Techno India NJR Institute of Technology

Academic Administration of Techno NJR Institute

Syllabus Deployment

Session 2019-20

Name of Faculty: Mr. Pankaj Chittora Subject Code: 5CS4-02
Subject Name: Compiler Design

Department: Department of Computer Science Engineering SEM: V

Total No. of Lectures Planned: 40

Textbook references:

TEXTBOOK:

1. Compilers, Principle, Techniques, and Tools. – Alfred.V Aho, Monica S.Lam, Ravi Sethi, Jeffrey
   D. Ullman.
2. Modern Compiler implementation in C , - Andrew N.Appel Cambridge University Press.

REFERENCE BOOKS:

1. lex & yacc , -John R Levine, Tony Mason, Doug Brown; O'reilly.
2. Compiler Construction,- LOUDEN, Thomson.
3. Engineering a compiler – Cooper & Linda, Elsevier
4. Modern Compiler Design – Dick Grune, Henry E.Bal, Cariel TH Jacobs, Wiley Dreatech

Additional Resources (NPTEL):

1. https://nptel.ac.in/courses/106/105/106105190/

Course Outcomes

CO35402.1	Students will be able to learn major concepts in areas of language translation and compiler design.
CO35402.2	Students will be able to ability to identify, formulate, and solve computer engineering problems with proper systematic & semantic approach.
CO35402.3	Students will be able to Develop possible program constructs for further code generation with Type checking.
CO35402.4	Students will be able to learn various concepts of symbol tables, Run time environments, memory management strategy.
CO35402.5	Students will get the concepts of Intermediate code generation, Code optimization and Code generations.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Lecture No.	Unit	Topic
1	1	Introduction of Compiler, Translator, Interpreter
2	1	Phase of compiler
3	1	Introduction to one pass & Multipass compilers, Bootstrapping
4	1	Review of Finite automata
5	1	Lexical analyzer, Input, buffering,
6	1	Recognition of tokens
7	1	A lexical analyzer generator, Error handling
8	2	Review of Context Free Grammar, Ambiguity of grammars
9	2	Introduction to parsing: Bottom up parsing
10	2	Top down parsing techniques
11	2	Shift reduce parsing, Operator precedence parsing
12	2	Recursive descent parsing predictive parsers
13	2	LL grammars & passers error handling of LL parser
14	2	LR parsers
15	2	Construction of SLR
16	2	Conical LR & LALR parsing tables
17	2	Parsing with ambiguous grammar
18	2	Introduction of automatic parser generator: YACC error handling in LR parsers
19	3	Construction of syntax trees
20	3	L-attributed definitions, Top down translation
21	3	Specification of a type checker
22	3	Intermediate code forms using postfix notation
23	3	Three address code, Representing TAC using triples and quadruples
24	3	Translation of assignment statement
25	3	Boolean expression and control structures
26	4	Storage organization, Storage allocation, Strategies
27	4	Activation records, Accessing local and non-local names in a block structured language -1
28	4	Activation records, Accessing local and non-local names in a block structured language -2
29	4	Parameters passing, Symbol table organization

30	4	Data structures used in symbol tables
31	5	Definition of basic block control flow graphs
32	5	DAG representation of basic block
33	5	Advantages of DAG, Sources of optimization, Loop optimization
34	5	Idea about global data flow analysis, Loop invariant computation
35	5	Peephole optimization
36	5	Issues in design of code generator
37	5	A simple code generator, Code generation from DAG
38		Revision of Important topics
39		Problem solving
40		Problem Solving

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-V Semester: B.Tech. Computer Science and Engineering

5CS4-02: Compiler Design

Credit: 3 **Max. Marks: 150(IA:30, ETE:120)**

3L+0T+0P **End Term Exam: 3 Hours**

SN	Contents	Hours
1	**Introduction:**Objective, scope and outcome of the course.	01
2	**Introduction:** Objective, scope and outcome of the course. Compiler, Translator, Interpreter definition, Phase of compiler, Bootstrapping, Review of Finite automata lexical analyzer, Input, Recognition of tokens, Idea about LEX: A lexical analyzer generator, Error handling.	06
3	**Review of CFG Ambiguity of grammars:** Introduction to parsing. Top down parsing, LL grammars & passers error handling of LL parser, Recursive descent parsing predictive parsers, Bottom up parsing, Shift reduce parsing, LR parsers, Construction of SLR, Conical LR & LALR parsing tables, parsing with ambiguous grammar. Operator precedence parsing, Introduction of automatic parser generator: YACC error handling in LR parsers.	10
4	**Syntax directed definitions;** Construction of syntax trees, S-Attributed Definition, L-attributed definitions, Top down translation. Intermediate code forms using postfix notation, DAG, Three address code, TAC for various control structures, Representing TAC using triples and quadruples, Boolean expression and control structures.	10
5	**Storage organization;** Storage allocation, Strategies, Activation records, Accessing local and non-local names in a block structured language, Parameters passing, Symbol table organization, Data structures used in symbol tables.	08
6	**Definition of basic block control flow graphs;** DAG representation of basic block, Advantages of DAG, Sources of optimization, Loop optimization, Idea about global data flow analysis, Loop invariant computation, Peephole optimization, Issues in design of code generator, A simple code generator, Code generation from DAG.	07
	Total	42

UNIT-I

INTRODUCTION TO LANGUAGE PROCESSING:

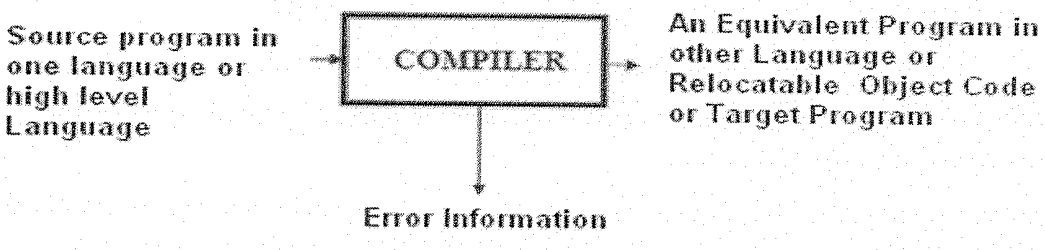
As Computers became inevitable and indigenous part of human life, and several languages with different and more advanced features are evolved into this stream to satisfy or comfort the user in communicating with the machine , the development of the translators or mediator Software's have become essential to fill the huge gap between the human and machine understanding. This process is called Language Processing to reflect the goal and intent of the process. On the way to this process to understand it in a better way, we have to be familiar with some key terms and concepts explained in following lines.

LANGUAGE TRANSLATORS :

Is a computer program which translates a program written in one (Source) language to its equivalent program in other [Target]language. The Source program is a high level language where as the Target language can be any thing from the machine language of a target machine (between Microprocessor to Supercomputer) to another high level language program.

Σ Two commonly Used Translators are Compiler and Interpreter

1. **Compiler :** Compiler is a program, reads program in one language called Source Language and translates in to its equivalent program in another Language called Target Language, in addition to this its presents the error information to the User.



Σ If the target program is an executable machine-language program, it can then be called by the users to process inputs and produce outputs.

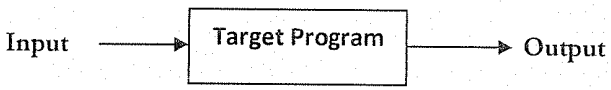


Figure1.1: Running the target Program

2. **Interpreter:** An interpreter is another commonly used language processor. Instead of producing a target program as a single translation unit, an interpreter appears to directly execute the operations specified in the source program on inputs supplied by the user.

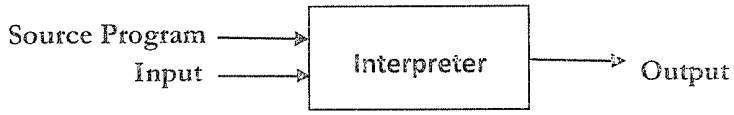


Figure 1.2: Running the target Program

LANGUAGE PROCESSING SYSTEM:

Based on the input the translator takes and the output it produces, a language translator can be called as any one of the following.

Preprocessor: A preprocessor takes the skeletal source program as input and produces an extended version of it, which is the resultant of expanding the Macros, manifest constants if any, and including header files etc in the source file. For example, the C preprocessor is a macro processor that is used automatically by the C compiler to transform our source before actual compilation. Over and above a preprocessor performs the following activities:

- ∑ Collects all the modules, files in case if the source program is divided into different modules stored at different files.
- ∑ Expands short hands / macros into source language statements.

Compiler: Is a translator that takes as input a source program written in high level language and converts it into its equivalent target program in machine language. In addition to above the compiler also

- ∑ Reports to its user the presence of errors in the source program.
- ∑ Facilitates the user in rectifying the errors, and execute the code.

Assembler: Is a program that takes as input an assembly language program and converts it into its equivalent machine language code.

Loader / Linker: This is a program that takes as input a relocatable code and collects the library functions, relocatable object files, and produces its equivalent absolute machine code.

Specifically,

- ∑ **Loading** consists of taking the relocatable machine code, altering the relocatable addresses, and placing the altered instructions and data in memory at the proper locations.
- ∑ **Linking** allows us to make a single program from several files of relocatable machine code. These files may have been result of several different compilations, one or more may be library routines provided by the system available to any program that needs them.

In addition to these translators, programs like interpreters, text formatters etc., may be used in language processing system. To translate a program in a high level language program to an executable one, the Compiler performs by default the compile and linking functions.

Normally the steps in a language processing system includes Preprocessing the skeletal Source program which produces an extended or expanded source program or a ready to compile unit of the source program, followed by compiling the resultant, then linking / loading , and finally its equivalent executable code is produced. As I said earlier not all these steps are mandatory. In some cases, the Compiler only performs this linking and loading functions implicitly.

The steps involved in a typical language processing system can be understood with following diagram.

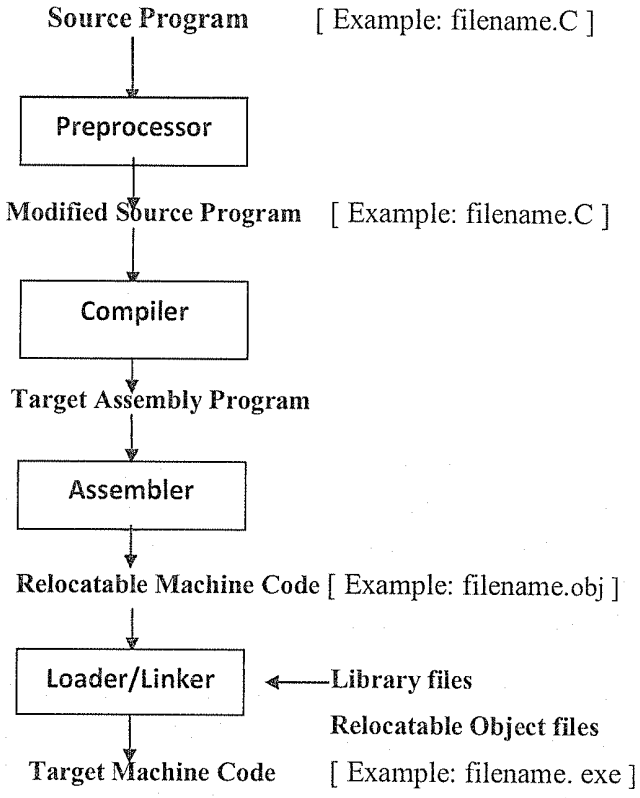


Figure1.3 : Context of a Compiler in Language Processing System

TYPES OF COMPILERS:

Based on the specific input it takes and the output it produces, the Compilers can be classified into the following types;

Traditional Compilers(C, C++, Pascal): These Compilers convert a source program in a HLL into its equivalent in native machine code or object code.

Interpreters(LISP, SNOBOL, Java1.0): These Compilers first convert Source code into intermediate code, and then interprets (emulates) it to its equivalent machine code.

Cross-Compilers: These are the compilers that run on one machine and produce code for another machine.

Incremental Compilers: These compilers separate the source into user defined–steps; Compiling/recompiling step- by- step; interpreting steps in a given order

Converters (e.g. COBOL to C++): These Programs will be compiling from one high level language to another.

Just-In-Time (JIT) Compilers (Java, Micosoft.NET): These are the runtime compilers from intermediate language (byte code, MSIL) to executable code or native machine code. These perform type –based verification which makes the executable code more trustworthy

Ahead-of-Time (AOT) Compilers (e.g., .NET ngen): These are the pre-compilers to the native code for Java and .NET

Binary Compilation: These compilers will be compiling object code of one platform into object code of another platform.

PHASES OF A COMPILER:

Due to the complexity of compilation task, a Compiler typically proceeds in a Sequence of compilation phases. The phases communicate with each other via clearly defined interfaces. Generally an interface contains a Data structure (e.g., tree), Set of exported functions. Each phase works on an abstract **intermediate representation** of the source program, not the source program text itself (except the first phase)

Compiler Phases are the individual modules which are chronologically executed to perform their respective Sub-activities, and finally integrate the solutions to give target code.

It is desirable to have relatively few phases, since it takes time to read and write immediate files. Following diagram (Figure1.4) depicts the phases of a compiler through which it goes during the compilation. There fore a typical Compiler is having the following Phases:

1. Lexical Analyzer (Scanner), 2. Syntax Analyzer (Parser), 3.Semantic Analyzer, 4.Intermediate Code Generator(ICG), 5.Code Optimizer(CO) , and 6.Code Generator(CG)

In addition to these, it also has **Symbol table management**, and **Error handler** phases. Not all the phases are mandatory in every Compiler. e.g, Code Optimizer phase is optional in some

cases. The description is given in next section.

The Phases of compiler divided in to two parts, first three phases we are called as Analysis part remaining three called as Synthesis part.

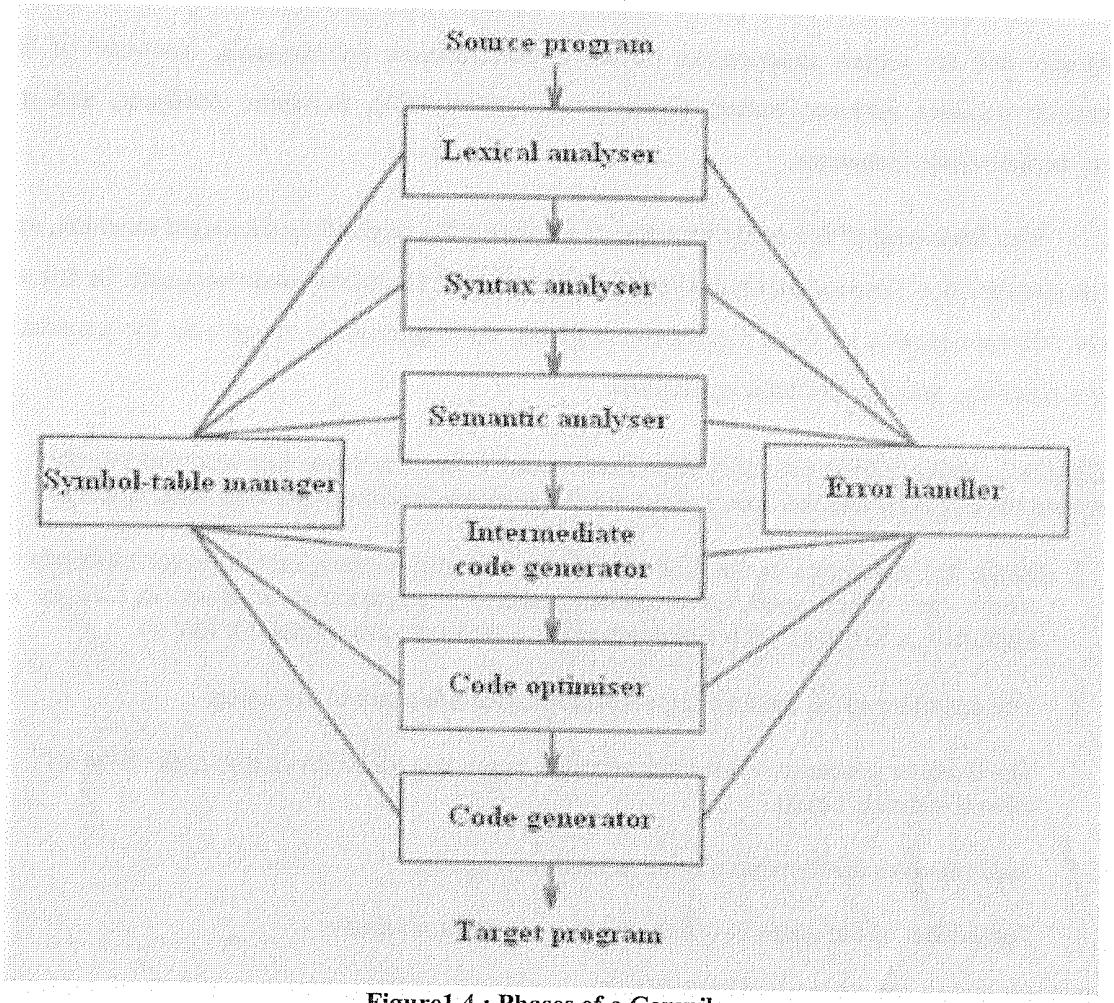


Figure1.4 : Phases of a Compiler

PHASE, PASSES OF A COMPILER:

In some application we can have a compiler that is organized into what is called passes. Where a pass is a collection of phases that convert the input from one representation to a completely deferent representation. Each pass makes a complete scan of the input and produces its output to be processed by the subsequent pass. For example a two pass Assembler.

THE FRONT-END & BACK-END OF A COMPILER

All of these phases of a general Compiler are conceptually divided into **The Front-end**, and **The Back-end**. This division is due to their dependence on either the Source Language or the Target machine. This model is called an Analysis & Synthesis model of a compiler.

The **Front-end** of the compiler consists of phases that depend primarily on the Source language and are largely independent on the target machine. For example, front-end of the compiler includes Scanner, Parser, Creation of Symbol table, Semantic Analyzer, and the Intermediate Code Generator.

The **Back-end** of the compiler consists of phases that depend on the target machine, and those portions don't dependent on the Source language, just the Intermediate language. In this we have different aspects of Code Optimization phase, code generation along with the necessary Error handling, and Symbol table operations.

LEXICAL ANALYZER (SCANNER): The Scanner is the first phase that works as interface between the compiler and the Source language program and performs the following functions:

- Reads the characters in the Source program and groups them into a stream of tokens in which each token specifies a logically cohesive sequence of characters, such as an identifier , a Keyword , a punctuation mark, a multi character operator like := .
- The character sequence forming a token is called a **lexeme** of the token.
- The Scanner generates a token-id, and also enters that identifiers name in the Symbol table if it doesn't exist.
- Also removes the Comments, and unnecessary spaces.

The format of the token is **< Token name, Attribute value>**

SYNTAX ANALYZER (PARSER): The Parser interacts with the Scanner, and its subsequent phase Semantic Analyzer and performs the following functions:

- Groups the above received, and recorded token stream into syntactic structures, usually into a structure called **Parse Tree** whose leaves are tokens.
- The interior node of this tree represents the stream of tokens that logically belongs together.
- It means it checks the syntax of program elements.

SEMANTIC ANALYZER: This phase receives the syntax tree as input, and checks the semantically correctness of the program. Though the tokens are valid and syntactically correct, it

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY UDAIPUR

Computer Science and Engineering

B. TECH III– YEAR (V Sem)

SUBJECT 5CS402

COMPILER DESIGN

ASSIGNMENT 1

Answer all questions. Each question carries 5 marks

1. Explain lexical analyzer generator and error handling. [CO-1]
2. What are the phases of a compiler? [CO-1]
3. Describe recursive descent parsing and predictive parsers. [CO-2]
4. What do you mean by ambiguous grammar ? [CO-2]
5. Construct any syntax tree. [CO-3]
6. Cite an example of top down translation. [CO-3]

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY UDAIPUR

Computer Science and Engineering

B. TECH III– YEAR (V Sem)

SUBJECT 5CS402

COMPILER DESIGN

ASSIGNMENT 2

Answer all questions. Each question carries 5 marks

1. Comment on Boolean expression and control structures. [CO-3]
2. Explain storage organization, allocation and strategies. [CO-4]
3. Explain parameters passing and symbol table organization. [CO-4]
4. How can you access local and non-local names in a block structured language-1 ? [CO-4]
5. Explain advantages of DAG. [CO-5]
6. Describe (i) loop variant computation (ii) peephole optimization [CO-5]

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY UDAIPUR

Computer Science and Engineering

B. TECH III– YEAR (V Sem)

SUBJECT 5CS402

COMPILER DESIGN

VIVA-VOCE SET OF QUESTIONS

1. Define regular expression. Give example. Write its applications.
2. What are the limitations of recursive descent parser?
3. Write the rules to compute operator precedence.
4. Why are quadruples preferred over triples in an optimizing compiler?
5. Write the usage of reference counting garbage collector
6. How redundant sub expression elimination can be done at global level in a given program?
7. Describe how various phases could be combined as a pass in a compiler?
8. Explain the transition diagram for recognition of tokens and reserved words.
9. Discuss briefly about the classification of parsing techniques.
10. Eliminate left recursion in the following grammar A → ABd | Aa | a B → Be | b
11. Show that Bottom up parsing is right most derivation in reverse order.
12. What are the common conflicts that can be encountered in shift reduce parsers? Explain.
13. Write the translation scheme to generate intermediate code for assignment statements with array references.
14. What is type system? Discuss static and dynamic checking of types.
15. What is an activation record? Explain how it is related with run time storage organization?
16. Explain various ways to access non local variables.
17. Discuss the role of semantic preserving transformations and dominators in code optimization. Justify the statement "Copy propagation Leads to Dead code"
18. Discuss in brief about left Recursion and Left Factoring with examples.
19. Define Regular Expression? Write about the identity rules for regular expressions.
20. Construct a Predictive parsing table for the Grammar E→E+T/T, T→T*F/F, F→(E)/id .
21. Define Ambiguous grammar? Explain it with an Example.
22. Construct CLR Parsing table for the grammar S→L=R/R, L→*R/id, R→L.
23. What is Dangling ELSE ambiguity? How to reduce it.
24. Translate the expression –(a+b)*(c+d)+(a+b+c) in to quadruple, triple and indirect triple.
25. Differentiate between Synthesized and Inherited attributes with suitable examples.
26. Define Symbol table? Explain about the data structures used for Symbol table.
27. Explain in brief about Stack Storage allocation strategy.
28. What are loop invariant Computations? Explain how they affect the efficiency of a program.
29. Explain in brief about different Principal sources of optimization techniques with suitable examples.
30. Define Boot strapping.
31. What are the draw backs of predictive parsing?

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY UDAIPUR

Computer Science and Engineering

B. TECH III– YEAR (V Sem)

SUBJECT 5CS402

COMPILER DESIGN

QUIZ

Attempt all questions. Each question carries 1 mark. No negative marking. *Time – 15 mins*

1. The action of parsing the source program into proper syntactic classes is called
(a) General syntax analysis
(b) Interpretation analysis
(c) Syntax analysis
(d) Lexical analysis

2. The bottom-up parsing method is also called
(a) Shift reduce parsing
(b) Predictive parsing
(c) Recursive descent parsing
(d) None of these

3. Synthesized attribute can be easily simulated by a
(a) LR grammar
(b) LL grammar
(c) Ambiguous grammar
(d) None of these

4. The most general phase structured grammar
(a) Context sensitive
(b) Context free
(c) Regular
(d) None of these

5. Which of the following is known as a compiler for a high-level language that runs on one machine and produces code for a different machine?
a) Cross compiler
b) Multipass compiler
c) Optimizing compiler
d) One pass compiler

6. Which of the following is correct regarding an optimizer Compiler?
a) Optimize the code
b) Is optimized to occupy less space
c) Both of the mentioned
d) None of the mentioned

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

7. Characters are grouped into tokens in which of the following phase of the compiler design?
a) Code generator
b) Lexical analyzer
c) Parser
d) Code optimization

8. Consider the following two sets of LR(1) items of an LR(1) grammar.

X -> c.X, c/d
X -> .cX, c/d
X -> .d, c/d
X -> c.X, $
X -> .cX, $
X -> .d, $

Which of the following statements related to merging of the two sets in the corresponding LALR parser is/are FALSE?

1. Cannot be merged since look aheads are different.
2. Can be merged but will result in S-R conflict.
3. Can be merged but will result in R-R conflict.
4. Cannot be merged since goto on c will lead to two different sets.

a.	1 only
b.	2 only
c.	1 and 4 only
d.	1, 2, 3, and 4

9.

The grammar S → aSa | bS | c is
a) LL(1) but not LR(1)
b) LR(1)but not LR(1)
c) Both LL(1)and LR(1)
d) Neither LL(1)nor LR(1)

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

10. What is the similarity between LR, LALR and SLR?
a) Use same algorithm, but different parsing table
b) Same parsing table, but different algorithm
c) Their Parsing tables and algorithm are similar but uses top down approach
d) Both Parsing tables and algorithm are different

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)