SRM VALLIAMMAI ENGINEERING COLLEGE
(An Autonomous Institution)
SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

QUESTION BANK

V SEMESTER

CS3562– COMPILER DESIGN

2023 Regulation

Academic Year 2025 – 2026 (ODD)

Prepared by

Ms. Shanthi S, Assistant Professor (Sel.Gr)

Ms. V. Prema Assistant Professor (Sr.G)

Ms. B. Christina Sweetline, Assistant Professor (O.G)

SRM VALLIAMMAI ENGNIEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603203.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

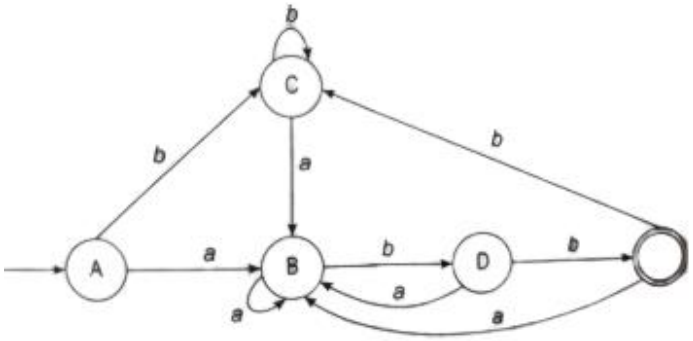
QUESTION BANK

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.	What are the approaches 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.	Differentiate syntax tree and parse tree.	Understand	BTL2
7.	What are 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	Define parse tree and give the parse tree for –(id + id)	Understand	BTL2
12.	Name the operations on languages.	Remember	BTL1
13.	List out the phases of a compiler.	Remember	BTL1
14.	Give the advantage of having sentinels at the end of each buffer halves in buffer pairs.	Understand	BTL2
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.	What is the Structure of lex program.	Remember	BTL1
18.	Give a grammar for branching statements.	Remember	BTL1
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.	Understand	BTL2
23	Outline the role of lexical analysis in compiler design.	Understand	BTL2
24	What are the use of Input Buffering with simple examples.	Remember	BTL1
	PART-B (13- MARKS)		
1.	Analyze the various phases of compiler with suitable example (16)	Analyze	BTL4
2	(i)Give the structure of compiler. (6) (ii)Analyze structure of compiler with an assignment statement (10)	Analyze	BTL4
3.	Discuss in detail about the role of Lexical analyzer with the possible error recovery schemes and describe in detail about issues in lexical analysis. (16)	Apply	BTL3
4	(i)Describe the Input buffering techniques in detail. (8) (ii)Discuss about the recognition of tokens with example (8)	Apply	BTL3
5	(i)Discuss in detail about how the tokens are specified by the compiler with suitable example. (5) (ii) Evaluate that the following two regular expressions are equivalent by showing that the minimum state DFA's are same. (11) (a / b) * (a * / b *) *	Create	BTL6
6	(i)Define Finite Automata. Differentiate Deterministic Finite Automata and Non-Deterministic Finite Automata with examples. (8) (ii)) Create languages denoted by the following regular expressions (8) a) (a|b)*a(a|b)(a|b) b) a*ba*ba*ba*	Create	BTL6
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 the following regular expression.((ε /a)b*)* (16)	Create	BTL6
9	(i)Illustrate the algorithm for minimizing the number of states of a DFA (8)	Apply	BTL3

	(ii)Minimize the following states of DFA (8) 		
10.	Describe in detail about the subset construction of DFA from NFA (16)	Evaluate	BTL5
11	Define Lex and Lex specifications. How lexical analyzer is constructed using lex? Give an example. (16)	Apply	BTL3
12	Find transition diagrams for the following regular expression and regular definition. (16) a(a|b)*a ((ε|a)b*)* All strings of digits with at most one repeated digit. All strings of a's and b's that do not contain the substring abb. All strings of a's and b's that do not contain the subsequence abb.	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: (16) (0 + 1) * (0 + 1) 1 0.	Analyze	BTL4
15	Evaluate that the following two regular expressions are equivalent by showing that the minimum state DFA's are sam (16) (a / b) * (a * / b *) *	Evaluate	BTL5
16	Demonstrate the role of lexical analyzer in detail with necessary diagrams and explain in detail the tool for generating Lexical-Analyzer with an example program. (16)	Apply	BTL3
17	Determine the minimum -state DFA for the regular expression (16) (a / b)* a (a/b)	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 → Aa | b A →Ac | Sd |ε	Understand	BTL2
2.	Define handle pruning.	Remember	BTL1
3.	Compute FIRST and FOLLOW for the following grammar S → AS S →b A →SA A → 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.	Understand	BTL2
9.	What is the main idea of Left factoring? Give an example.	Understand	BTL2
10.	Define LL(1) Grammar.	Remember	BTL1
11.	Difference between ambiguous and unambiguous grammar.	Understand	BTL2
12.	Define parser. Explain the advantages and disadvantages of LR parsing?	Remember	BTL1
13.	Define Augmented Grammar with an example.	Remember	BTL1
14.	What are the conflicts encountered while parsing?	Remember	BTL1
15.	Point out the categories of shift reduce parsing.	Understand	BTL2
16.	How to create an input and output translator with YACC.	Understand	BTL2
17.	Give the four possible actions of LR Parsing.	Understand	BTL2
18.	Solve the following grammar is ambiguous: S→aSbS / bSaS / €	Understand	BTL2
19.	Discuss when Dangling reference occur?	Understand	BTL2
20.	Illustrate the use of GOTO function.	Remember	BTL1
21.	Give the comparison between various LR parsers	Remember	BTL1
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 (13- MARKS)			
1.	Explain left recursion and Left Factoring and eliminate left recursion and left factoring for the following grammar. (16) E → E + T | E - T | T T → a | b | (E).	Analyze	BTL4

2.	Parse the input string 000111 for the grammar S-> 0S1| 01 And construct a parse tree for the input string w->cad using top down parser . S->cAd A->ab|a	(16)	Create	BTL6
3.	Analyze the give grammar to construct predictive parser S→ +SS | *SS | a with the string "+*aaa.	(16)	Analyze	BTL4
4.	(i)Evaluate predictive parsing table for the following grammar E→E+T | T T→T*F | F F→(E) | id (ii) Parse the string id+id*id	(11) (5)	Evaluate	BTL5
5.	Solve the following grammar for the predictive parser and parse the string 000111 S>0S1 S->01	(16)	Analyze	BTL4
6.	(i).Describe on detail about the various types of parser (ii)Discuss about the context-free grammar.	(11) (5)	Analyze	BTL4
7.	Discuss in detail about the role of parser and What are the Error recovery techniques used in Predictive parsing?	(16)	Analyze	BTL4
8.	(i)Give the predictive parser table for the following grammar. S→ (L) | a L→ L, S | S (ii)Parse the string (a, (a, a)) .	(11) (5)	Create	BTL6
9.	(i_Analyze the following grammar is a LALR grammar. S->CC C->cC|d Parse the input string ba using the table generated.	(16)	Analyze	BTL4
10.	(i)Define YACC parser generator. Generator with an example program (ii) What is Leftmost derivation and Rightmost derivation. Draw leftmost derivation and Rightmost derivation for the following. E->E+E|E*E| id	(11) (5)	Create	BTL6
11	(i)Show SLR parsing table for the following grammar A->(A)|a ii)Differentiate SLR and CLR	(11) (5)	Apply	BTL3
12.	Solve the following grammar to generate the SLR parsing table. E→E+T | T T→T*F | F F→F*| a| b	(16)	Create	BTL6
13.	(i)Consider the following grammar	(11)	Apply	BTL3

	S →AS|b A→SA|a. Construct the SLR parse table for the grammar. (ii)Show the actions of the parser for the input string "abab". (5)		
14.	Give the LALR for the given grammar. S->AA A->Aa|b (16)	Evaluate	BTL5
15.	Examine the following grammar using canonical parsing table. S->CC C->cC|d (16)	Evaluate	BTL5
16.	Explain SLR parser. Construct SLR parse for the given grammar. S->L=R S->R L->*R L->id R->L (16)	Evaluate	BTL5
17.	Show the bottom up parser for the following The input aaa*a++ for the grammar S->SS+ S->SS* S->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.	Differentiate the 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.	Give 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.	Give the methods of implementing three-address statements.	Understand	BTL2
13.	Differentiate S-attribute and L-attribute definitions.	Understand	BTL2
14.	Give the 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.	Discuss whether the following rules are L-attribute or not? Semantic rules A.s = B.b; B.i = f(C.c,A.s)	Understand	BTL2
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.	What is the usage of syntax directed definition	Remember	BTL1
20.	Give the three address code sequence for the assignment statement. d=(a-b)+(a-c)+(a-c)	Understand	BTL2
21.	Give the evaluation order of a SDD	Understand	BTL2
22	What is translation scheme?	Understand	BTL2
23.	How to evaluate semantic rules?	Understand	BTL2
24.	Evaluate syntax tree for an expression	Understand	BTL2
PART-B (13- MARKS)			
1.	Demonstrate the Syntax Directed Definitions of Inherited Attributes and Synthesized attributes. and evaluate SDD of a parse tree. (16)	Apply	BTL3
2.	Identify the annotated parse tree for the following expression (i)(3+4)*(5+6)n (8) (ii)1*2*3*(4+5)n (8) Using the given SDD Production — Semantic Rules D —>TL — L.inh = T.type T —> int — T.type =integer T —> float — T.type = float L —> L1, id — L1.inh = L.inh; addType (id.entry, Linh)	Evaluate	BTL5
3.	Suppose that we have a production A→BCD. Each of the four non terminal A, B, C and D have two attributes: S is a synthesized attribute and i is an inherited attribute. Analyze (16) For each of the sets of rules below tell whether (i)the rules are consistent with an S-attributed definition(ii) the rules are	Analyze	BTL4

	consistent with an L-attributed definition and(iii) whether the rules are consistent with any evaluation order at all? A.s = B.i + C.s A.s = B.i + C.s and D.i = A.i + B.s.			
4.	Generate an intermediate code for the following code segment with the required syntax-directed translation scheme. (i)if (a > b) x = a + b else x = a – b (ii) p>q AND r<s OR u>r	(16)	Apply	BTL3
5.	Explain in detail about (i)Dependency graph (ii)Ordering Evaluation of Attributes.	(11) (5)	Evaluate	BTL5
6.	(i)Create variants of Syntax tree and explain in detail about it with suitable examples. (ii)What is Type conversion? What are the two types of type conversion? Formulate the rules for the type conversion (8)	(8)	Create	BTL6
7.	(i). Analyze the common three address instruction forms. (ii). Explain the two ways of assigning labels to the following three address statements Do i=i+1; While (a[i]<v);	(8) (8)	Analyze	BTL4
8.	Analyze intermediate code for the following code segment with the required syntax-directed translation scheme. (i)if (a > b) x = a + b else x = a – b (ii) p>q AND r<s OR u>r	(8) (8)	Analyze	BTL4
9.	(i) Explain in detail about addressing array Elements. (ii) Explain in detail about Translation of array reference.	(8) (8)	Evaluate	BTL5
10.	Explain in detail about types and declaration with suitable examples.	(16)	Evaluate	BTL5
11.	Compare three address code for expression with the Incremental translation.	(16)	Analyze	BTL4
12.	Analyze the intermediate code for the following code	(16)		BTL4

	segment along with the required syntax directed translation scheme
	while (i < 10)
	if (i % 2 == 0)
	evensum = evensum + i
	else
	oddsum = oddsum + i
			Analyze	

13.	(i) Show the rules for type checking with example. (ii) Give an algorithm for type inference and polymorphic function.	(8) (8)	Apply	BTL3
14.	Illustrate an algorithm for unification with its operation.	(16)	Apply	BTL3
15.	Explain the SDD for constructing syntax tree for the expression a+b*5	(16)	Evaluate	BTL5
16.	Illustrate in detail about Bottom-up evaluation of S-attribute definitions and explain the evaluation order for SDD	(16)	Apply	BTL3
17.	Explain the following for the arithmetic expression a+-(b+c)* into (i)Syntax tree (ii)Quadruples (iii)Triples (iv)Indirect Triples	(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?		Apply	BTL3
3.	What is heap allocation?		Remember	BTL1
4.	Describe the activation record is pushed onto the stack.		Understand	BTL2
5.	Give 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

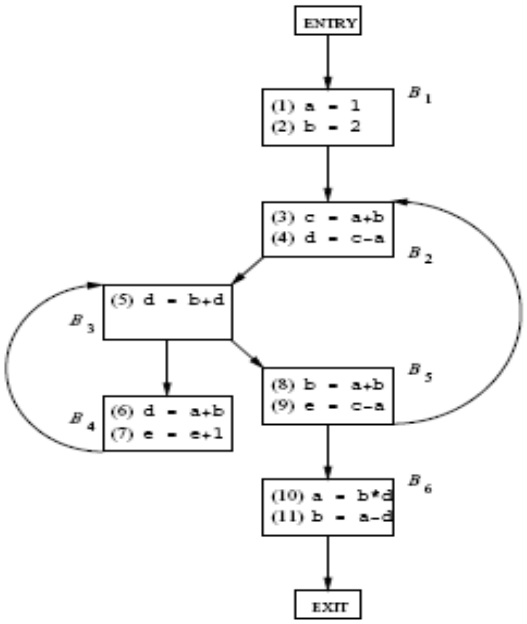
12.	Discuss the main idea of Activation tree.	Understand	BTL2
13.	Give the fields in an Activation record.	Understand	BTL2
14.	Differentiate space efficiency and program efficiency.	Understand	BTL2
15.	Discuss with diagram the typical memory hierarchy configuration of a computer.	Understand	BTL2
16.	Describe the issues in the design of code generators?	Remember	BTL1
17.	Define Best-fit and Next-fit object placement.	Remember	BTL1
18.	Give optimal code sequence for the given sequence t=a+b t=t*c t=t/d	Understand	BTL2
19.	Discuss the different forms of machine instructions.	Understand	BTL2
20.	Discuss the four principle uses of registers in code generation.	Understand	BTL2
21	What is the input to code generator? Discuss	Remember	BTL1
22	What are the advantages and disadvantages of register allocation and assignments?	Understand	BTL2
23	Discus the use of registers is subdivided into 2 sub-problems?	Understand	BTL2
24	Organize the contents of activation record.	Apply	BTL3
	PART-B (13- MARKS)		
1.	(i)Illustrate the storage organization memory in the perspective of compiler writer with neat diagram. (8) (ii)Compare static versus dynamic memory allocation. (8)	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. (11) (ii)Give the structure of the action record. (5)	Create	BTL6
4.	How to a design a call sequences and analyze the principles of activation records with an example? (13)	Analyze	BTL4
5.	Discuss in detail about the activation tree and activation record with suitable example (13)	Understand	BTL2
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)Describe in detail about the memory hierarchy of a computer (8)	Understand	BTL2
8.	Explain fragmentation. Describe in detail about how to (16)	Analyze	BTL4

	reduce the fragment.			
9.	Analyze the following i. Best fit and next object placement. (8) ii. Managing and coalescing free space (8)		Analyze	BTL4
10.	Illustrate 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.	Analyze the heap management of memory manager and locality of programs in detail .	(16)	Analyze	BTL3
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	Discuss in detail about access links, manipulation of access links and access links for procedure	(16)	Understand	BTL2
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.	Describe the concepts of copy propagation.		Understand	BTL 2
3.	State the use of machine Idioms.		Remember	BTL1
4.	Give the flow graph for the quicksort algorithm		Remember	BTL1
5.	List two principal sources of code optimization.		Remember	BTL1
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.	Identify the liveness of a variable .		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.	Identify 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?	Remember	BTL1
17.	Identify the DAG for the statement a= (a*b+c) – (a*b+c) and evaluate it.	Remember	BTL1
18.	Give the code for the follow C statement assuming three registers are available. x = a / (b + c) – d * (e + f)	Understand	BTL2
19.	List the characteristics of peephole optimization.	Remember	BTL1
20.	Define algebraic transformations. Give an example	Understand	BTL2
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	Describe how the strength reduction is applied in code optimization?	Understand	BTL2
PART-B(13 MARKS)			
1.	Explain briefly about the principal sources of optimization. (13)	Evaluate	BTL5
2.	(i).Explain in detail about optimization of basic blocks. (ii).Construct the DAG for the following Basic block & explain it. 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). (8) (8)	Analyze	BTL4
3.	Demonstrate the following in detail (i)Semantic preserving transformation (8) (ii)Global Common subexpression (8)	Apply	BTL3
4.	Illustrate about the following in detail (6) (i)copy propagation (6) (ii)Dead code Elimination (4)	Apply	BTL3

	(iii)code motion			
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 (ii)Discuss the live variable analysis	(8) (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. (ii) Explain in detail about the Use of Algebraic Identities.	(11) (5)	Evaluate	BTL5

10.	Create DAG and three – address code for the following C program. (15)
	i = 1; s = 0;
	while (i<= 10)
	{
	s = s+ a[i] [i];
	i = i + 1;
	}
		(16)	Create	BTL6

11.	 Develop the loops of the flow graph Develop the global common sub expression for each loop Develop Induction variables for each loop Develop loop invariant computation for each loop	(16)	Create	BTL6
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. (ii) Describe about induction variable and end reduction in strength	(8) (8)	Create	BTL6

14.	Describe the efficient data flow algorithms in detail.	(16)	Remember	BTL1
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)	Evaluate	BTL5
17	Illustrate in detail about DAG Representation of basic block and Write algorithm for DAG Construction.	(16)	Apply	BTL3