Automata Theory and Compiler Design

I. Automata Theory

Introduction: **3**
Definition of a Finite Automaton, Non-deterministic Finite state Automaton (NDFA), Deterministic Finite state Automaton (DFA), NDFA to DFA conversion, Minimized Equivalent Machine, State Minimization Algorithm – Row elimination method, Implication Table Method, Basics of regular expression.

Formal languages and grammar: **3**
Introduction to Formal Grammar and Language, Chomsky's Classification of Grammar – Type 0, Type-1 or Context Sensitive, Type-2 or Context Free and Type-3 or Regular Grammar, CNF, GNF. Illustration of each of these classes with example, Derivation tree, Parse Tree, Syntax Tree, Ambiguous and Unambiguous Grammar.
Regular expression to Finite Automata conversion, FA to Regular Grammar and Regular Grammar to FA conversion

Push-down automata (PDA): **3**
Definition, PDA and CFL: design and conversion, Acceptance of Strings.

Turing Machine: **4**
Introduction, Turing Machine Model, Computable languages and function

II. Compiler Design

Introduction to Compiling **2**
Introduction, Analysis-synthesis model, Phases of the compiler.

Lexical analysis **2**
Role of lexical analyser, Tokens, Patterns, Lexemes, Input buffering, Specifications of a token, Recognition of tokens, Design of a lexical analyser generator (Lex).

Syntax analysis **9**
The role of a parser, Context free grammars, Writing a grammar, Top down Parsing, Non-recursive Predictive parsing (LL), Bottom up parsing, Handles, Viable prefixes, Operator precedence parsing, LR parsers (SLR, LALR), Parser generators (YACC). Error Recovery strategies for different parsing techniques, Syntax directed translation, Syntax directed definitions, Construction of syntax trees, Bottom-up evaluation of S-attributed definitions, L-attributed definitions, and Bottom-up evaluation of inherited attributes.

Run time environment: **2**
Parameter passing, symbol table, dynamic storage allocation techniques

Intermediate code generation: **4**
Intermediate languages, Graphical representation, Three-address code, Implementation of three address statements (Quadruples, Triples, Indirect triples).

Code generation and optimization: **8**
Issues in the design of code generator, a simple code generator, Register allocation and assignment, Introduction to code optimization, Basic blocks & flow graphs, Transformation of basic blocks, DAG representation of basic blocks, the principle sources of optimization, Loops in flow graph, Peephole optimization.