|  |  |
| --- | --- |
| **Course – 65 Title: Compiler Design and Automata Theory** |  |
| **Course No.: CIT-411 Credit : 3.00 Contact Hours: 3** | **Total Marks: 100** |

**11.1 Rationale**

Computer Engineers should be competent in compiler design and automata theory. They must learn the fundamental concepts of compiler design and automata theory and also various phases in the design of a compiler, how to generate a machine code from a C program statement.

**11.2 Objectives**

- To design a Lex compiler

- to construct a DFA from the NFA

- to design an NFA for the corresponding regular expressions.

|  |  |  |  |
| --- | --- | --- | --- |
| **11.3 Learning Outcomes** | **11.4 Course Content** | **11.5 Teaching  Learning Strategy** | **11.6 Assessment Strategy** |
| * Define finite automata * Describe pushdown automata * Design Turing machines | Automata: Finite automata and regular languages, pushdown automata and context-free languages; Turing machines and recursively enumerable sets; linear-bounded automata and context sensitive languages; computability and the halting problem | * Lecture * Group Assignment | * Assignment * Questionnaire |
| * Construct NFA from Regular Expression * Design DFA from NFA * Design of a lexical analyzer generator using LEX | Lexical analysis: Role, from regular expression to NFA, from NFA to DFA, design of a lexical analyzer generator using LEX. | * Lecture * Case Studies * Assignment | * Exercise * Assignment |
| * Classify different types of parsing * Design different types of grammar | Syntax analysis: role, CFG, writing a grammar, top-down parsing, bottom-up parsing, operator precedence parsing, LR parser, using ambiguous grammar, parser generators (YACC).  Symbol table, structure and management. | * Demonstration * Group Discussion | * Observation * Self-rating |
| * Explain different types of intermediate code generation | Intermediate code generation: Intermediate languages, declarations, assignment statement, Boolean expression, case statements, back patching, procedure calls. | * Lecture * Reading Assignment | * True or False * Multiple Choice |
| * Discuss the several issues in the design of a code generator * Draw a basic block & corresponding flow graphs for the given three-address statements | Code generation: issues in the design of a code generator, target machine, runtime storage management, basic blocks and flow graphs, register allocation and assignment, dag representation of basic blocks, peephole optimizations, generating code from dags. | * Lecture * Group Assignment | * Questionnaire * Completion * Assignment |
| * Role of the code optimization in compiler design | Code optimization: principle of source optimization, optimization of basic blocks, loop in flow graphs, global data flow analysis, iterative solution of data flow equations. | * Lecture * Group Discussion | * Short Answer * Group Exercise * Viva Voce |

**RECOMMENDED BOOKS AND PERIODICALS**

**Text Books**:

1. Alfred V.Aho, Ravi Sethi, Jeffrey D.Ullman : Compilers Principles, Techniques and tools. Third edition.
2. Hopcroft and Ullman : Introduction to Automata Theory, Languages and Computation

**References:**

1. Compiler Design Theory - by Lewis and Stern