

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

QUESTION BANK



V SEMESTER

CS3562 COMPILER DESIGN

Regulation-2023

Academic Year 2026-27(Odd Sem)

Prepared by

Ms. N. Leo Bright Tennisson, Assistant Professor (Sr. G) /CSE

Ms.D.Ramya, Assistant Professor (O.G) /CSE

SRM VALLIAMMAI ENGINEERING COLLEGE
(An Autonomous Institution)
SRM Nagar, Kattankulathur – 603203.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
QUESTION BANK

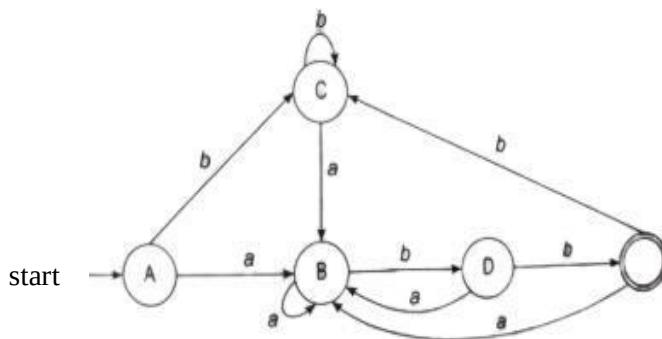
SUBJECT : CS3562 COMPILER DESIGN
SEM / YEAR : V / III

UNIT I -INTRODUCTION TO COMPILERS			
Phases of a compiler – Lexical Analysis – Role of Lexical Analyzer – Input Buffering – Specification of Tokens – Recognition of Tokens – Lex – Finite Automata – Regular Expressions to Automata – NFA to DFA- Minimizing DFA.			
PART-A (2 - MARKS)			
Q. No	QUESTIONS	Competence	BT Level
1.	Define tokens, patterns and lexemes.	Remember	BTL1
2.	Explain the approach would you use to recover the errors in lexical analysis phase?	Understand	BTL2
3.	Give the regular expression for identifier and white space.	Understand	BTL2
4.	Why is buffering used in lexical analysis?	Remember	BTL1
5.	Define transition diagram for an identifier.	Remember	BTL1
6.	Distinguish syntax tree and parse tree.	Understand	BTL2
7.	Write the issues in a lexical analyzer.	Remember	BTL1
8.	Define buffer pair.	Remember	BTL1
9.	Differentiate the features of DFA and NFA.	Understand	BTL2
10.	Identify the interactions between the lexical analyzer and the parser.	Remember	BTL1
11.	State parse tree and Draw a parse tree for $-(id + id)$	Remember	BTL1
12.	Name the operations on languages.	Remember	BTL1
13.	List the phases of a compiler.	Remember	BTL1
14.	Point out the advantage of having sentinels at the end of each buffer halves in buffer pairs.	Remember	BTL1
15.	Give the symbol table for the following statements. int a,b; float c; char z;	Understand	BTL2
16.	Discuss Regular expression and the Algebraic properties of Regular Expression.	Understand	BTL2
17.	Give the Structure of lex program.	Understand	BTL2
18.	Give a grammar for branching statements.	Understand	BTL2

19.	Express the main idea of NFA? And discuss with examples $(a/b)^*$	Understand	BTL2
20.	Define lex and give its execution steps.	Understand	BTL2
21	Differentiate interpreters and compilers	Understand	BTL2
22	Draw the parse tree for the statement $z := x + y * 130$.	Remember	BTL1
23	Outline the role of lexical analysis in compiler design.	Understand	BTL2
24	Describe the use of Input Buffering with simple examples.	Remember	BTL1

PART-B (13- MARKS)

1.	Explain the various phases of compiler with suitable example (16)	Evaluate	BTL5
2	(i) Give the structure of compiler. (6) (ii) Analyze structure of compiler with an assignment statement (10)	Analyze	BTL4
3.	(i) Express in detail about the role of Lexical analyzer with the possible error recovery schemes. (8) (ii) Examine in detail about issues in lexical analysis. (8)	Apply	BTL3
4	(i) Explain the Input buffering techniques in detail. (8) (ii) Explain about the recognition of tokens with example. (8)	Analyze	BTL4
5	Summarize in detail about how the tokens are specified by the compiler with suitable example. (16)	Evaluate	BTL5
6	Define Finite Automata. Differentiate Deterministic Finite Automata and Non-Deterministic Finite Automata with examples. (16)	Analyze	BTL4
7	Solve the given regular expression into NFA using Thompson construction i) $(a/b)^* abb (a/b)^*$. (8) ii) ab^*/ab (8)	Apply	BTL3
8	Create DFA for the following regular expression. $((\epsilon / a)b^*)^*$ (16)	Create	BTL6
9	(i) Illustrate the algorithm for minimizing the number of states of a DFA. (8) (ii) Minimize the following states of DFA. (8)	Apply	BTL3



10.	Explain in detail about the subset construction of DFA from NFA (16)	Create	BTL6
11	Define Lex and Lex specifications. Explain how lexical analyzer is constructed using lex? Give an example. (16)	Create	BTL6
12	(i) Explain the lex program for tokens. (8) (ii) Describe in detail the tool for generating lexical analyzer. (8)	Evaluate	BTL5
13	Find the NFA for the given regular expression and find the minimized DFA for the constructed NFA. $(a^* / b^*)^*$ (16)	Analyze	BTL4
14	Find the minimized DFA for the regular expression: $(0 + 1)^* (0 + 1) 1 0$. (16)	Analyze	BTL4
15	Explain in detail about the output of each phase of compiler for the expression $a:=b+c*50$. (16)	Analyze	BTL4
16	Demonstrate the role of lexical analyzer in detail with necessary diagrams (16)	Apply	BTL3
17	Determine the minimum -state DFA for the regular expression $(a / b)^* a (a/b)$ (16)	Evaluate	BTL5

UNIT II SYNTAX ANALYSIS

Role of Parser – Grammars – Error Handling – Context-free grammars – Writing a grammar – Top Down Parsing - General Strategies Recursive Descent Parser Predictive Parser-LL(1) Parser-Shift Reduce Parser-LR Parser-LR (0)Item Construction of SLR Parsing Table -Introduction to LALR Parser - Error Handling and Recovery in Syntax Analyzer-YACC.

PART-A (2 - MARKS)

1.	Eliminate the left recursion for the grammar. $S \rightarrow Aa \mid b$ $A \rightarrow Ac \mid Sd \mid \epsilon$	Understand	BTL2
2.	Define handle pruning.	Remember	BTL1
3.	Compute FIRST and FOLLOW for the following grammar $S \rightarrow AS$ $S \rightarrow b$ $A \rightarrow SA$ $A \rightarrow a$	Understand	BTL2
4.	State the concepts of Predictive parsing.	Remember	BTL1
5.	Differentiate Top Down parsing and Bottom Up parsing?	Understand	BTL2
6.	Define Recursive Descent Parsing.	Remember	BTL1
7.	State the different error recovery methods of predictive parsing.	Remember	BTL1
8.	Write an algorithm for finding FOLLOW.	Remember	BTL1
9.	What is the main idea of Left factoring? Give an example.	Understand	BTL2

10.	Define LL(1) Grammar.	Remember	BTL1
11.	Differentiate between ambiguous and unambiguous grammar.	Understand	BTL2
12.	Define parser. Explain the advantages and disadvantages of LR parsing?	Understand	BTL2
13.	Define Augmented Grammar with an example.	Remember	BTL1
14.	Explain the conflicts encountered while parsing.	Understand	BTL2
15.	Point out the categories of shift reduce parsing.	Understand	BTL1
16.	How to create an input and output translator with YACC.	Remember	BTL1
17.	Give the four possible actions of LR Parsing.	Understand	BTL2
18.	Prove that the following grammar is ambiguous: $S \rightarrow aSbS / bSaS / \epsilon$	Understand	BTL2
19.	Discuss when Dangling reference occur?	Understand	BTL2
20.	Illustrate the use of GOTO function.	Understand	BTL2
21.	Give the comparison between various LR parsers.	Understand	BTL2
22.	Write down the structure of YACC file.	Remember	BTL1
23.	Differentiate Lex and yacc.	Understand	BTL2
24.	Write about Closure Operation.	Remember	BTL1

PART-B (16-MARKS)				
1.	(i) Explain left recursion and Left Factoring. (ii) Eliminate left recursion and left factoring for the following grammar. $E \rightarrow E + T \mid E - T \mid T$ $T \rightarrow a \mid b \mid (E)$.	(8) (8)	Analyze	BTL4
2.	(i) Parse the input string 000111 for the grammar $S \rightarrow 0S1 \mid 01$ (ii) Construct a parse tree for the input string w-cad using top down parser. $S \rightarrow cAd$ $A \rightarrow ab \mid a$	(8) (8)	Create	BTL6
3.	(i) Analyze the give grammar to construct predictive parser $S \rightarrow +SS \mid *SS \mid a$ with the string “+*aaa.”	(16)	Analyze	BTL4
4.	(i) Evaluate predictive parsing table for the following grammar $E \rightarrow E + T \mid T$ $T \rightarrow T * F \mid F$ $F \rightarrow (E) \mid id$ (ii) Parse the string $id + id * id$	(10) (6)	Evaluate	BTL5
5.	Solve the following grammar for the predictive parser and parse the string 000111 $S > 0S$ 1 S- >01	(16)	Apply	BTL3
6.	(i). Explain in detail about the various types of parser (ii) Explain about the context-free grammar.	(8) (8)	Analyze	BTL4
7.	(i). Explain in detail about the role of parser. (ii). What are the Error recovery techniques used in Predictive parsing? Explain in detail.	(8) (8)	Evaluate	BTL5
8.	(i) Construct the predictive parser table for the following grammar. $S \rightarrow (L) \mid a$ $L \rightarrow L, S \mid S$ (ii) Parse the string (a, (a, a)).	(10) (6)	Create	BTL6
9.	(i) Analyze the following grammar is a LALR grammar. $S \rightarrow CC$ $C \rightarrow cC \mid d$ (ii) Parse the input string ba using the table generated.	(8) (8)	Analyze	BTL4
10.	(i) Define YACC parser generator. Explain the Error recovery actions in YACC. (ii) Define SLR (1) parser. Describe the Steps for the SLR parser.	(10) (6)	Evaluate	BTL5
11.	(i) Show SLR parsing table for the following grammar $A \rightarrow (A) \mid a$ (ii) Differentiate SLR and CLR	(10) (6)	Apply	BTL3

12.	Solve the following grammar to generate the SLR parsing table. $E \rightarrow E+T \mid T$ $T \rightarrow T * F \mid F$ $F \rightarrow F * \mid a \mid b$	(16)	Apply	BTL3
13.	(i) Consider the following grammar $S \rightarrow AS \mid b$ $A \rightarrow SA \mid a$. Construct the SLR parse table for the grammar. (ii) Show the actions of the parser for the input string “abab”. (6)	(10)	Apply	BTL3
14.	Formulate the LALR for the given grammar. $S \rightarrow AA$ $A \rightarrow Aa \mid b$	(16)	Create	BTL6
15.	Examine the following grammar using canonical parsing table. $S \rightarrow CC$ $C \rightarrow cC \mid d$	(16)	Apply	BTL3
16.	Explain SLR parser. Construct SLR parse for the given grammar. $S \rightarrow L=R$ $S \rightarrow R$ $L \rightarrow *R$ $L \rightarrow id$ $R \rightarrow L$	(16)	Evaluate	BTL5
17.	Show the bottom-up parser for the following The input $aaa*a++$ for the grammar $S \rightarrow SS+$ $S \rightarrow SS*$ $S \rightarrow a$	(16)	Apply	BTL3

UNIT-III INTERMEDIATE CODE GENERATION			
Syntax Directed Definitions, Evaluation Orders for Syntax Directed Definitions, Intermediate Languages: Syntax Tree, Three Address Code, Types and Declarations, Translation of Expressions, Type Checking.			
PART-A (2 - MARKS)			
1.	List out the two rules for type checking.	Remember	BTL1
2.	Compare synthesized attributes and inherited attributes.	Understand	BTL2
3.	What is Annotated parse tree?	Remember	BTL1
4.	Define Type checker.	Remember	BTL1
5.	What is a syntax tree? Draw the syntax tree for the assignment statement $a := b * -c + b * -c$	Remember	BTL1

6.	Define type systems.	Remember	BTL1
7.	Express the rule for checking the type of a function.	Understand	BTL2
8.	Define Syntax directed definition of a simple desk calculator.	Remember	BTL1
9.	Identify the different types of intermediate representation.	Remember	BTL1
10.	Give the difference between syntax-directed definitions and translation schemes.	Understand	BTL2
11.	State the type expressions.	Remember	BTL1
12.	Discuss the methods of implementing three-address statements.	Understand	BTL2
13.	Differentiate S-attribute and L-attribute definitions.	Understand	BTL2
14.	Give postfix notation for the given expression $a+b*c$.	Understand	BTL2
15.	Translate the conditional statement if $a < b$ then 1 else 0 into three address code.	Understand	BTL2
16.	Identify whether the following rules are L-attribute or not? Semantic rules. $A.s = B.b;$ $B.i = f(C.c, A.s)$	Remember	BTL1
17.	What are the methods of representing a syntax tree?	Remember	BTL1
18.	Give the syntax directed definition for if-else statement.	Understand	BTL2
19.	Explain the usage of syntax directed definition.	Understand	BTL2
20.	State the three-address code sequence for the assignment statement. $d = (a-b) + (a-c) + (a-c)$	Remember	BTL1
21.	Give the evaluation order of a SDD	Understand	BTL2
22.	What is translation scheme?	Remember	BTL1
23.	How will you evaluate semantic rules?	Remember	BTL1
24.	How to construct syntax tree for an expression?	Remember	BTL1
PART-B (13- MARKS)			
1.	Explain the following in detail about the Syntax Directed Definitions. (i) Inherited Attributes and Synthesized attributes. (8) (ii) Evaluate SDD of a parse tree. (8)	Analyze	BTL4

2.	Identify the annotated parse tree for the following expression (i)(3+4)*(5+6)n (ii)1*2*3*(4+5)n Using the given SDD Production
----	--

13.	(i) Specify the rules for type checking with example. (ii) Give an algorithm for type inference and polymorphic function.	(8) (8)	Create	BTL6
14.	Illustrate an algorithm for unification with its operation.	(16)	Apply	BTL3
15.	Formulate SDD for constructing syntax tree for the expression $a+b*5$	(16)	Create	BTL6
16.	Illustrate in detail about Bottom-up evaluation of S-attribute definitions	(16)	Apply	BTL3
17.	Explain the evaluation order for SDD	(16)	Evaluate	BTL5

UNIT IV- RUN-TIME ENVIRONMENT AND CODE GENERATION

Storage Organization, Stack Allocation Space, Access to Non-local Data on the Stack, Heap Management - Issues in Code Generation - Design of a simple Code Generator.

PART-A (2 -MARKS)

1.	List out limitations of the static memory allocation.	Remember	BTL1
2.	How the storage organization for the run-time memory is organized?	Remember	BTL1
3.	What is heap allocation?	Remember	BTL1
4.	How the activation record is pushed onto the stack.	Remember	BTL1
5.	Explain the storage allocation strategies.	Understand	BTL2
6.	State the principles for designing calling sequences.	Remember	BTL1
7.	List out the dynamic storage techniques.	Remember	BTL1
8.	Define the non-local data on stack.	Remember	BTL1
9.	Define variable data length on the stack.	Remember	BTL1
10.	Differentiate between stack and Heap allocation	Understand	BTL2
11.	Distinguish between static and dynamic storage allocation.	Understand	BTL2
12.	Discuss the main idea of Activation tree.	Understand	BTL2
13.	Give the fields in an Activation record.	Understand	BTL2
14.	Distinguish space efficiency and program efficiency.	Understand	BTL2
15.	Outline typical memory hierarchy configuration of a computer.	Remember	BTL1
16.	How would you solve the issues in the design of code generators?	Remember	BTL1
17.	Estimate Best-fit and Next-fit object placement.	Understand	BTL2
18.	Write optimal code sequence for the given sequence $t=a+b$ $t=t*c$ $t=t/d$	Remember	BTL1
19.	Describe the different forms of machine instructions.	Remember	BTL1
20.	Discuss the four principle uses of registers in code generation.	Understand	BTL2
21.	What is the input to code generator?	Remember	BTL1
22.	What are the advantages and disadvantages of register allocation and assignments?	Understand	BTL2
23.	How the use of registers is subdivided into 2 sub-problems?	Remember	BTL1
24.	State the contents of activation record.	Remember	BTL1

PART-B (13- MARKS)

1.	(i) Illustrate the storage organization memory in the perspective of compiler writer with neat diagram. (ii) Compare static versus dynamic memory allocation.	(10) (6)	Apply	BTL3
----	--	-------------	-------	------

2.	Explain in detail about the various issues in code generation with examples. (16)	Evaluate	BTL5
3.	(i)Develop a quicksort algorithm to reads nine integers into an array a and sorts them by using the concepts of activation tree. (10) (ii)Give the structure of the action record. (6)	Create	BTL6
4.	How to a design a call sequences and analyze the principles of activation records with an example. (16)	Analyze	BTL4
5.	Explain in detail about the activation tree and activation record with suitable example (16)	Analyze	BTL4
6.	(i) Analyze the data access without nested procedure and the issues with nested procedure. (8) (ii)Give the version of quicksort in ML style using nested procedure. (8)	Analyze	BTL4
7.	(i)Discuss in detail about heap manager. (8) (ii)Explain in detail about the memory hierarchy of a computer (8)	Analyze	BTL4
8.	Define fragmentation? Determine how to reduce the fragment. (16)	Create	BTL6
9.	Explain the following i. Best fit and next object placement. (8) ii. Managing and coalescing free space (8)	Create	BTL6
10.	Examine the problems with manual deallocation of memory and explain how the conventional tools are used to cope with the complexity in managing memory. (16)	Apply	BTL3
11.	Explain in detail about instruction selection and register allocation of code generation. (16)	Analyze	BTL4
12.	Illustrate in detail about the code generation algorithm with an example. (16)	Apply	BTL3
13.	Explain the usage of stack in the memory allocation and discuss in detail about stack allocation space of memory. (16)	Evaluate	BTL5
14.	Explain the heap management of memory manager and locality of programs in detail. (16)	Evaluate	BTL5
15.	Explain the problem that occurs in code generation with example (16)	Evaluate	BTL5
16.	Illustrate the function of code generation algorithm in detail (16)	Analyze	BTL3
17.	Summarize about access links, manipulation of access links and access links for procedure (16)	Evaluate	BTL5

UNIT V- CODE OPTIMIZATION

Principal Sources of Optimization – Peep-hole optimization - DAG- Optimization of Basic Blocks
Global Data Flow Analysis - Efficient Data Flow Algorithm.

PART-A (2 -MARKS)

1.	List out the examples of function preserving transformations.	Remember	BTL1
----	---	----------	------

2.	Explain the concepts of copy propagation.	Understand	BTL2
3.	State the use of machine Idioms.	Remember	BTL1
4.	Draw the flow graph for the quicksort algorithm	Remember	BTL1
5.	Explain depth first ordering.	Understand	BTL2
6.	Identify the constructs for optimization in basic block.	Remember	BTL1
7.	List out the properties of optimizing compilers.	Remember	BTL1
8.	Define the term data flow analysis.	Remember	BTL1
9.	How is liveness of a variable calculated? Identify it.	Remember	BTL1
10.	What is DAG? Point out advantages of DAG.	Remember	BTL1
11.	Give the uses of gen and Kill functions	Understand	BTL2
12.	Discuss the concepts of basic blocks and flow graphs.	Understand	BTL2
13.	Give the main idea of constant folding.	Understand	BTL2
14.	Give the three-address code sequence for the assignment statement. $d := (a - b) + (a - c) + (a - c)$.	Understand	BTL2
15.	Draw and explain the DAG for the follow basic block. $d := b * c$ $e := a + b$ $b := b * c$ $a := e - d$.	Remember	BTL1
16.	What role does the target machine play on the code generation phase of the compiler? Analyze it.	Remember	BTL1
17.	Draw the DAG for the statement $a = (a * b + c) - (a * b + c)$ and evaluate it.	Remember	BTL1
18.	Write the code for the follow C statement assuming three registers are available. $x = a / (b + c) - d * (e + f)$	Remember	BTL1
19.	Point out the characteristics of peephole optimization.	Remember	BTL1
20.	Define algebraic transformations. Give an example	Remember	BTL1
21.	What is a flow graph?	Remember	BTL1
22.	What is dead code elimination? Give example.	Understand	BTL2
23.	Give an example for code motion.	Understand	BTL2
24.	How the strength reduction is applied in code optimization?	Remember	BTL1
PART-B (13 MARKS)			
1.	Explain briefly about the principal sources of optimization. (16)	Evaluate	BTL5
2.	(i) Explain in detail about optimization of basic blocks. (6) (ii) Construct the DAG for the following Basic block & explain it. (10)	Analyze	BTL4

	<pre> t1:= 4 * i t2:= a [t1] t3:= 4 * i t4:= b [t3] t5:=t2*t4 t6:=Prod+t5 Prod:=t6 t7:=i+1 i:= t7 if i<= 20 goto (1).</pre>		
3.	Explain the following in detail (i)Semantic preserving transformation (8) (ii)Global Common subexpression (8)	Evaluate	BTL5
4.	Explain the following in detail (i)copy propagation (6) (ii)Dead code Elimination (6) (iii)code motion (4)	Evaluate	BTL5
5.	Explain in detail about the data-flow schemas on basic block and the transfer equations for reaching definitions with example (16)	Analyze	BTL4
6.	(i) Illustrate the Iterative algorithm for reaching definitions (8) (ii)Discuss the live variable analysis (8)	Apply	BTL3
7.	Analyze Peephole optimization with suitable examples. (16)	Analyze	BTL4
8.	Demonstrate optimization of Basic Blocks with an example. (16)	Apply	BTL3
9.	(i)Explain in detail about how to find Local Common Sub expressions. (10) (ii)Explain in detail about the Use of Algebraic Identities. (6)	Evaluate	BTL5
10.	(i)Describe in detail about the flow of control optimization. (8) (ii)Identify the methods to eliminate the unreachable code, load and store data. (8)	Analyze	BTL4
11.	(i)Give an example to identify the dead code in the DAG. (6) (ii)Demonstrate the representation of array using DAG with example. (10)	Apply	BTL3
12.	Summarize in detail about the dataflow analysis of available expression with suitable example. (16)	Evaluate	BTL5
13.	(i)Formulate steps to identify the loops in the basic block. (8) (ii) Describe about induction variable and end reduction in strength (8)	Create	BTL6
14.	Explain the efficient data flow algorithms in detail. (16)	Analyze	BTL4
15.	Explain in detail about optimization method performed on a small set of compiler generated instructions (16)	Evaluate	BTL5
16.	Summarize in detail about structure preserving transformation in detail (16)	Create	BTL6
17.	Illustrate in detail about DAG Representation of basic block and Write algorithm for DAG Construction. (16)	Apply	BTL3