



# INSTITUTE OF AERONAUTICAL ENGINEERING (Autonomous)

Dundigal, Hyderabad - 500 043

## INFORMATION TECHNOLOGY QUESTION BANK

Course Title	COMPILER DESIGN				
Course Code	ACSC40				
Program	B.Tech				
Semester	V	IT			
Course Type	Core				
Regulation	UG-20				
Course Structure	Theory			Practical	
	Lecture	Tutorials	Credits	Laboratory	Credits
	3	1	4	-	-
Course Coordinator	Dr.U Sivaji, Associate Professor				

### COURSE OBJECTIVES:

The students will try to learn:

I	The process of translating a high-level language to machine code required for compiler construction.
II	The Software tools and techniques used in compiler construction such as lexical analyser and parser generators.
III	The data structures used in compiler construction such as abstract syntax trees, symbol tables, three-address code, and stack machines.
IV	The deeper insights into the syntax and semantic aspects of programming languages, dynamic memory allocation and code generation.

### COURSE OUTCOMES:

After successful completion of the course, students should be able to:

CO 1	<b>Summarize</b> phases of a compiler in the construction of language processors.	Understand
CO 2	<b>Make use of</b> finite automata for designing a lexical analyzer for a specific programming language constructs.	Apply
CO 3	<b>Choose</b> top down, bottom up parsing methods for developing a parser with representation of a parse table or tree.	Apply

CO 4	<b>Outline</b> syntax directed translations, intermediate forms for performing semantic analysis along with code generation.	Understand
CO 5	<b>Relate</b> symbol table, type checking and storage allocation strategies used in run-time environment.	Understand
CO 6	<b>Select</b> code optimization techniques on intermediate code form for generating target code.	Apply

### QUESTION BANK:

Q.No	QUESTION	Taxonomy	How does this subsume the level	CO's
<b>MODULE I</b>				
<b>INTRODUCTION TO COMPILERS</b>				
<b>PART A-PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS</b>				
1	Consider the following fragment of C code: float i, j; i = i*70+j+2; Construct the output at all phases of the compiler for above C code	Apply	This would require the learner to <b>recall</b> the various phases of compiler and <b>explain</b> the task of each phase and <b>apply</b> the concepts for the conversion of the input of each phase for given expression.	CO 1
2	Describe the languages denoted by the following regular expressions. i. $(0+1)^*0(0+1)$ ii. $0^*10^*10^*10^*$	Understand	This would require the learner to <b>recall</b> regular expressions and <b>explain</b> the languages for given expression.	CO 2
3	Explain how LEX program perform lexical analysis to identify Identifiers, Comments, Numerical constants, Keywords, Arithmetic operators?	Understand	This would require the learner to <b>recall</b> software tool such LEX to <b>demonstrate</b> a complete compiler.	CO 2
4	For the following expression $\text{total} = \text{count} + \text{rate} * 5$ Construct the output after each phase of compiler?	Apply	This would require the learner to <b>recall</b> the various phases of compiler and <b>explain</b> the task of each phase and <b>apply</b> the concepts for the conversion of the input of each phase for given expression.	CO 1

5	Convert Regular Expression (b+aa)*a* to Finite Automata.	Understand	This would require the learner to <b>recall</b> the finite automata, regular expressions and <b>show</b> the steps for the conversion of RE to FA.	CO 1
6	Explain the different types of buffering methods used in input buffering	Understand	This would require the learner to <b>recall</b> the buffering and <b>show</b> the types of buffering.	CO 2
7	Contrast single pass and multi pass compiler? What is the effect of reducing the number of passes?	Understand	This would require the learner to recall the compiler phases and show the steps for the reducing number of steps.	CO 2
8	Explain about several steps involved in bootstrapping process for Compilers with neat diagrams	Understand	This would require the learner to <b>recall</b> the bootstrapping and <b>show</b> the steps followed for the process of compilers.	CO 2
9	Write a Lex program to determine whether input is an identifier, keyword, digit and a number.	Understand	This would require the learner to <b>recall</b> the Lex and <b>show</b> the steps for the process of lexical analyzer generator.	CO 2
10	Explain Regular Expressions and outline the transition diagrams for different programming constructs like identifier, number, relation operators.	Understand	This would require the learner to <b>recall</b> the finite automata, regular Expressions, recognition of tokens and <b>write</b> the RE, transition diagrams for programming constructs.	CO 2
<b>PART-B LONG ANSWER QUESTIONS</b>				
1	Draw the structure of a compiler and describe various phases in the compilation process mention the output of the following statement: id1=id2+id3*50 at each phase	Remember	-	CO 1

2	Explain the various phases of a compiler in detail. Also Write down the output for the following expression after each phase $x: = a + b * c - d$ ?	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>discuss</b> the output of each phase for given expression.	CO 1
3	Explain the cousins of a Compiler? Explain them in detail.	Understand	This would require the learner to <b>recall</b> the different phases of compiler, <b>classify</b> the cousins of compiler.	CO 1
4	Describe how various phases could be combined as a pass in compiler?	Remember	-	CO 1
5	Convert Regular Expression $(11+0)^*(00+1)^*$ to Finite Automata.	Understand	This would require the learner to <b>recall</b> the finite automata and <b>show</b> the steps for the conversion of NFA to DFA.	CO 2
6	For the following expression Position:=initial+rate*60,Show the output after each phase of compiler?	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>demonstrate</b> the output of each phase for given expression.	CO 1
7	Explain the role and issues of Lexical Analysis and syntax analysis ?	Understand	This would require the learner to <b>recall</b> the different phases of compiler, <b>classify</b> the cousins of compiler	CO 2
8	Define Regular Expression and its properties. Give examples for Regular Expressions for given Finite Automatas.	Remember	-	CO 2
9	Explain single pass and multi pass compiler with example?	Understand	This would require the learner to <b>recall</b> the different phases of compiler <b>classify</b> the cousins of compiler	CO 1

10	Explain with an example the steps involved in bootstrapping concept and also mention its advantages and disadvantages	Remember	-	CO 1
11	Explain with an example how lex program perform lexical analysis	Understand	This would require the learner to <b>recall</b> software tools such LEX to develop a complete compiler and explain with an example.	CO 2
12	Explain and differentiate frontend and backend of a compiler	Understand	This would require the learner to <b>recall</b> the different phases of compiler and <b>discuss</b> frontend and backend.	CO 1
13	For the following expression a[index]=4+2 Explain output after each phase of compiler?	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>demonstrate</b> the output of each phase for given expression.	CO 1
14	How to specify the Tokens? Differentiate Token, Lexeme and Pattern with suitable examples.	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>show</b> the output of each phase.	CO 2
15	Convert Regular Expression $01^* + 1$ to Finite Automata.	Understand	This would require the learner to <b>recall</b> the finite automata, regular expressions and <b>show</b> the steps for the conversion of RE to FA.	CO 2
16	Compare compiler and interpreter.	Understand	<b>recall</b> components of a language processing system for the <b>convert</b> high level languages to machine level languages.	CO 1
17	Explain the properties of strings and languages.	Understand	This would require the learner to <b>recall</b> the finite automata and <b>write</b> about strings and languages.	CO 2

18	State the reasons for separating Lexical analysis and Syntax analysis.	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>discuss</b> the lexical and syntax analyzers.	CO 1
19	Explain the reasons for separating scanner from parser.	Understand	This would require the learner to <b>recall</b> the various phases of compiler and <b>discuss</b> the scanner, parser	CO 1
20	Compare the pass and phase in compiler construction?	Understand	This would require the learner to <b>recall</b> the different phases of compiler <b>classify</b> the cousins of compiler	CO 1
<b>PART-C SHORT ANSWER QUESTIONS</b>				
1	Name the cousins of compiler?	Remember	-	CO 1
2	What are the parts of compiler in compiler construction?	Remember	-	CO 1
3	How many phases does analysis phase consists define it?	Remember	-	CO 1
4	Define linker and loader.	Remember	-	CO 1
5	Define preprocessor?	Remember	-	CO 1
6	What are the phases of compiler	Remember	-	CO 1
7	Define lexeme and token?	Remember	-	CO 2
8	What are the issues of lexical analyzer?	Remember	-	CO 2
9	List some compiler construction tools?	Remember	-	CO 1
10	Define the term Symbol table?	Remember	-	CO 1
11	Define the term Interpreter?	Remember	-	CO 1
12	Define an error Handler in compiler?	Remember	-	CO 1
13	Explain a translator and types of translator?	Understand	<b>recall</b> components of a language processing system for the <b>convert</b> high level languages to machine level languages.	CO 1

14	Define parser and What are the different types of LR parsers?	Remember	-	CO 1
15	Explain bootstrap?	Understand	This would require the learner to <b>recall</b> (knowledge) basic components of a language processing system to <b>construct</b> tokens, lexemes, different symbols and their importance and applicability in implementing the lexical analyser	CO 1
16	Define pass?	Remember	-	CO 1
17	Define phase?	Remember	-	CO 1
18	What is cross compiler?	Remember	-	CO 1
19	Define multi pass compiler?	Remember	-	CO 1
20	Define DFA,NFA,Regular Expressions.	Remember	-	CO 2
<b>MODULE II</b>				
<b>SYNTAX ANALYSIS</b>				
<b>PART-A PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS</b>				
1	Consider the grammar given below. $E \rightarrow E+T \mid T$ , $T \rightarrow T * F \mid F$ , $F \rightarrow (E) \mid id$ . construct LR parsing table for the above grammar .Give the moves of LR parser on $id * id + id$ ?	Apply	This would require the learner to <b>recall</b> the LR(0) grammar and LR(1) grammar, and demonstrate the rules for given grammar and to <b>construct</b> the LR parsing table	CO 3
2	Identify whether the following grammar is LR(0) with reasons? $S \rightarrow xAy \mid xBy \mid xAz$ , $A \rightarrow as \mid q$ , $B \rightarrow q$	Understand	This would require the learner to <b>recall</b> the LR(0) grammar and <b>demonstrate</b> the rules for given grammar and find out the grammar is LR(0) or not with specific reasons.	CO 3

3	Construct CLR parsing table for the below grammar? $S \rightarrow Aa \mid aAc \mid Bc \mid bBa$ , $A \rightarrow d$ , $B \rightarrow d$	Apply	This would require the learner to <b>recall</b> the CLR parsing method and <b>demonstrate</b> the rules for given grammar and to <b>construct</b> the CLR parsing table	CO 3
4	Identify whether the following grammar is SLR or not with reasons. $S \rightarrow L = R$ , $S \rightarrow R$ , $L \rightarrow *$ $R$ , $L \rightarrow id$ , $R \rightarrow L$ .	Understand	This would require the learner to <b>recall</b> the LR(0) grammar and <b>demonstrate</b> the rules for given grammar and find out the grammar is SLR or not with specific reasons.	CO 3
5	Identify whether the following grammar is CLR or not with reasons? $S \rightarrow AA$ , $A \rightarrow aA \mid b$	Understand	This would require the learner to <b>recall</b> the LR(1) grammar and <b>demonstrate</b> the rules for given grammar and find out the grammar is CLR(1) or not with specific reasons.	CO 3
6	Construct SLR parsing table for the below grammar? $E \rightarrow E+T \mid T$ $T \rightarrow T * F \mid F$ $F \rightarrow (E) \mid id$ .	Apply	This would require the learner to <b>recall</b> the LR(0) grammar and <b>demonstrate</b> the rules for given grammar and to <b>construct</b> the SLR parsing table.	CO 3
7	The following grammar for if-then-else statements is proposed to remedy the dangling-else ambiguity: $Stmt \rightarrow \text{if Expr then Stmt} \mid$ $\text{if Expr then Stmt else Stmt}$ $\mid \text{other}$ Show that how shift and reduce conflicts can be handled in ambiguous grammar.	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the procedure to handle the ambiguous grammar.	CO 3
8	Construct LALR (1) Parsing table for following grammar? $S \rightarrow Aa \mid aAc \mid Bc \mid bBa$ , $A \rightarrow d$ , $B \rightarrow d$	Apply	This would require the learner to <b>recall</b> the LR(1) grammar and <b>demonstrate</b> the rules for given grammar and to <b>construct</b> the LALR parsing table	CO 3



9	Consider the grammar $S \rightarrow aSbS \mid bSaS \mid \epsilon$ a) Construct the corresponding leftmost derivation and rightmost derivation For abab. b) Construct the corresponding parse trees for abab and identify whether the grammar is ambiguous or not.	Apply	This would require the learner to <b>recall</b> context free grammars and <b>explain</b> the procedure to <b>construct</b> derivations.	CO 3
10	Construct the FIRST and FOLLOW sets for following grammar $S \rightarrow aBDh$ , $B \rightarrow cC$ , $C \rightarrow bC / \epsilon$ , $D \rightarrow EF$ , $E \rightarrow g / \epsilon$ , $F \rightarrow f / \epsilon$	Apply	This would require the learner to <b>recall</b> the top down parsing methods and <b>demonstrate</b> the rules for given grammar and to <b>construct</b> the FIRST and FOLLOW.	CO 3
<b>PART-B LONG ANSWER QUESTIONS</b>				
1	List the FIRST and FOLLOW sets for following grammar? $S \rightarrow ACB / CbB / Ba$ , $A \rightarrow da / BC$ , $B \rightarrow g / \epsilon$ , $C \rightarrow h / \epsilon$	Remember	-	CO 3
2	Explain the common conflicts that can be encountered in a shift-reduce parser?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the common conflicts encountered in shift reduce parser and also explain the conflicts with suitable grammar.	CO 3
3	Explain handle pruning in detail with example?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the term handle pruning with example grammar	CO 3

4	Consider the grammar $E \rightarrow E + E \mid E * E \mid (E) \mid id$ Show the sequence of moves made by the shift-reduce parser on the input $(id1+id2)*id3$ and determine whether the given string is accepted by the parser or not	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the acceptance of the string	CO 3
5	Explain the role of stack in shift reduce parsing method?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> role of stake in shift reduce parsing method.	CO 3
6	Explain YACC-automatic parser generator.	Understand	This would require the learner to <b>recall</b> context free grammar and <b>explain</b> how CFG is represented in YACC	CO 3
7	State the difference between SLR,CLR and LALR parsers in detail?	Remember	-	CO 3
8	Explain briefly about panic mode and phrase level error recovery techniques?	Remember	-	CO 3
9	Explain how to handle the error in ambiguous grammar with example?	Understand	This would require the learner to <b>recall</b> context free grammar and <b>explain</b> how to handle error in ambiguous grammar with example grammar.	CO 3
10	Outline the LR Parsing model and write the LR parsing algorithm for constructing the parsing table?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the components in the LR parsing diagram.	CO 3
11	Consider the grammar $P \rightarrow E, E \rightarrow E+T, E \rightarrow T,$ $T \rightarrow id(E) \ T \rightarrow id$ And, state whether the following grammar is LR(0) or not?	Remember	-	CO 3

12	Write shift reduce parsing algorithm and show shift and reduce moves with an example?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the shift reduce parsing method.	CO 3
13	Explain the following terms i) Canonical collection of items ii) Augmented Grammar iii) Closure and go to Operation	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> operations performed on LR parsing techniques such as Augmented grammar, closure and goto operations along with LR(0) items.	CO 3
14	Consider the grammar $P \rightarrow E, E \rightarrow E+T, E \rightarrow T$ $T \rightarrow id (E) T \rightarrow id$ And, State whether the following grammar is SLR(1) or not?	Remember	-	CO 3
15	Outline the CLR Parsing model and write the CLR parsing algorithm for constructing the parsing table?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the steps in algorithm for the construction of CLR parsing tabl	CO 3
16	Explain the SLR(1) parsing table for the following grammar $S \rightarrow Aa \mid bAc \mid dc \mid bd, A \rightarrow d$	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the steps in algorithm for the construction of SLR parsing table	CO 3
17	Compare LR parsers in detail?	Understand	TThis would require the learner to <b>recall</b> bottom up parsing technique and <b>compare</b> the LR parsing methods.	CO 3
18	Consider the grammar $S \rightarrow AS \mid b, A \rightarrow SA \mid a$ Explain the collection of sets of LR(0) items for this grammar?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the steps in algorithm for writing the LR items	CO 3

19	Show that the following grammar $S \rightarrow AaAb \mid BbBa$ $A \rightarrow \epsilon$ $B \rightarrow \epsilon$ is SLR(1) or not?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the steps in algorithm for the construction of SLR parsing table and specify is SLR or not.	CO 3
20	Consider the grammar $bexpr \rightarrow bexpr \text{ or } bterm \mid bterm bterm \rightarrow bterm$ and $bfactor \mid bfactor$ $bfactor \rightarrow \text{not} bfactor \mid (bexpr) \text{ — true } \mid \text{false}.$ Explain whether the grammar is CLR or not?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the procedure to check the grammar is CLR or not.	CO 3
<b>PART-C SHORT ANSWER QUESTIONS</b>				
1	Define about FIRST and state its rules?	Remember	-	CO 3
2	Define about FOLLOW and state its rules?	Remember	-	CO 3
3	Define LR(0) items in bottom up parsing?	Remember	-	CO 3
4	What LR(k) parsing stands for?	Remember	-	CO 3
5	List types of bottom up parsing techniques?	Remember	-	CO 3
6	Define goto function and closure function in LR parser?	Remember	-	CO 3
7	Why SLR and LALR are more economical to construct Canonical LR?	Remember	-	CO 3
8	Tell about handle pruning?	Remember	-	CO 3
9	What are error recovery types?	Remember	-	CO 3
10	List down the conflicts during shift-reduce parsing.	Remember	-	CO 3
11	List out the types LR(0) and LR(1) parsers?	Remember	-	CO 3

12	Describe about shift reduce parsing?.	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> basic idea of shift reduce parsing.	CO 3
13	Define YACC parser?	Remember	CO 3	
14	Compare CLR and LALR?	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>compare</b> between two bottom up parsing techniques CLR and LALR	CO 3
15	Define an augmented grammar?	Remember	-	CO 3
16	Define shift action?	Remember	-	CO 3
17	Define Reduce action?.	Remember	-	CO 3
18	Is left recursion elimination is required in bottom up parsing ?justify.	Understand	This would require the learner to <b>recall</b> bottom up parsing technique and <b>explain</b> the left recursion and then justify whether left recursion elimination is required for bottom up parsing or not.	CO 3
19	List out differences between LL and LR parsers?	Remember	-	CO 3
20	List out the operations of shift reduce parsing?	Remember	-	CO 3
<b>MODULE III</b>				
<b>SYNTAX-DIRECTED TRANSLATION AND INTERMEDIATE CODE GENERATION</b>				
<b>PART A-PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS</b>				
1	Construct production rules and semantic actions for S-attributed grammar for the following grammar along with syntax tree and annotated parse tree for the given string a*b-c/d+e? L→E E→ E+T   E-T T T→ T*F   T/F  F F→ P-F   P P→(E) P→ID	Apply	This would require the learner to <b>recall</b> Syntax-Directed Translation and explain the rules for the conversions and <b>Construct</b> the S-attributed grammar.	CO 4

2	Construct production rules and semantic actions for the following grammar along with annotated parse tree for the string 9-5+4? $\text{expr} \rightarrow \text{expr} + \text{term}$ $  \text{expr} - \text{term}   \text{term}$ $\text{term} \rightarrow 0 1 2 3 4 5 6 7 8 9$	Apply	This would require the learner to recall Syntax-Directed Translation and <b>explain</b> the rules for the conversions and <b>Construct</b> the parse tree.	CO 4
3	Construct production rules and semantic actions for the following grammar along with annotated parse tree for the expression: "int a, b, c"? $D \rightarrow T L$ $T \rightarrow \text{int}$ $T \rightarrow \text{float}; L \rightarrow L1, \text{id} L \rightarrow \text{id}$	Apply	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the rules for the conversions and <b>Construct</b> the parse tree.	CO 4
4	Construct production rules and semantic actions for the following grammar along with annotated parse tree for the string (3+4)*(5+6)? $L \rightarrow E$ $E \rightarrow T E \rightarrow E1 + T$ $T \rightarrow F T \rightarrow T1 * F F \rightarrow (E)$ $F \rightarrow \text{digit}$	Apply	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the rules for the conversions and <b>Construct</b> the parse tree	CO 4
5	Construct production rules and semantic actions for the following grammar along with annotated parse tree for the string a- 4+c? $E \rightarrow E1 + T$ $E \rightarrow E1 - T$ $E \rightarrow T T \rightarrow (E)$ $T \rightarrow \text{id}$ $T \rightarrow \text{num}$	Apply	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the rules for the conversions and <b>Construct</b> the parse tree	CO 4
6	Construct the three address code and draw the abstract tree for the following expressions? a) $(x-y)*z+m-n$ b) $a+(b-c)+(b+c)*(a*e)$	Apply	This would require the <b>recall</b> Intermediate Code Generation and <b>explain</b> the concepts to <b>construct</b> the three address code.	CO 4

7	Construct the three-address code for the following C program fragment? while(a >b) { if (c <d) x = y +z; else x = y -z; }	Apply	This would require the <b>recall</b> Intermediate Code Generation and <b>explain</b> the concepts to <b>construct</b> the three address code.	CO 4
8	Construct triples, Indirect and quadruples of an expression: $a = b * -c + b * -c$ ?	Apply	This would require the <b>recall</b> Intermediate Code Generation and <b>explain</b> the concepts to <b>construct</b> the triples, Indirect and quadruples.	CO 4
9	Construct triples, Indirect and quadruples of an expression : $x = (a + b)^* - c/d$ ?	Apply	This would require the <b>recall</b> Intermediate Code Generation and <b>explain</b> the concepts to <b>construct</b> the triples, Indirect and quadruples.	CO 4
10	Why are quadruples preferred over triples in an optimizing compiler with example?	Remember	–	CO 4
<b>PART-B LONG ANSWER QUESTIONS</b>				
1	Explain briefly about syntax directed definition and it types?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the types of SDD	CO 4
2	Explain briefly about Synthesized and Inherited attribute in detail?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the attributes.	CO 4
3	Define translation scheme and write for $a < b$ or $b > c$ ?	Remember	-	CO 4
4	Explain briefly about S-attributed and L-attributed grammar in detail?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the attributes.	CO 4

5	Explain how declaration is done in a procedure using syntax directed translation?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the SDT for procedure.	CO 4
6	Explain briefly about postfix Translation Scheme?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the postfix Translation Scheme	CO 4
7	Describe the method of generating syntax directed definition for control Statements?	Remember	-	CO 4
8	Show SDT for the simple assignment statement with example?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the SDT for simple assignment statement	CO 4
9	Explain the construction steps and construct the syntax tree for expression using functions? $(m * n + p) + (m - n + p)$ ?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the types of three address code.	CO 4
10	Explain briefly syntax directed translation into three address code with suitable example?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the conversion of SDT to three address code.	CO 4
11	Explain three address codes and mention its types. How would you implement the three address statements? Explain with suitable examples?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the three address code forms.	CO 4



12	Explain with an example to generate the intermediate code for the flow of control statements?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the generation of intermediate code for the flow of control statements.	CO 4
13	Explain about Quadruple and Triple with its structure?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the quadruple and triple.	CO 4
14	Define and represent the Triple, indirect triple and quadruple for the assignment statement? $x := -b + d * -b + d$	Remember	-	CO 4
15	Translate the arithmetic expression $a * - (b + c)$ into a) A syntax tree b) Postfix notation ?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the arithmetic expression translation into syntax tree and Postfix notation .	CO 4
16	Translate the expression $-(a + b) * (c + d) + (a + b + c)$ into a) quadruples b) triples	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the arithmetic expression translation into triple, indirect triple and quadruple.	CO 4
17	Show translation scheme for Boolean Expressions with example?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the Boolean Expression translation scheme.	CO 4
18	Show translation scheme for Control Flow with example?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the Control Flow.	CO 4

19	Translate the expression – $(a + b) * (c + d) + (a + b + c)$ into a) triples b) indirect triples.	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the arithmetic expression translation into triple, indirect triple.	CO 4
20	Explain the three address code and draw the abstract tree for the following expressions? $(a-b)*c+m-n$	Understand	This would require the <b>recall</b> Intermediate Code Generation and <b>explain</b> the three address code.	CO 4
<b>PART-C SHORT ANSWER QUESTIONS</b>				
1	What is the usage of syntax directed definition?	Remember	-	CO 4
2	Define Attribute Grammar?	Remember	-	CO 4
3	List the types of Attribute Grammar?	Remember	-	CO 4
4	Explain syntax directed translation?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the translations.	CO 4
5	Compare synthesized and inherited attributes?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the synthesized and inherited attributes.	CO 4
6	Define L attributed grammar?	Remember	-	CO 4
7	Define S attribute grammar?	Remember	-	CO 4
8	show the Syntax tree for Expression using functions? $(a + b) * (b - c)$	Remember	-	CO 4
9	Explain the functions to create nodes of Syntax tree for expression?	Understand	This would require the learner to <b>recall</b> Syntax-Directed Translation and <b>explain</b> the nodes of Syntax tree.	CO 4
10	Define syntax tree? Draw the syntax tree for the assignment statement? $a := b * -c + b * -c.$	Remember	-	CO 4

11	Define Translation schemes?	Remember	-	CO 4
12	Define Annotated Parse Tree?	Remember	-	CO 4
13	Explain the three kinds of intermediate representation?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>explain</b> the intermediate forms	CO 4
14	What are the benefits of using machine-independent intermediate form?	Remember	-	CO 4
15	What is postfix notation?	Remember	-	CO 4
16	How can you generate three-address code?	Remember	-	CO 4
17	Translate $x+y-(a*b)+c$ into three address code?	Understand	This would require the learner to <b>recall</b> Intermediate Code Generation and <b>convert</b> to the three address code form.	CO 4
18	Discuss back-end and front-end?	Understand	This would require the learner to <b>recall</b> phases of a compiler and explain the back-end and front-end.	CO 4
19	Define abstract syntax tree?	Remember	-	CO 4
20	List out types of three address code?	Remember	-	CO 4

#### MODULE IV

#### TYPE CHECKING AND RUN TIME ENVIRONMENT

#### PART A- PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS

1	Suppose that the type of each identifier is a sub range of integers, for expressions with operators +, -, *, div and mod, as in Pascal. Explain type- checking rules that assign to each sub expression the sub range its value must lie in?	Understand	This would require the learner to <b>recall</b> type checking <b>explain</b> how type checking rules implemented for each type of identifier	CO 5
2	Explain briefly about Source language issues?	Understand	This would require the learner to <b>recall</b> type checking <b>explain</b> about Source language issues	CO 5

3	Explain briefly about Activation record with block diagram?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the Activation record with block diagram	CO 5
4	Discuss about variable length data on stack with neat diagram?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the variable length data on stack	CO 5
5	Explain briefly about heap storage allocation with block diagram?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the heap storage allocation	CO 5
6	Explain briefly about stack storage allocation with block diagram?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the stack storage allocation	CO 5
7	Explain briefly about language facilities for dynamic storage allocation?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the language facilities for dynamic storage allocation	CO 5
8	Describe the parameter passing methods with examples?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the various parameter passing methods.	CO 5
9	Explain Over loading of Operators and Functions with examples?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the Over loading of Operators and Functions	CO 5
10	Differentiate the call by reference and call by copy restore with examples?	Understand	This would require the learner to <b>recall</b> run time environment <b>explain</b> the call by reference and call by copy restore.	CO 5
<b>PART-B LONG ANSWER QUESTIONS</b>				
1	Explain the specification of a simple type checker	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the specification of a simple type checker	CO 5

2	Define a type expression? Explain the equivalence of type expressions with an appropriate example?	Remember	—	CO 5
3	Explain about reusing the storage space for names?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the procedure for using storage space efficiently.	CO 5
4	Discuss about all allocation strategies in run-time storage environment?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the allocation strategies.	CO 5
5	Explain the data structures used for implementing Symbol Table?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the Symbol Table implementation	CO 5
6	Explain Static and Dynamic Checking of types with examples?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the Static and Dynamic Checking	CO 5
7	Compare the call by value and call by name with examples?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>compare</b> the call by value and by name	CO 5
8	Distinguish between static and dynamic storage allocation?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the Static and Dynamic storage allocation	CO 5
9	Explain the type checking of expressions?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the procedure for expressions	CO 5
10	Explain storage organization in runtime environment?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the storage organization	CO 5

11	Explain the types of storage allocations?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the storage allocations types	CO 5
12	Describe the name and structure equivalence in type expressions?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the name and structure equivalence in type expressions	CO 5
13	Explain the type checking of control flow statements?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the steps for control flow statements	CO 5
14	Explain briefly about storage allocation strategies?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the storage allocations strategies	CO 5
15	Describe the basic implementation techniques for symbol table?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the symbol table implementation	CO 5
16	Explain the calling sequences of activation record?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the calling sequences.	CO 5
17	Differentiate ordered, unordered and binary search tree in symbol table?	Understand	This would require the learner to the <b>recall</b> Run Time Environment and <b>explain</b> binary search tree in symbol table	CO 5
18	Explain briefly about static storage allocation with block diagram?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the static storage allocation	CO 5
19	Differentiate explicit and implicit allocation of memory to variables?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>compare</b> the allocation types of memory to variables	CO 5

20	Differentiate stack and heap storage allocation strategies?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the stack and heap storage allocation strategies.	CO 5
<b>PART-C SHORT ANSWER QUESTIONS</b>				
1	List different data structures used for symbol table?	Remember	—	CO 5
2	Define Type checking?	Remember	—	CO 5
3	List the different types of type checking?	Remember	—	CO 5
4	Define Type Expression?	Remember	—	CO 5
5	Explain about the type systems?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the type systems	CO 5
6	Show the Translation scheme for checking the type of Assignment statement $S \rightarrow id := E$	Remember	—	CO 5
7	Explain Dynamic type checking?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the Dynamic type checking	CO 5
8	Define Structural Equivalence?	Remember	—	CO 5
9	What is the Strongly typed language?	Remember	—	CO 5
10	Define Type error?	Remember	—	CO 5
11	Write a short note on static type checking?	Understand	This would require the learner to <b>recall</b> type checking and <b>explain</b> the static type checking	CO9
12	Show the Translation scheme for checking the type of Conditional statement - $S \rightarrow \text{if } E \text{ then } S1$	Remember	—	CO 5
13	Show the Translation scheme for checking the type of while statement $S \rightarrow \text{While } E \text{ do } S1$	Remember	—	CO 5
14	Define Type conversion?	Remember	—	CO 5

15	List the types of type conversion?	Remember	—	CO 5
16	Write about general activation record?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the general activation record.	CO 5
17	Define Symbol table?	Remember	—	CO 5
18	Define Dynamic storage allocation?	Remember	—	CO 5
19	Write short note on procedures?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the procedures	CO 5
20	Define Activation tree?	Remember	—	CO 5
21	Define stack storage allocation	Remember	—	CO 5
22	Define static storage allocation?	Remember	—	CO 5
23	Define heap storage allocation?	Remember	—	CO 5
24	Write a short note on parameter passing?	Understand	This would require the learner to <b>recall</b> Run Time Environment and <b>explain</b> the parameter passing	CO 5
25	Define Control stack?	Remember	—	CO 5

#### MODULE V

#### CODE OPTIMIZATION AND CODE GENERATION

#### PART A-PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)

1	Construct the code sequence generated by the simple code generation algorithm for $x*y+(m-k)-(g+b)$	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the code sequence.	CO 6
2	Construct target code for the given program segments: main() { int i=4,j; j = i + 5; }	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the target code.	CO 6



3	Consider the following basic block of 3-address instructions .Construct target code for the source language statement and finds its cost. a := b + c x := a + b b := a- d c := b + c d := a -d y := a -d	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the target code.	CO 6
4	Identify the register descriptor target code for the source language Statement and its cost. (a-b) + (a-c) + (a-c)	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the target code.	CO 6
5	Consider the following part of code. int main() { int n,k=0; scanf("%d",&n); for(i=2;i<n;i++) { if(n%i,==0)break; } k=1; if(i==n) printf(number is prime); else printf(number is not printed); } Identify the basic block in the given program	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the basic blocks.	CO 6
6	Construct the DAG for the following basic block. D:=B*C E:=A+B B:=B+C A:=E-D	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the DAG.	CO 6

7	Construct basic block for following code <pre> void quicksort(m, n) int m, n; { int i, j; if (n &lt;= m ) return; /* fragment begins here */ i = m-1; j = n; v = a[n]; while(1) { do i = i+1; while( a[i] &lt;v); do j = j-1; while( a[j] &gt;v ); if( i &gt;= j ) break; x = a[i]; a[i] = a[j]; a[j] =x; } x = a[i]; a[i] = a[n]; a[n]= x; /* fragment ends here */ quicksort(m, j); quicksort(i+1, n); }</pre>	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the basic blocks.	CO 6
8	Construct DAG and explain the procedure for the conversion. $a+b*(a+b)+c+d$	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the DAG.	CO 6
9	Explain role of DAG representation in optimization with example?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the DAG and its role in optimization.	CO 6

10	Construct the basic block and flow graph for the following code begin prod :=0; i:=1; do begin prod :=prod+ a[i] * b[i]; i :=i+1; end while i <= 20 end	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the basic blocks.	CO 6
11	Construct optimal machine code for the following C program. main() { inti,a[10]; while(i<=10) a[i]=0; }	Apply	This would require the learner to <b>recall</b> code generation and <b>explain</b> the procedure to <b>Construct</b> the optimal machine code.	CO 6
<b>PART-B LONG ANSWER QUESTIONS</b>				
1	Explain the concept of Function-Preserving Transformations?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the transformations.	CO 6
2	Explain Machine dependent code optimization in detail with an example?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the Machine dependent code optimization.	CO 6
3	Write about target code forms and Explain how the instruction forms effect the computation time?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the how the instruction forms effect the computation time	CO 6
4	Explain about machine dependent and machine independent optimization?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the machine dependent and machine independent optimization	CO 6

5	Explain the role of code generator in a compiler?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the code generator in a compiler.	CO 6
6	Explain in detail the issues in the design of code generator?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the issues in the design of code generator	CO 6
7	Explain the instructions and address modes of the target machine?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the instructions and address modes of the target machine	CO 6
8	Explain the principle sources of code optimization in detail?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the sources of code optimization	CO 6
9	Explain the primary structure preserving transformations on basic blocks?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the structure preserving transformations	CO 6
10	Explain peephole optimization in detail?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the peephole optimization	CO 6
11	Define the following i. Copy propagation ii. Dead code elimination	Remember	—	CO 6
12	Explain in the DAG representation of the basic block with example?	Understand	This would require the learner to <b>recall</b> DAG and <b>Explain</b> the representation of the basic blocks	CO 6
13	Explain loop optimization in detail with example?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the loop optimizations	CO 6
14	Explain various Global optimization techniques in detail?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the global optimizations	CO 6

15	Explain Loops in flow graph in detail with example?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the flow graphs	CO 6
16	Explain Local optimization in detail with example?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the local optimization	CO 6
17	Explain Redundant-instructions elimination and Flow-of-control Optimizations?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> optimizations.	CO 6
18	Explain the simple code generator with a suitable example?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the simple example for code generator.	CO 6
19	Explain the procedure to detect induction variable and dead code elimination with example?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> optimizations.	CO 6
20	Explain briefly about register allocation and assignment?	Understand	This would require the learner to <b>recall</b> code generation and <b>explain</b> the register allocation.	CO 6
<b>PART-C SHORT ANSWER QUESTIONS</b>				
1	List the principle sources of optimization?	Remember	—	CO 6
2	Define the 3 areas of code optimization?	Remember	—	CO 6
3	Explain the techniques used for loop optimization and Reduction in strength?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the loop optimization methods.	CO 6
4	Define constant folding?	Remember	—	CO 6
5	Define Common Sub expressions?	Remember	—	CO 6
6	Explain Dead Code?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the concept of dead code.	CO 6
7	Define local optimization?	Remember	—	CO 6

8	What is Register allocation and assignment?	Remember	—	CO 6
9	Define flow graph and basic block?	Remember	—	CO 6
10	Explain about inner loops?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the flow graph representation to find the inner loops.	CO 6
11	Define a DAG? Mention its Remember?	Remember	—	CO 6
12	Define peephole optimization?	Remember	—	CO 6
13	Define the machine instructions for operations and copy statement?	Remember	—	CO 6
14	Explain global data flow?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the global data flow.	CO 6
15	Explain about live variable analysis?	Understand	This would require the learner to <b>recall</b> code optimization and <b>Explain</b> the live variable analysis.	CO 6
16	Define the term copy propagation?	Remember	—	CO 6
17	Define the term Code motion?	Remember	—	CO 6
18	What is induction variable?	Remember	—	CO 6
19	How do you calculate the cost of an instruction?	Remember	—	CO 6
20	What is the Unreachable Code?	Remember	—	CO 6

Prepared By:  
Dr.U Sivaji, Associate Professor

HOD IT