INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad-500043

INFORMATION TECHNOLOGY

TUTORIAL QUESTION BANK

<table>
<tr><td><b>Course Title</b></td><td colspan="5"><b>COMPILER DESIGN</b></td></tr>
<tr><td><b>Course Code</b></td><td colspan="5">AIT004</td></tr>
<tr><td><b>Programme</b></td><td colspan="5">B.Tech</td></tr>
<tr><td><b>Semester</b></td><td>V</td><td colspan="4">CSE | IT</td></tr>
<tr><td><b>Course Type</b></td><td colspan="5">Core</td></tr>
<tr><td><b>Regulation</b></td><td colspan="5"><b>IARE - R16</b></td></tr>
<tr><td rowspan="3"><b>Course Structure</b></td><td colspan="3"><b>Theory</b></td><td colspan="2"><b>Practical</b></td></tr>
<tr><td><b>Lectures</b></td><td><b>Tutorials</b></td><td><b>Credits</b></td><td><b>Laboratory</b></td><td><b>Credits</b></td></tr>
<tr><td>3</td><td>1</td><td>4</td><td>-</td><td>-</td></tr>
<tr><td><b>Chief Coordinator</b></td><td colspan="5">Dr. K Srinivasa Reddy, Professor</td></tr>
<tr><td><b>Course Faculty</b></td><td colspan="5">Y Harika, Assistant Professor</td></tr>
</table>

COURSE OBJECTIVES:

The course should enable the students to:	
I	Apply the principles of theory of computation to the various stages in the design of compilers.
II	Demonstrate the phases of the compilation process and able to describe the purpose and operation of each phase.
III	Analyze problems related to the stages in the translation process.
IV	Exercise and reinforce prior programming knowledge with a non-trivial programming project to construct a compiler.

COURSE OUTCOMES (COs):

CO 1	Understand the various phases of compiler and design the lexical analyzer
CO 2	Explore the similarities and differences among various parsing techniques and grammar transformation techniques.
CO 3	Analyze and implement syntax directed translations schemes and intermediate code generation.
CO 4	Describe the concepts of type checking and analyze runtime allocation strategies.
CO 5	Demonstrate the algorithms to perform code optimization and code generation.

COURSE LEARNING OUTCOMES (CLOs):

AIT004.01	Define the phases of a typical compiler, including the front and backend.
AIT004.02	Recognize the underlying formal models such as finite state automata, push-down automata and their connection to language definition through regular expressions and grammars.
AIT004.03	Identify tokens of a typical high-level programming language; define regular expressions for tokens and design and implement a lexical analyzer using a typical scanner generator.
AIT004.04	Explain the role of a parser in a compiler and relate the yield of a parse tree to a grammar derivation
AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; construct a parser for a given context-free grammar.
AIT004.06	Demonstrate Lex tool to create a lexical analyzer and Yacc tool to create a parser.
AIT004.07	Understand syntax directed translation schemes for a given context free grammar.
AIT004.08	Implement the static semantic checking and type checking using syntax directed definition (SDD) and syntax directed translation (SDT).
AIT004.09	Understand the need of intermediate code generation phase in compilers.
AIT004.10	Write intermediate code for statements like assignment, conditional, loops and functions in high level language.
AIT004.11	Explain the role of a semantic analyzer and type checking; create a syntax-directed definition and an annotated parse tree; describe the purpose of a syntax tree.
AIT004.12	Design syntax directed translation schemes for a given context free grammar.
AIT004.13	Explain the role of different types of runtime environments and memory organization for implementation of programming languages.
AIT004.14	Differentiate static vs. dynamic storage allocation and the usage of activation records to manage program modules and their data..
AIT004.15	Understand the role of symbol table data structure in the construction of compiler.
AIT004.16	Learn the code optimization techniques to improve the performance of a program in terms of speed & space.
AIT004.17	Implement the global optimization using data flow analysis such as basic blocks and DAG.
AIT004.18	Understand the code generation techniques to generate target code.
AIT004.19	Design and implement a small compiler using a software engineering approach.
AIT004.20	Apply the optimization techniques to intermediate code and generate machine code.

TUTORIAL QUESTION BANK

UNIT- I				
INTRODUCTION TO COMPILERS AND PARSING				
Part - A (Short Answer Questions)				
S No	QUESTIONS	Blooms Taxonomy Level	Course Outcomes	Course Learning Outcomes (CLOs)
1	Explain the cousins of compiler?	Understand	CO 1	AIT004.01
2	Define the two main parts of compilation? What they perform?	Understand	CO 1	AIT004.01
3	How many phases does analysis phase consists define it?	Understand	CO 1	AIT004.01
4	Define and explain the Loader?	Remember	CO 1	AIT004.01
5	Write about preprocessor?	Remember	CO 1	AIT004.01
6	State the general phases of a compiler?	Understand	CO 1	AIT004.01
7	Define a lexeme and token?	Remember	CO 1	AIT004.01
8	List the issues of lexical analyzer?	Understand	CO 1	AIT004.01
9	State some compiler construction tools?	Understand	CO 1	AIT004.01
10	Define the term Symbol table?	Understand	CO 1	AIT004.01
11	Define the term Interpreter?	Remember	CO 1	AIT004.03
12	How would you Write about error Handler?	Understand	CO 1	AIT004.01
13	Define a translator and types of translator?	Understand	CO 1	AIT004.01
14	Define parser and list its types?	Understand	CO 1	AIT004.01
15	Define bootstrap and cross compiler?	Understand	CO 1	AIT004.01
16	Define pass and phase?	Understand	CO 1	AIT004.01
17	Analyze the output of syntax analysis phase? What are the three general types of parsers for grammars?	Remember	CO 1	AIT004.01
18	What are the goals of error handler in a parser?	Understand	CO 1	AIT004.01
19	Define context free grammar. When will you say that two CFGs are equal?	Remember	CO 1	AIT004.02
20	Give the definition for leftmost and rightmost derivations?	Understand	CO 1	AIT004.02
21	Define a parse tree?	Understand	CO 1	AIT004.02
22	Explain an ambiguous grammar with an example?	Remember	CO 1	AIT004.02
23	When will you call a grammar as the left recursive one?	Remember	CO 1	AIT004.02
24	Define elimination of left factoring?	Remember	CO 1	AIT004.05
25	Define back tracking?	Understand	CO 1	AIT004.05
26	Define topdown parsing and its types?	Understand	CO 1	AIT004.05
27	Write about recursive descent parsing?	Understand	CO 1	AIT004.05
28	Write about predictive parser?	Understand	CO 1	AIT004.05
29	Define about FIRST and state its rules?	Remember	CO 1	AIT004.05
30	Define about FOLLOW and state its rules?	Remember	CO 1	AIT004.05
31	State the condition to check the grammar is LL(1) or not?	Remember	CO 1	AIT004.05
32	Write down the difficulties in top down parsing.?	Understand	CO 1	AIT004.05
33	How to eliminating ambiguity from dangling-else grammar?	Remember	CO 1	AIT004.05
Part - B (Long Answer Questions)				
1	Define compiler? State various phases of a compiler and explain them in detail?	Understand	CO 1	AIT004.01
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?	Remember	CO 1	AIT004.01
3	Explain the cousins of a Compiler? Explain them in detail.	Understand	CO 1	AIT004.01
4	Describe how various phases could be combined as a pass in acompiler?	Understand	CO 1	AIT004.01
5	For the following expression Position:=initial+ rate*60 Write down the output after each phase?	Remember	CO 1	AIT004.01

6	Explain the role Lexical Analyzer and issues of Lexical Analyzer?	Understand	CO 1	AIT004.01
7	Differentiate the pass and phase in compiler construction?	Understand	CO 1	AIT004.01
8	Explain single pass and multi pass compiler? with example?	Understand	CO 1	AIT004.01
9	Define bootstrapping concept in brief?	Understand	CO 1	AIT004.03
10	Explain the general format of a LEX program with example?	Remember	CO 1	AIT004.06
11	Construct the predictive parser the following grammar: S->(L)|a L->L,S|S. Construct the behavior of the parser on the sentence (a,a) using the above grammar?	Remember	CO 1	AIT004.05
12	State the limitations of recursive descent parser?	Understand	CO 1	AIT004.05
13	Consider the grammar below E→ E+E | E-E | E*E | E/E | a | b Obtain left most and right most derivation for the string a+b*a-b?	Remember	CO 1	AIT004.05
14	Explain problems in topdown parsing along with algorithms and examples?	Understand	CO 1	AIT004.05
15	Find the FIRST and FOLLOW sets for following grammar? S → ACB / CbB / Ba A → da / BC B → g / ∈ C → h / ∈	Remember	CO 1	AIT004.05
16	Explain briefly about compiler construction tools?	Remember	CO 1	AIT004.03
17	Explain briefly left recursion amd left factoring with example?	Understand	CO 1	AIT004:05
18	Differentiate the compiler and interpreter in detail?	Understand	CO 1	AIT004.05
19	Describe the rules for finding FIRST and FOLLOW sets of any context free grammar?	Remember	CO 1	AIT004.05
20	Find the FIRST and FOLLOW sets for following grammar? S→ aBDh B → cC C → bC / ∈ D → EF E → g / ∈ F → f / ∈	Remember	CO 1	AIT004.05
Part - C (Problem Solving and Critical Thinking Questions)				

1	Consider the following fragment of C code:
	float i, j;
	i = i*70+j+2;
	Write the output at all phases of the compiler for above „C‟ code?
		Remember	CO 1	AIT004.01

2	Describe the languages denoted by the following regular expressions. i. (0+1)*0(0+1)(0+1) ii. 0*10*10*10*	Remember	CO 1	AIT004.03
3	Explain how LEX program perform lexical analysis to identify Identifiers, Comments, Numerical constants, Keywords, Arithmetic operators?	Remember	CO 1	AIT004.06
4	Check whether the following grammar is a LL(1)grammar S → iEtS|iEtSeS|a E→ b Also define the FIRST and Follows.	Remember	CO 1	AIT004.05
5	Analyze whether the following grammar is LL(1) or not. Explain your answer with reasons? S→ L,R S → R L → * R L → id R→ L.	Remember	CO 1	AIT004.05

6	Define ambiguous grammar? Test whether the following grammar is ambiguous or not? E → E+E | E-E | E*E | E/E | (E) | id	Remember	CO 1	AIT004.04
7	Prepare the predictive parser for the following grammar: S→ a | b| (T) T →T, S|S Write down the necessary algorithms and define FIRST and FOLLOW. Show the behavior of the parser in the sentences, i. (a,(a,a)) ii. ((a,a),a,(a),a)	Remember	CO 1	AIT004.05
8	Convert the following grammar into LL(1)grammar, S→ ABC A→ aA|C B→ b C→ c.	Remember	CO 1	AIT004.05
9	Write a recursive descent parser for the grammar. bexpr→bexpr or bterm | bterm bterm→bterm and bfactor | bfactor bfactor→not bfactor | (bexpr) |true | false. Where or, and , not,(,),true, false are terminals of the grammar.	Remember	CO 1	AIT004.05
10	Consider the grammar, E → E+T | T T → T*F | F F→ (E) | id. Construct a predictive parsing table for the grammar given above. Verify whether the input string id + (id * id) is accepted by the grammar or not?	Remember	CO 1	AIT004.05
UNIT-II				
BOTTOM-UP PARSING				
Part – A (Short Answer Questions)				
1	Define the term handle?	Understand	CO 2	AIT004.05
2	Define bottomup parsing?	Understand	CO 2	AIT004.05
3	Define LR(0) items in bottom up parsing?	Remember	CO 2	AIT004.05
4	LR(k) parsing stands for what?	Remember	CO 2	AIT004.05
5	List types of bottomup parsing techniques?	Understand	CO 2	AIT004.05
6	Define goto function and closure function in LR parser?	Remember	CO 2	AIT004.05
7	Why SLR and LALR are more economical to construct Cananonical LR?	Understand	CO 2	AIT004.05
8	Write about handle pruning?	Understand	CO 2	AIT004.05
9	What are error recovery types?	Understand	CO 2	AIT004.05
10	List down the conflicts during shift-reduce parsing.	Understand	CO 2	AIT004.05
11	List out the types LR(0) and LR(1) parsers?	Understand	CO 2	AIT004.05
12	Write about shift reduce parsing?	Understand	CO 2	AIT004.05
13	Define YACC parser?	Understand	CO 2	AIT004.06
14	State the difference between CLR and LALR?	Understand	CO 2	AIT004.05
15	Define an augmented grammar?	Remember	CO 2	AIT004.05
16	Define shift action?	Remember	CO 2	AIT004.05
17	Define Reduce action?	Remember	CO 2	AIT004.05
18	Is left recursion elimination is required in bottomup parsing ?justify.	Understand	CO 2	AIT004.05
19	List out difference between LL and LR parsers?	Understand	CO 2	AIT004.05
20	List out the operations of shift reduce parsing?	Remember	CO 2	AIT004.05
Part - B (Long Answer Questions)				
1	Discuss briefly about types of error recovery in parsing?	Remember	CO 2	AIT004.05

2	Explain the common conflicts that can be encountered in a shift-reduce parser?	Understand	CO 2	AIT004.04
3	Explain handle pruning in detail with example?	Understand	CO 2	AIT004.04
4	Consider the grammar E → E + E | E *E | (E) | 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?	Remember	CO 2	AIT004.04
5	Demonstrate stack implementation in implementation of shift reduce Parsing?	Remember	CO 2	AIT004.04
6	Explain briefly about YACC-automatic parser generator?	Remember	CO 2	AIT004.06
7	State the difference between SLR,CLR and LALR parsers in detail?	Remember	CO 2	AIT004.04
8	Explain briefly about panic mode and phrase level error recovery techniques?	Remember	CO 2	AIT004.05
9	Explain how to handle the error in ambiguous grammar with example?	Understand	CO 2	AIT004.05
10	Describe LR Parsing algorithm in detail with diagram?	Understand	CO 2	AIT004.05
11	Consider the grammar, P→ E E → E+T E→ T T→ id(E) T→ id And,check whether the following grammar is LR(0) or not?	Remember	CO 2	AIT004.05
12	Explain briefly about shift reduce parsing algoritm?	Understand	CO 2	AIT004.05
13	Explain the following terms i)Canonical collection of items ii)Augmented Grammar iii)Closure and goto Operation	Understand	CO 2	AIT004.05
14	Consider the grammar, P→ E E → E+T E→ T T→ id(E) T→ id And,check whether the following grammar is SLR(1) or not?	Remember	CO 2	AIT004.05
15	Explain the algorithm for construction of CLR(1) parsing table?	Understand	CO 2	AIT004.05
16	Construct the SLR(1) parsing table for the following grammar S→ Aa | bAc|dc|bd A→d	Remember	CO 2	
17	List out the comparisions of LR parsers in detail?	Remember	CO 2	AIT004.05
18	Consider the grammar S→ AS| b A → SA | a Construct the collection of sets of LR(0) items for this grammar?	Remember	CO 2	AIT004.05
19	Show that the following grammar S→ AaAb | BbBa A→ ∈ B→ ∈ is SLR(1) or not?	Remember	CO 2	AIT004.05
20	Consider the grammar bexpr→bexpr or bterm | bterm bterm→bterm and bfactor | bfactor bfactor→not bfactor | (bexpr) |true | false. Check whether the grammar is CLR or not?	Remember	CO 2	AIT004.05

Part - C (Problem Solving and Critical Thinking Questions)				
1	Consider the grammar given below. E → E+T | T T → T*F | F F→ (E) | id. Prepare LR parsing table for the above grammar .Give the moves of LR parser on id * id + id?	Remember	CO 2	AIT004.04
2	Analyze whether the following grammar is LR(0). Explain your answer with reasons? S → xAy | xBy | xAz A → as | q B → q	Analysis	CO 2	AIT004.04
3	Analyze whether the following grammar is CLR or not. Explain your answer with reasons? S→ Aa | aAc | Bc | bBa A→ d B → d	Remember Analysis	CO 2	AIT004.04
4	Analyze whether the following grammar is SLR or not. Explain? your answer with reasons. S→ L = R S→ R L→ * R L→ id R → L.	Remember Analysis	CO 2	AIT004.04
5	Analyze whether the following grammar is CLR or not. Explain your answer with reasons? S→ AA A →aA | b	Remember Analysis	CO 2	AIT004.05
6	Prepare SLR parsing table for the below grammar? E → E+T | T T → T*F | F F→ (E) | id.	Remember	CO 2	AIT004.05
7	The following grammar for if-then-else statements is proposed to remedy the dangling-else ambiguity: Stmt → **if** expr **then** stmt | matched_stmt Matched_stmt → **if** expr **then** matched_stmt **else** stmt | **other** Show that this grammar is still ambiguous.	Remember Analysis	CO 2	AIT004.05
8	Construct LALR (1) Parsing table for following grammar? S → Aa |aAc | Bc | bBa A → d B → d	Remember	CO 2	AIT004.05
9	Consider the grammar S→ aSbS | bSaS|∈ a) Show that this grammar is ambiguous by constructing two different leftmost derivations for the sentence abab b) Construct the corresponding rightmost derivations for abab. c) Construct the corresponding parse trees for abab.	Remember	CO 2	AIT004.05

10	Consider the grammar S→ AS| b A → SA | a Check whether the given grammar is LALR(1) or not?	Remember	CO 2	AIT004.05
UNIT -III				
SYNTAX-DIRECTED TRANSLATION AND INTERMEDIATE CODE GENERATION				
Part - A (Short Answer Questions)				
1	What is the usage of syntax directed definition?	Understand	CO 3	AIT004.08
2	Define Attribute Grammar?	Understand	CO 3	AIT004.07
3	List the types of Attribute Grammar?	Understand	CO 3	AIT004.07
4	Write a note on syntax directed translation?	Understand	CO 3	AIT004.07
5	State the difference between synthesized and inherited attributes?	Understand	CO 3	AIT004.08
6	Define L attributed grammar?	Remember	CO 3	AIT004.08
7	Define S attribute grammar?	Remember	CO 3	AIT004.08
8	Construct the Syntax tree for Expression using functions? (a + b) * (b - c)	Remember	CO 3	AIT004.08
9	Explain the functions to create nodes of Syntax tree for expression?	Understand	CO 3	AIT004.08
10	Define syntax tree? Draw the syntax tree for the assignment statement? . a :=b * -c + b * -c.	Remember	CO 3	AIT004.08
11	Define Translation schemes?	Understand	CO 3	AIT004.07
12	Define Annotated Parse Tree?	Remember	CO 3	AIT004.07
13	List the three kinds of intermediate representation?	Understand	CO 3	AIT004.09
14	State the benefits of using machine-independent intermediate form?	Understand	CO 3	AIT004.09
15	What is postfix notation?	Understand	CO 3	AIT004.09
16	How can you generate three-address code?	Remember	CO 3	AIT004.10
17	Translate x+y-(a*b)+c into three address code?	Remember	CO 3	AIT004.10
18	Discuss back-end and front-end?	Understand	CO 3	AIT004.10
19	List common methods for associating actual and formal parameters?	Understand	CO 3	AIT004.10
20	Define abstract or syntax tree?	Understand	CO 3	AIT004.11
21	List out types of three address code?	Understand	CO 3	AIT004.11
Part – B (Long Answer Questions)				
1	Explain briefly about syntax directed definition and it types?	Understand	CO 3	AIT004.08
2	Explain briefly about Synthesized and Inherited attribute in detail?	Understand	CO 3	AIT004.09
3	Define translation scheme and write three address code for a<b or b>c?	Remember	CO 3	AIT004.07
4	Explain briefly about S-attributed and L- attributed grammar in detail?	Remember	CO 3	AIT004.07
5	Explain how declaration is done in a procedure using syntax directed translation?	Understand	CO 3	AIT004.07
6	Explain briefly about postfix Translation Scheme?	Understand	CO 3	AIT004.08
7	Describe the method of generating syntax directed definition for control Statements?	Remember	CO 3	AIT004.08
8	Construct SDT for the simple assignment statement with example?	Understand	CO 3	AIT004.08
9	Explain the construction steps and construct the syntax tree for expression using functions? (m * n + p) + (m – n + p)?	Remember	CO 3	AIT004.08
10	Explain briefly syntax directed translation into three address code with suitable example?	Remember	CO 3	AIT004.08
11	Explain 3 addresscodes and mention its types. How would you implement the three address statements? Explain with suitable examples?	Remember	CO 3	AIT004.08
12	Explain with an example to generate the intermediate code for the flow of control statements?	Understand	CO 3	AIT004.09
13	Write about Quadruple and Triple with its structure?	Remember	CO 3	AIT004.09
14	Define and represent the Triple,indirect triple and qudraple for the assignment statement ? x:= -b + d * -b+d	Remember	CO 3	AIT004.09
15	Translate the arithmetic expression a* - (b+c) into	Remember	CO 3	AIT004.09

	a) A syntax tree b) Postfix notation c) Three-address code			
16	Translate the expression – (a + b) * (c + d) + (a + b +c) into a) quadruples b) triples C) indirect triples.	Remember	CO 3	AIT004.09
17	Explain translation scheme for Boolean Expressions with example?	Remember	CO 3	AIT004.11
18	Explain translation scheme for Control Flow with example?	Remember	CO 3	AIT004.11
	Part – C (Problem Solving and Critical Thinking)			
1	Write 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	Remember	CO 3	AIT004.11
2	Write production rules and semantic actions for the following grammar along with annotated parse tree for the string 9-5+4? expr→ expr + term | expr - term | term term→0|1|2|3|4|5|6|7|8|9	Remember	CO 3	AIT004.11
3	Write production rules and semantic actions for the following grammar along with annotated parse tree for the expression: “int a, b, c”? D →T L T →int T → float L→ $L_{1,}$id L → id	Remember	CO 3	AIT004.11
4	Write production rules and semantic actions for the following grammar along with annotated parse tree for the string (3+4)*(5+6)? L→E E→ T E→ E_1+T T→ F T→ T_1*F F→ (E) F→ digit	Remember	CO 3	AIT004.11
5	Write production rules and semantic actions for the following grammar along with annotated parse tree for the string a-4+c? E→E_1+T E→E_1-T E→T T→ (E) T→id T→ num	Remember	CO 3	AIT004.11

06	Generate 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)	Remember	CO 3	AIT004.09

07	Generate the three-address code for the following C program fragment
	?while(a > b)
	{
	if (c < d)
	x = y + z;
	else
	x = y - z;
	}
		Remember	CO 3	AIT004.09

08	Construct triples, Indirect and quadriples of an expression: a = b * - c + b * - c?	Remember	CO 3	AIT004.09
09	Construct triples, Indirect and quadriples of an expression : x = (a + b)* - c/d?	Remember	CO 3	AIT004.09
10	Why are quadruples preferred over triples in an optimizing compiler with example?	Remember	CO 3	AIT004.09
	UNIT -IV			
	TYPE CHECKING AND RUN TIME ENVIRONMENT			
	Part – A (Short Answer Questions)			
1	List different data structures used for symbol table?	Understand	CO 4	AIT004.14
2	Define Typechecking?	Understand	CO 4	AIT004.12
3	List the different types of type checking?	Understand	CO 4	AIT004.12
4	Define Type Expression?	Understand	CO 4	AIT004.12
5	Write about the type systems?	Understand	CO 4	AIT004.12
6	Write a short note on static type checking?	Understand	CO 4	AIT004.12
7	Write a short note on Dynamic type checking?	Understand	CO 4	AIT004.12
8	Define Structural Equivalence?	Understand	CO 4	AIT004.12
9	What is the Strongly typed language?	Understand	CO 4	AIT004.13
10	Define Type error?	Understand	CO 4	AIT004.13
11	Write Translation scheme for checking the type of Assignment statement S→id:=E	Remember	CO 4	AIT004.12
12	Write Translation scheme for checking the type of Conditional statement S→if E then S1	Remember	CO 4	AIT004.12
13	Write Translation scheme for checking the type of while statement S→While E do S1	Remember	CO 4	AIT004.12
14	Define Type conversion?	Understand	CO 4	AIT004.12
15	List the types of type conversion?	Understand	CO 4	AIT004.12
16	Write about general activation record?	Understand	CO 4	AIT004.14
17	Define Symbol table?	Understand	CO 4	AIT004.14
18	Define Dynamic storage allocation?	Understand	CO 4	AIT004.14
19	Write short note on procedures?	Understand	CO 4	AIT004.14
20	Define Activation tree?	Understand	CO 4	AIT004.14
21	Define stack storage allocation?	Understand	CO 4	AIT004.13
22	Define static storage allocation?	Understand	CO 4	AIT004.13
23	Define heap storage allocation?	Understand	CO 4	AIT004.13
24	Write a short note on parameter passing?	Understand	CO 4	AIT004.13
25	Define Control stack?	Understand	CO 4	AIT004.13
	Part – B (Long Answer Questions)			
1	Write a note on the specification of a simple type checker/	Understand	CO 4	AIT004.12
2	Define a type expression? Explain the equivalence of type expressions with an appropriate example?	Understand	CO 4	AIT004.12
3	Write about reusing the storage space for names?	Understand	CO 4	AIT004.14

4	Discuss and analyze about all allocation strategies in run-time storage environment?	Understand	CO 4	AIT004.14
5	Explain the data structures used for implementing Symbol Table?	Understand	CO 4	AIT004.15
6	Explain Static and Dynamic Checking of types with examples?	Understand	CO 4	AIT004.14
7	Differentiate the call by value and call by name with examples?	Understand	CO 4	AIT004.15
8	Distinguish between static and dynamic storage allocation?	Understand	CO 4	AIT004.14
9	Explain the type checking of expressions?	Understand	CO 4	AIT004.12
10	Write a short note on storage organization in runtime environment?	Understand	CO 4	AIT004.15
11	Explain the static and dynamic storage allocations?	Understand	CO 4	AIT004.13
12	Describe the name and structure equivalence in type expressions?	Understand	CO 4	AIT004.12
13	Explain the type checking of control flow statements?	Understand	CO 4	AIT004.12
14	Explain briefly about storage allocation strategies?	Understand	CO 4	AIT004.14
15	Describe the basic implementation techniques for symbol table?	Understand	CO 4	AIT004.15
16	Explain the calling sequences of activation record?	Remember	CO 4	AIT004.14
17	Differentiate ordered, unordered and binary search tree in symbol table?	Understand	CO 4	AIT004.15
18	Explain briefly about static storage allocation with block diagram?	Understand	CO 4	AIT004.14
19	Differentiate explicit and implicit allocation of memory to variables?	Understand	CO 4	AIT004.14
20	Differentiate stack and heap storage allocation strategies?	Understand	CO 4	AIT004.14
Part – C (Problem Solving and Critical Thinking)				
1	Suppose that the type of each identifier is a sub range of integers, for expressions with operators +, -, *, div and mod, as in Pascal. Write type-checking rules that assign to each sub expression the sub range its value must lie in?	Analysis	CO 4	AIT004.12
2	Explain briefly about Source language issues?	Understand	CO 4	AIT004.13
3	Explain briefly about Activation record with block diagram?	Understand	CO 4	AIT004.14
4	Discuss about varaiable length data on stack with neat diagram?	Understand	CO 4	AIT004.14
5	Explain briefly about heap storage allocation with block diagram?	Understand	CO 4	AIT004.14
6	Explain briefly about stack storage allocation with block diagram?	Understand	CO 4	AIT004.14
7	Explain briefly about language facilities for dynamic storage allocation?	Understand	CO 4	AIT004.14
8	Describe the parameter passing methods with examples?	Understand	CO 4	AIT004.14
9	Explain Over loading of Operators & Functions with examples?	Understand	CO 4	AIT004.14
10	Differentiate the call by reference and call by copy restore with examples?	Understand	CO 4	AIT004.14
UNIT-V				
CODE OPTIMIZATION AND CODE GENERATOR				
Part - A (Short Answer Questions)				
1	List the principle sources of optimization?	Understand	CO 5	AIT004.15
2	Define the 3 areas of code optimization?	Understand	CO 5	AIT004.15
3	Define local optimization?	Understand	CO 5	AIT004.15
4	Define constant folding?	Understand	CO 5	AIT004.15
5	Define Common Sub expressions?	Understand	CO 5	AIT004.15
6	Explain Dead Code?	Understand	CO 5	AIT004.15
7	Write the techniques used for loop optimization and Reduction in strength?	Remember	CO 5	AIT004.15
8	What is Register allocation and assignment?	Remember	CO 5	AIT004.13
9	Write about inner loops?	Remember	CO 5	AIT004.13
10	Define flow graph and basic block?	Understand	CO 5	AIT004.16
11	Define a DAG? Mention its Remember?	Understand	CO 5	AIT004.16
12	Define peephole optimization?	Remember	CO 5	AIT004.16
13	Write the machine instruction for operations and copy statement?	Remember	CO 5	AIT004.16
14	Analyze global data flow?	Understand	CO 5	AIT004.16
15	Write about live variable analysis?	Understand	CO 5	AIT004.15
16	Define the term copy propagation?	Understand	CO 5	AIT004.15
17	Define the term Code motion?	Understand	CO 5	AIT004.15
18	What is induction variable?	Understand	CO 5	AIT004.15
19	How do you calculate the cost of an instruction?	Understand	CO 5	AIT004.15

20	what is the Unreachable Code?	Understand	CO 5	AIT004.15
21	Generate the code for x: =x+1 for target machine?	Remember	CO 5	AIT004.17
22	Show the DAG for a: =b *-c + b * -c?	Remember	CO 5	AIT004.16
23	List the different types of loops in flowgraph?	Understand	CO 5	AIT004.16
24	Define Algebraic Simplification?	Understand	CO 5	AIT004.15
25	Define Dominators?	Understand	CO 5	AIT004.16
Part - B (Long Answer Questions)				
1	Explain the concept of Function-Preserving Transformations?	Remember	CO 5	AIT004.15
2	Explain Machine dependent code optimization in detail with an example?	Understand	CO 5	AIT004.15
3	Write about target code forms and explain how the instruction forms effect the computation time?	Understand	CO 5	AIT004.15
4	Write about machine dependent and machine independent optimization?	Understand	CO 5	AIT004.15
5	Explain the role of code generator in a compiler?	Understand	CO 5	AIT004.15
6	Write in detail the issues in the design of code generator?	Understand	CO 5	AIT004.17
7	Explain the instructions and address modes of the target machine?	Understand	CO 5	AIT004.12
8	Explain the principle sources of code optimization in detail?	Understand	CO 5	AIT004.15
9	Define the primary structure preserving transformations on basic blocks?	Understand	CO 5	AIT004.17
10	Explain peephole optimization in detail?	Understand	CO 5	AIT004.17
11	Discuss about the following i. Copy propagation ii. Dead code elimination iii. Code motion	Remember	CO 5	AIT004.16
12	Explain in the DAG representation of the basic block with example?	Remember	CO 5	AIT004.16
13	Explain loop optimization in detail with example?	Remember	CO 5	AIT004.15
14	Explain various Global optimization techniques in detail?	Remember	CO 5	AIT004.16
15	Explain Loops in flowgraph in detail with example?	Remember	CO 5	AIT004.17
16	Explain Local optimization in detail with example?	Remember	CO 5	AIT004.16
17	Discuss Redundant-instructions elimination and Flow-of-control optimizations?	Understand	CO 5	AIT004.17
18	Demonstrate the simple code generator with a suitable example?	Remember	CO 5	AIT004.17
19	Write the procedure to detect induction variable and dead code elimination with example?	Remember	CO 5	AIT004.20
20	Explain briefly about register allocation and assignment?	Understand	CO 5	AIT004.16
21	Explain the instruction cost in detail with example?	Understand	CO 5	AIT004.16
Part – C (Problem Solving and Critical Thinking)				
1	Show the code sequence generated by the simple code generation algorithm x*y+(m-k)-(g+b)	Remember	CO 5	AIT004.17

2	Generate target code for the given program segments:
	main()
	{
	int i=4,j;
	j = i + 5;
	}
		Remember	CO 5	AIT004.17

3	Consider the following basic block of 3-address instructions .Generate 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
		Remember	CO 5	AIT004.16

4	Identify the register descriptor target code for the source language Statement and its cost. (a-b) + (a-c) + (a-c)	Remember	CO 5	AIT004.17

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
		Remember	CO 5	AIT004.16

6	Construct the DAG for the following basic block.
	D:=B*C
	E:=A+B
	B:=B+C
	A:=E-D
		Remember	CO 5	AIT004.16

7	Design basic block for following code
	void quicksort(m, n)
	int m, n;
	{
	int i, j;
	if (n <= m)
	return; /* fragment begins here */
	i = m-1;
	j = n;
	v = a[n];
	while(1)
	{
	do
	i = i+1;
	while(a[i] < v);
	do
	j = j-1;
	while(a[j] > v);
	if(i >= 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);
	}.
		Remember	CO 5	AIT004.17

8	Explain how the following expression can be converting in a DAG. a+b*(a+b)+c+d	Remember	CO 5	AIT004.16

9	Explain role of DAG representation in optimization with example?	Remember	CO 5	AIT004.16

10	Deign the basicblock and flowgraph 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
		Remember	CO 5	AIT004.20

11	Generate optimal machine code for the following c program.
	main()
	{
	int i,a[10];
	while(i<=10)
	a[i]=0;
	}
		Remember		AIT004.18

Prepared by:
Ms. E Uma Shankari, Assistant Professor

HOD, IT