Hall Ticket No | | | | | | | | | | | Question Paper Code: AIT004

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)

MODEL QUESTION PAPER II

B.Tech V Semester End Examinations (Regular), November – 2019

Regulation: IARE–R16

COMPILER DESIGN
(Common to CSE / IT)

Time: 3 Hours **Max Marks: 70**

Answer ONE Question from each Unit
All Questions Carry Equal Marks
All parts of the question must be answered in one place only

UNIT – I

1. a) What is Complier? With a neat block diagram, explain the various phases of a compiler. **[7M]**
   b) Construct deterministic finite automata in which all the strings containing total number of a mod 3 = 0 and total number of b mod 3 = 1. **[7M]**

2. a) What are the necessary conditions to be carried out before construction of predictive parsing? **[7M]**
   b) Construct the predictive parsing table for the following grammar **[7M]**
   S –> (L) | a
   L –> L, S | S
   Also show the behavior of the parser on the sentence (a, a).

UNIT – II

3. a) Define the terms reduction, handle and right sential form. Explain with an example, the importance of picking the right-handles during a reduction sequence. **[7M]**
   b) Consider the following grammar **[7M]**
   S –> TL;
   T –> int | float
   L –> L, id | id
   Parse the input string"float id, id;" using shift reduce parser.

4. a) What are the different methods of constructing the LR parsing table from a grammar? What are the common steps involved in constructing the LR parsing table from a grammar. **[7M]**
   b) Construct the LALR(1) parsing table for the following grammar **[7M]**
   S –> CC
   C –> aC
   C –> d
   Parse the input string "aadd" using the LALR(1) parsing table.

UNIT – III

5. a) What is a translation scheme? How is it different from a syntax directed definition? Illustrate the order of execution of semantic actions in a translation scheme? **[7M]**
   b) Define syntax tree. Construct the syntax tree for the conditional statement if a == b then a = c + d else b = c - d. **[7M]**

6. a) A Compiler can choose one of the two options **[7M]**
   i) Translate the input source into intermediate code and then convert it to final machine code;
   ii) Directly generate the final machine code from the input source.
   What is the preferred option and why?

b) Write quadruples, triples and indirect triples for the expression: **[7M]**
(a + b) * (c + d) - (a + b + c).

UNIT – IV

7. a) Explain static and dynamic type checking with examples. **[7M]**
b) Draw and explain the runtime memory organization static storage allocation strategy with pros and cons. **[7M]**

8. a) What is run-time environment? What are the important elements of runtime environment? How is it controlled in a program that is compiled? **[7M]**
b) Define symbol table. Explain different data structures that are used in symbol table organization. **[7M]**

UNIT – V

9. a) What is the importance of loop optimization? Discuss peephole optimization in detail along with suitable examples. **[7M]**
b) Explain about the sources and criterions of code optimization as machine dependent and independent types. **[7M]**

10. a) What is a leader of basic block? Write and explain the algorithm used to find leaders. **[7M]**
b) Construct the DAG for the following basic block **[7M]**
d = b * c
e = a + b
b = b * c
a = e - d
Then generate the code for above constructed DAG using only one register.

– ◦ ◦ ◦ ◦ –

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)

COURSE OBJECTIVES:

The course should enable the students to:

I.	Apply the principles in the 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:

CO 1	Understand the various phases of compiler and design the lexical analyzer.Demonstrate the phases of the compilation process and able to describe the purpose and operation of each phase.
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:

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

Mapping of Semester End Examinations to Course Learning Outcomes:

SEE Question No		Course Learning Outcomes		Course Outcomes	Blooms Taxonomy Level
1	a	AIT004.01	Define the phases of a typical compiler	CO 1	Understand
	b	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.	CO 1	Remember
2	a	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 1	Remember
	b	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 1	Remember
3	a	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 2	Remember
	b	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 2	Remember
4	a	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 2	Remember
	b	AIT004.05	Apply an algorithm for a top-down or a bottom-up parser construction; Construct a parser for a given context-free grammar.	CO 2	Remember
5	a	AIT004.07	Understand syntax directed translation schemes for a context free grammar.	CO 3	Understand
	b	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.	CO 3	Understand
6	a	AIT004.09	Understand the need of intermediate code generation phase in compilers.	CO 3	Understand
	b	AIT004.10	Write intermediate code for statements like assignment, conditional, loops and functions in high level language.	CO 3	Understand
7	a	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.	CO 4	Understand
	b	AIT004.13	Differentiate static vs. dynamic storage allocation and the usage of activation records to manage program modules and their data.	CO 4	Remember
8	a	AIT004.12	Explain the role of different types of runtime environments and memory organization for implementation of typical programming languages.	CO 4	Understand

	b	AIT004.14	Understand the role of symbol table data structure in the construction of compiler.	CO 4	Understand
9	a	AIT004.15	Learn the code optimization techniques to improve the performance of a program in terms of speed and space.	CO 5	Understand
	b	AIT004.16	Implement the global optimization using data flow analysis such as basic blocks and DAG.	CO 5	Apply
10	a	AIT004.17	Understand the code generation techniques to generate target code.	CO 5	Understand
	b	AIT004.16	Implement the global optimization using data flow analysis such as basic blocks and DAG.	CO 5	Apply

Signature of Course Coordinator **HOD, CSE**