performs the text creation and modification functions of an ordinary text editor but it also analyzes the program text, putting an appropriate hierarchical structure on the source program.

6.What are a Pretty Printer and Static Checker?

- A Pretty printer analyses a program and prints it in such a way that the structure of the program becomes clearly visible.

- A static checker reads a program, analyzes it and attempts to discover potential bugs without running the program.

7. How many phases does analysis consist of?

Analysis consists of three phases

I. Linear analysis
II. Hierarchical analysis
III. Semantic analysis

8. What happens in linear analysis?

This is the phase in which the stream of characters making up the source program is read from left to right and grouped into tokens that are sequences of characters having collective meaning.

9. What happens in Hierarchical analysis?

This is the phase in which characters or tokens are grouped hierarchically in to nested collections with collective meaning.

10. What happens in Semantic analysis?

This is the phase in which certain checks are performed to ensure that the components of a program fit together meaningfully.

11. State some compiler construction tools?April /May 2008

Parse generator
Scanner generators
Syntax-directed translation engines
Automatic code generator
Data flow engines.

12. What is a Loader? What does the loading process do?

A Loader is a program that performs the two functions

Loading
ii .Link editing

The process of loading consists of taking relocatable machine code, altering the relocatable address and placing the altered instructions and data in memory at the proper locations.

13. What does Link Editing do?

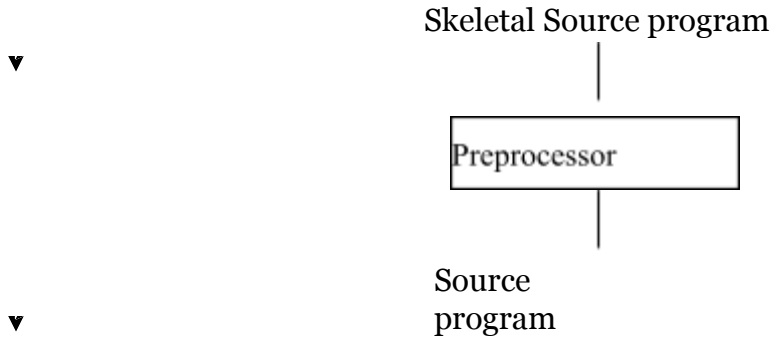
Link editing: This allows us to make a single program from several files of relocatable machine code. These files may have been the result of several compilations, and one or more may be library files of routines provided by the system

and available to any program that needs them.

14. What is a preprocessor?

A preprocessor is one, which provides input to compilers. A source program may be divided into modules stored in separate files. The task of collecting the source program is sometimes entrusted to a distinct program called a preprocessor.

The preprocessor may also expand macros into source language statements.



15. State some functions of Preprocessors

i) Macro processing
ii) File inclusion
iii) Relational Preprocessors
iv) Language extensions

16. What is a Symbol table?

A Symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier. The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly.

17. State the general phases of a compiler

Lexical analysis
Syntax analysis
Semantic analysis
Intermediate code generation
Code optimization
Code generation

18. What is an assembler?

Assembler is a program, which converts the source language into assembly language.

19. What is the need for separating the analysis phase into lexical analysis and parsing? (Or) What are the issues of lexical

analyzer?

- Simpler design is perhaps the most important consideration. The separation of lexical analysis from syntax analysis often allows us to simplify one or the other of these phases.
- Compiler efficiency is improved.
- Compiler portability is enhanced.

20. What is Lexical Analysis?

The first phase of the compiler is Lexical Analysis. This is also known as linear analysis in which the stream of characters making up the source program is read from left-to- right and grouped into tokens that are sequences of characters having a collective meaning.

21. What is a lexeme? Define a regular set. Nov/Dec 2006

- A Lexeme is a sequence of characters in the source program that is matched by the pattern for a token.
- A language denoted by a regular expression is said to be a regular set

22. What is a sentinel? What is its usage? April/May 2004

A Sentinel is a special character that cannot be part of the source program. Normally we use 'eof' as the sentinel. This is used for speeding-up the lexical analyzer.

23. What is a regular expression? State the rules, which define regular expression?

Regular expression is a method to describe regular language Rules:

1) ε-is a regular expression that denotes {ε} that is the set containing the empty string
2) If a is a symbol in ∑,then a is a regular expression that denotes {a}
3) Suppose r and s are regular expressions denoting the languages L(r) and L(s) Then,
    a) (r)/(s) is a regular expression denoting L(r) U L(s).
    b) (r)(s) is a regular expression denoting L(r)L(s)
    c) (r)* is a regular expression denoting L(r)*.
    d) (r) is a regular expression denoting L(r).

24. What are the Error-recovery actions in a lexical analyzer?

1. Deleting an extraneous character
2. Inserting a missing character
3. Replacing an incorrect character by a correct character
4. Transposing two adjacent characters

25. Construct Regular expression for the language

L= {w ε{a,b}/w ends in abb} Ans:
{a/b}*abb.

26. What is a recognizer?

Recognizers are machines. These are the machines which accept strings belonging to a certain language. If the valid strings of such language are accepted by the machine then it is said that the corresponding language is accepted by that machine, otherwise it is rejected.

16 MARKS

1. PHASES OF COMPILER

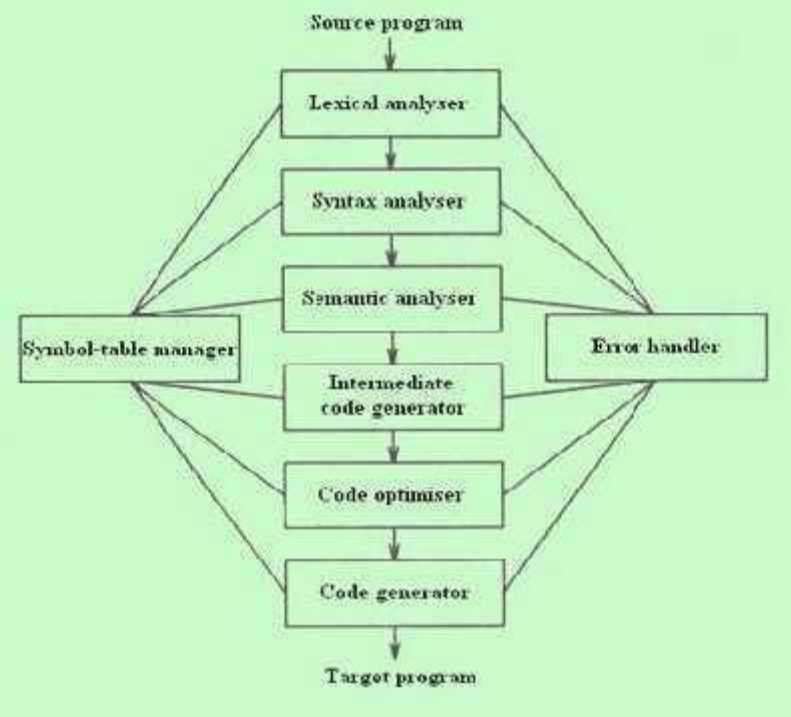
A Compiler operates in phases, each of which transforms the source program from one representation into another. The following are the phases of the compiler:

Main phases:

1) Lexical analysis
2) Syntax analysis
3) Semantic analysis
4) Intermediate code generation
5) Code optimization
6) Code generation

Sub-Phases:

1) Symbol table management
2) Error handling



LEXICAL ANALYSIS:

- It is the first phase of the compiler. It gets input from the source
- program and produces tokens as output. It reads the characters one by
- one, starting from left to right and forms the tokens.
  Token : It represents a logically cohesive sequence of characters such as keywords,
    - operators, identifiers, special symbols etc.
    - Example: a + b = 20
    - Here, a,b,+,=,20 are all separate tokens.
    - Group of characters forming a token is called the **Lexeme**.
- The lexical analyser not only generates a token but also enters the lexeme into the symbol
    - table if it is not already there.

SYNTAX ANALYSIS:

- It is the second phase of the compiler. It is also known as parser.
- It gets the token stream as input from the lexical analyser of the compiler and generates
    - syntax tree as

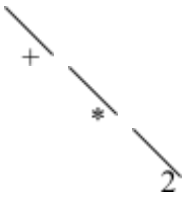
the output.
- Syntax tree:
    - It is a tree in which interior nodes are operators and exterior nodes are operands.
- Example: For a=b+c*2, syntax tree is

=

a

b

c



SEMANTIC ANALYSIS:

- It is the third phase of the compiler.
- It gets input from the syntax analysis as a parse tree and checks
- whether the given syntax is correct or not. It performs type conversion of all the data types into real data types.

INTERMEDIATE CODE GENERATION:

- It is the fourth phase of the compiler.
-

It gets input from the semantic analysis and converts the input into output as intermediate code such as three address code.

- The three-address code consists of a sequence of instructions, each of which has at most three operands.

Example: t1=t2+t3

CODE OPTIMIZATION:

- It is the fifth phase of the compiler.
- It gets the intermediate code as input and produces optimized intermediate code as output.

This phase reduces the redundant code and attempts to improve the intermediate code so that faster-running machine code will result.

- During the code optimization, the result of the program is not affected. To improve the code generation, the optimization involves

-deduction and removal of dead code (unreachable code).
-calculation of constants in expressions and terms.
-collapsing of repeated expressions into temporary strings.
-loop unrolling.
-moving code outside the loop.
-removal of unwanted temporary variables.

CODE GENERATION:

- It is the final phase of the compiler.
- It gets input from the code optimization phase and produces the target code or object code as result. Intermediate instructions are translated into a sequence of machine instructions that perform the same task. The code generation involves

-allocation of register and memory
-generation of correct references
-generation of correct data types
-generation of missing code

SYMBOL TABLE MANAGEMENT:

- Symbol table is used to store all the information about identifiers used in the program.
- It is a data structure containing a record for each identifier, with fields for the attributes of the identifier.

It allows us to find the record for each identifier quickly and to store or retrieve data from that record.

- Whenever an identifier is detected in any of the phases, it is stored in the symbol table.

ERROR HANDLING:

Each phase can encounter errors. After detecting an error, a phase must handle the error so that compilation can proceed.

- In lexical analysis, errors occur in separation of tokens.
- In syntax analysis, errors occur during construction of syntax trees.
- In semantic analysis, errors occur when the compiler detects constructs with right syntactic structure but no meaning and during type conversion.
- In code optimization, errors occur when the result is
- affected by the optimization. In code generation, it shows errors when code is missing etc.

2. COMPILER CONSTRUCTION TOOLS

These are specialized tools that have been developed for helping implement various phases of a compiler.

The following are the compiler construction tools:

1) Parser Generators:

-These produce syntax analyzers, normally from input that is based on a context-free grammar.

-It consumes a large fraction of the running time of a compiler.

-Example-YACC (Yet another Compiler-Compiler).

2) Scanner Generator:

-These generate lexical analyzers, normally from a specification based on regular expressions.

-The basic organization of lexical analyzers is based on finite automation.

3) Syntax-Directed Translation:

-These produce routines that walk the parse tree and as a result generate intermediate code.

-Each translation is defined in terms of translations at its neighbor nodes in the tree.

4) Automatic Code Generators:

-It takes a collection of rules to translate intermediate language into machine language. The rules must include sufficient details to handle different possible access methods for data.

5) Data-Flow Engines:

-It does code optimization using data-flow analysis, that is, the gathering of information about how values are transmitted from one part of a program to each other part.