

**UNIT I****Chapter 1 : Introduction to Systems Programming and Assemblers 1-1 to 1-44****Syllabus :**

- Introduction : Need of System Software, Components of System Software, Language Processing Activities, Fundamentals of Language Processing.
- Assemblers : Elements of Assembly Language Programming, A simple Assembly Scheme, Pass structure of Assemblers, Design of Two Pass Assembler, Single pass assembler.

|       |                                                        |      |
|-------|--------------------------------------------------------|------|
| 1.1   | Introduction .....                                     | 1-1  |
| 1.1.1 | Different Types of Application Software.....           | 1-2  |
| 1.1.2 | Need of System Software.....                           | 1-2  |
| 1.2   | Components of System Software .....                    | 1-3  |
| 1.2.1 | Operating System.....                                  | 1-3  |
| 1.2.2 | Language Translators.....                              | 1-3  |
| 1.2.3 | Loader .....                                           | 1-4  |
| 1.2.4 | Linker .....                                           | 1-5  |
| 1.2.5 | Macro Processor .....                                  | 1-5  |
| 1.3   | Language Processor .....                               | 1-5  |
| 1.3.1 | Language Processing Activities.....                    | 1-5  |
| 1.3.2 | Problem Oriented and Procedure-Oriented Languages..... | 1-6  |
| 1.4   | Fundamental of Language Processing .....               | 1-7  |
| 1.4.1 | Analysis Phase.....                                    | 1-8  |
| 1.4.2 | Synthesis Phase.....                                   | 1-9  |
| 1.4.3 | Forward Reference Problem.....                         | 1-10 |
| 1.4.4 | Pass Structure.....                                    | 1-10 |
| 1.4.5 | Intermediate Representation (IR).....                  | 1-11 |
| 1.4.6 | Toy Compiler.....                                      | 1-11 |
| 1.5   | Assemblers .....                                       | 1-13 |
| 1.5.1 | Elements of Assembly Language Programming .....        | 1-14 |
| 1.5.2 | Syntax of Assembly Language Statement .....            | 1-15 |
| 1.6   | A Simple Assembly Language.....                        | 1-16 |
| 1.7   | A Simple Assembly Scheme .....                         | 1-21 |
| 1.8   | Pass Structure of Assembler .....                      | 1-22 |
| 1.8.1 | Single Pass Assembler .....                            | 1-22 |

|          |                                                                           |      |
|----------|---------------------------------------------------------------------------|------|
| 1.8.2    | Design of Two Pass Assembler.....                                         | 1-26 |
| 1.8.2(A) | PASS I .....                                                              | 1-26 |
| 1.8.2(B) | Data Structure of Pass I .....                                            | 1-27 |
| 1.8.2(C) | Pass I Assembler Algorithm .....                                          | 1-33 |
| 1.8.3    | Intermediate Code Forms (Representations) for an Assembler .....          | 1-34 |
| 1.8.3(A) | Intermediate Code for Imperative Statement .....                          | 1-34 |
| 1.8.3(B) | Intermediate Code for Declaration Statement and Assembler Directive ..... | 1-35 |
| 1.8.4    | Comparison between Variant I and Variant II .....                         | 1-38 |
| 1.8.5    | PASS II .....                                                             | 1-38 |
| 1.8.5(A) | Assembler Pass II Algorithm .....                                         | 1-39 |
| 1.8.6    | Difference between Single Pass and Two Pass Assembler .....               | 1-41 |
| 1.9      | Listing and Error Reporting.....                                          | 1-41 |
| 1.10     | University Questions and Answers .....                                    | 1-42 |

**UNIT II****Chapter 2 : Macro Processors, Loaders and Linkers 2-1 to 2-48****Syllabus :**

- Macro Processor: Macro Definition and call, Macro Expansion, Nested Macro Calls and definition, Advanced Macro Facilities, Design of two-pass Macro Processor.
- Loaders : Loader Schemes, Compile and Go, General Loader Scheme, Absolute Loader Scheme, Subroutine Linkages, Relocation and linking concepts, Self-relocating programs, Relocating Loaders, Direct Linking Loaders, Overlay Structure.

|          |                                        |      |
|----------|----------------------------------------|------|
| 2.1      | Introduction .....                     | 2-1  |
| 2.1.1    | Macro .....                            | 2-1  |
| 2.1.2    | Macro Processor .....                  | 2-2  |
| 2.1.3    | Macro Definition .....                 | 2-3  |
| 2.1.4    | Macro Call .....                       | 2-4  |
| 2.1.5    | Macro Expansion.....                   | 2-5  |
| 2.1.5(A) | Procedure of Macro Expansion.....      | 2-5  |
| 2.1.5(B) | Classification of Macro Expansion..... | 2-6  |
| 2.1.5(C) | Algorithm (Macro Expansion) .....      | 2-7  |
| 2.1.6    | Lexical Substitution .....             | 2-7  |
| 2.1.7    | Nested Macro Calls .....               | 2-10 |
| 2.1.8    | Advanced Macro Facilities.....         | 2-11 |



|          |                                                                    |      |        |                                                               |      |
|----------|--------------------------------------------------------------------|------|--------|---------------------------------------------------------------|------|
| 2.1.9    | Conditional Expansion.....                                         | 2-13 | 3.3.3  | Compilation Process with Example .....                        | 3-6  |
| 2.2      | Design of a Macro Preprocessor .....                               | 2-15 | 3.4    | Front End and Back End of the Compiler .....                  | 3-12 |
| 2.2.1    | Design Architecture of Macro Pre-processor .....                   | 2-16 | 3.5    | Introduction of the Lexical Analysis .....                    | 3-13 |
| 2.2.2    | Data Structures for Macro Preprocessors.....                       | 2-17 | 3.5.1  | Review of the Lexical Analysis .....                          | 3-14 |
| 2.2.2(A) | Processing of Macro Definitions .....                              | 2-20 | 3.5.2  | Input to Lexical Analyzer .....                               | 3-15 |
| 2.2.2(B) | Macro Expansion.....                                               | 2-21 | 3.5.3  | How it works ? .....                                          | 3-15 |
| 2.2.2(C) | Nested Macro Calls Expansion.....                                  | 2-22 | 3.6    | Lexical Errors and Error Handling.....                        | 3-16 |
| 2.2.3    | Macro Assembler .....                                              | 2-23 | 3.7    | Input Buffering.....                                          | 3-17 |
| 2.2.3(A) | Difference between Macro Pre-processor<br>and Macro Assembler..... | 2-23 | 3.8    | Specification of Tokens .....                                 | 3-17 |
| 2.2.4    | Design of Two Pass Macro Processor.....                            | 2-24 | 3.8.1  | Construction of Lexical Analyzer.....                         | 3-18 |
| 2.3      | Loaders .....                                                      | 2-31 | 3.8.2  | Transition Diagram .....                                      | 3-19 |
| 2.3.1    | Loader Schemes .....                                               | 2-33 | 3.8.3  | Block Semantic .....                                          | 3-20 |
| 2.4      | Relocation and Linking Concepts .....                              | 2-39 | 3.9    | Recognition of Tokens.....                                    | 3-22 |
| 2.4.1    | Performing Relocation.....                                         | 2-41 | 3.10   | Design of Lexical Analyzer using<br>Uniform Symbol Table..... | 3-23 |
| 2.4.2    | Linking.....                                                       | 2-42 | 3.11   | LEX (Automatic Generation of Lexical Analyzer ) .....         | 3-29 |
| 2.4.3    | Public Definition and External Reference.....                      | 2-42 | 3.11.1 | LEX Specifications .....                                      | 3-30 |
| 2.4.4    | Resolving External References.....                                 | 2-43 | 3.11.2 | LEX Pattern.....                                              | 3-30 |
| 2.4.5    | Self-Relocating Loaders .....                                      | 2-43 | 3.11.3 | LEX Actions.....                                              | 3-31 |
| 2.4.6    | Dynamic Linking .....                                              | 2-44 | 3.11.4 | Programs using LEX .....                                      | 3-31 |
| 2.5      | Overlay Structure .....                                            | 2-44 | 3.12   | Role of the Finite Automata in the Compiler .....             | 3-34 |
| 2.5.1    | Difference Between Linker and Loader.....                          | 2-46 | 3.13   | University Questions and Answers .....                        | 3-34 |
| 2.6      | University Questions and Answers .....                             | 2-46 |        |                                                               |      |

**UNIT III****Chapter 3 : Introduction to Compilers 3-1 to 3-35****Syllabus :**

- Phase structure of Compiler and entire compilation process.
- Lexical Analyzer : The Role of the Lexical Analyzer, Input Buffering. Specification of Tokens, Recognition of Tokens, Design of Lexical Analyzer using Uniform Symbol Table, Lexical Errors.
- LEX : LEX Specification, Generation of Lexical Analyzer by LEX.

|       |                                        |     |
|-------|----------------------------------------|-----|
| 3.1   | Introduction to Compilers.....         | 3-1 |
| 3.2   | Design Issues.....                     | 3-2 |
| 3.3   | Phases of Compilation .....            | 3-4 |
| 3.3.1 | The Analysis of a Source Program ..... | 3-5 |
| 3.3.2 | Synthesis of a Source Program .....    | 3-6 |

**UNIT IV****Chapter 4 : Parsers****4-1 to 4-66****Syllabus :**

- Role of parsers, Classification of Parsers: Top down parsers-recursive descent parser and predictive parser.
- Bottom up Parsers - Shift Reduce: SLR, CLR and LALR parsers. Error Detection and Recovery in Parser. YACC specification and Automatic construction of Parser (YACC).

|       |                                           |     |
|-------|-------------------------------------------|-----|
| 4.1   | Introduction to Syntax Analysis.....      | 4-1 |
| 4.2   | Role of the Parser (Syntax Analyzer)..... | 4-2 |
| 4.2.1 | Generate Parse Tree.....                  | 4-2 |
| 4.2.2 | Error Handling .....                      | 4-2 |
| 4.3   | Specification of Grammar.....             | 4-3 |
| 4.3.1 | Types of Grammars.....                    | 4-3 |
| 4.3.2 | Context Free Grammar.....                 | 4-3 |



|        |                                                                            |      |
|--------|----------------------------------------------------------------------------|------|
| 4.3.3  | Derivation .....                                                           | 4-4  |
| 4.3.4  | Parse Trees and Derivations .....                                          | 4-5  |
| 4.3.5  | Ambiguity .....                                                            | 4-5  |
| 4.4    | Modifying Grammars .....                                                   | 4-6  |
| 4.4.1  | Eliminating Left Recursion .....                                           | 4-6  |
| 4.4.2  | Left Factoring .....                                                       | 4-7  |
| 4.5    | Classification of Parsers .....                                            | 4-8  |
| 4.6    | Top-Down Parser .....                                                      | 4-8  |
| 4.6.1  | Recursive Descent Parser (RDP) .....                                       | 4-10 |
| 4.6.2  | Predictive Parser .....                                                    | 4-13 |
| 4.6.3  | Transition Diagrams for Predictive Parser .....                            | 4-14 |
| 4.6.4  | Difference between Recursive Descent<br>Parser and Predictive Parser ..... | 4-14 |
| 4.6.5  | Non Recursive Predictive Parsing .....                                     | 4-16 |
| 4.6.6  | Predictive Parser Model (PP1) .....                                        | 4-16 |
| 4.6.7  | Construction of the Table .....                                            | 4-17 |
| 4.7    | LL(1) Grammars .....                                                       | 4-19 |
| 4.7.1  | Solved Examples of LL(1) .....                                             | 4-20 |
| 4.8    | Bottom-Up Parser .....                                                     | 4-25 |
| 4.8.1  | Difference between Top-Down<br>and Bottom-Up Parser .....                  | 4-25 |
| 4.9    | Shift Reduce Parser .....                                                  | 4-26 |
| 4.9.1  | LR Parser .....                                                            | 4-27 |
| 4.9.2  | SLR Parsing Table .....                                                    | 4-28 |
| 4.9.3  | The Closure Operation .....                                                | 4-29 |
| 4.9.4  | Algorithm for SLR Parsing Table .....                                      | 4-29 |
| 4.10   | Solved Examples of SLR .....                                               | 4-30 |
| 4.11   | Constructing Canonical LR Parsing Table .....                              | 4-41 |
| 4.11.1 | To Construct LR (31) Items Function Closure (I) .....                      | 4-42 |
| 4.11.2 | Construction of Canonical LR Parsing Table .....                           | 4-42 |
| 4.12   | Solved Examples of LR .....                                                | 4-42 |
| 4.13   | Constructing LALR Parsing Table .....                                      | 4-49 |
| 4.14   | Using Ambiguous Grammar .....                                              | 4-53 |
| 4.15   | Comparison of the LR Parsers .....                                         | 4-56 |
| 4.15.1 | Difference between LALR and CLR Parsers .....                              | 4-56 |
| 4.15.2 | Difference Between SLR, LALR and CLR Parsers .....                         | 4-57 |
| 4.16   | Error Detection and Recovery in Parser .....                               | 4-57 |
| 4.16.1 | Recovery from Syntax Errors .....                                          | 4-58 |
| 4.16.2 | Error Recovery in Top-Down Parsing .....                                   | 4-59 |
| 4.16.3 | Recovery in Bottom-Up Parsing .....                                        | 4-59 |

|        |                                                                         |      |
|--------|-------------------------------------------------------------------------|------|
| 4.17   | YACC Specification and Automatic<br>Construction of Parser (YACC) ..... | 4-59 |
| 4.17.1 | YACC Specification .....                                                | 4-59 |
| 4.17.2 | Automatic Generation of the LR Parser .....                             | 4-60 |
| 4.17.3 | Programs using Lex and YACC .....                                       | 4-61 |
| 4.18   | University Questions and Answers .....                                  | 4-65 |

## UNIT V

### Chapter 5 : Semantic Analysis and Storage Allocation

**5-1 to 5-48**

#### Syllabus :

- Need, Syntax Directed Translation, Syntax Directed Definitions, Translation of assignment Statements, iterative statements, Boolean expressions, conditional statements, Type Checking and Type conversion.
- Intermediate Code Formats : Postfix notation, Parse and syntax trees, Three address code, quadruples and triples. Storage Allocation : Storage organization and allocation strategies.

|       |                                                         |      |
|-------|---------------------------------------------------------|------|
| 5.1   | Introduction to Semantic Analysis .....                 | 5-1  |
| 5.1.1 | Need of Semantic Analysis .....                         | 5-1  |
| 5.2   | Introduction to Syntax Directed Translation .....       | 5-2  |
| 5.2.1 | Syntax-Directed Definitions and Attribute Grammar ..... | 5-2  |
| 5.2.2 | Form of a Syntax- Directed Definition .....             | 5-3  |
| 5.3   | Synthesized Attributes .....                            | 5-4  |
| 5.4   | Inherited Attributes .....                              | 5-7  |
| 5.4.1 | Dependency Graphs .....                                 | 5-8  |
| 5.4.2 | Evaluation Order .....                                  | 5-9  |
| 5.5   | Translation of Assignment Statements .....              | 5-10 |
| 5.5.1 | Type Conversion within Assignment .....                 | 5-11 |
| 5.6   | Flow Control Statements .....                           | 5-12 |
| 5.6.1 | Iteration Statements .....                              | 5-12 |
| 5.6.2 | Conditional Statements .....                            | 5-12 |
| 5.7   | Boolean Expressions .....                               | 5-13 |
| 5.7.1 | Methods of Translating Boolean Expressions .....        | 5-13 |
| 5.7.2 | Numerical Representation of Boolean Expressions .....   | 5-14 |
| 5.7.3 | Short-Circuit Code for Boolean Expressions .....        | 5-15 |
| 5.8   | Type Checking .....                                     | 5-15 |
| 5.8.1 | Type Systems .....                                      | 5-15 |
| 5.8.2 | Type Expressions .....                                  | 5-16 |
| 5.8.3 | Writing a Simple Type Checker .....                     | 5-16 |
| 5.8.4 | Type Checking of Expression .....                       | 5-17 |



|          |                                                  |      |        |                                                                           |      |
|----------|--------------------------------------------------|------|--------|---------------------------------------------------------------------------|------|
| 5.8.5    | Type Checking of Statements .....                | 5-17 | 6.1    | Introduction to Code Generation.....                                      | 6-1  |
| 5.8.6    | Type Conversions.....                            | 5-18 | 6.1.1  | Issues in Code Generation Phase.....                                      | 6-2  |
| 5.8.6(A) | Coercions .....                                  | 5-18 | 6.2    | Basic Blocks and Flow Graph.....                                          | 6-6  |
| 5.9      | Introduction to Intermediate Code .....          | 5-18 | 6.2.1  | Introduction .....                                                        | 6-6  |
| 5.9.1    | Intermediate Languages.....                      | 5-19 | 6.2.2  | Basic Block and Algorithm to Find Basic Block .....                       | 6-7  |
| 5.10     | Intermediate Code Formats.....                   | 5-19 | 6.2.3  | Transformations on Basic Blocks .....                                     | 6-9  |
| 5.10.1   | Syntax Tree and Parse Tree.....                  | 5-19 | 6.2.4  | Flow Graph.....                                                           | 6-11 |
| 5.10.2   | Postfix Expressions / Notations .....            | 5-20 | 6.3    | DAG Representation of Basic Blocks .....                                  | 6-13 |
| 5.10.3   | Construction of Syntax Trees .....               | 5-20 | 6.4    | A Simple Code Generator .....                                             | 6-13 |
| 5.10.4   | Directed Acyclic Graph for Expressions.....      | 5-23 | 6.4.1  | Code Generation Algorithm .....                                           | 6-14 |
| 5.10.5   | Three Address Code .....                         | 5-24 | 6.5    | Introduction and Need of Code Optimization .....                          | 6-19 |
| 5.10.6   | Types of Three-Address Statements .....          | 5-24 | 6.6    | Classification of Optimization .....                                      | 6-20 |
| 5.10.7   | Implementations of Three-Address Statements..... | 5-25 | 6.6.1  | Difference between Machine Dependent<br>and Independent Optimization..... | 6-21 |
| 5.11     | Storage Allocation .....                         | 5-35 | 6.7    | Principal Sources of Code Optimization .....                              | 6-21 |
| 5.12     | Source Language Issues.....                      | 5-35 | 6.8    | Machine Independent Techniques.....                                       | 6-22 |
| 5.12.1   | Procedures .....                                 | 5-36 | 6.8.1  | Peephole Optimisation .....                                               | 6-22 |
| 5.12.2   | Activation Trees.....                            | 5-36 | 6.8.2  | Elimination of Common Sub-expression.....                                 | 6-23 |
| 5.12.3   | The Scope of a Declaration .....                 | 5-38 | 6.8.3  | Elimination of Dead Code.....                                             | 6-26 |
| 5.12.4   | Binding of Names .....                           | 5-38 | 6.8.4  | Removing of Loop Invariants : Loop Optimization.....                      | 6-26 |
| 5.13     | Storage Organization.....                        | 5-39 | 6.8.5  | Dynamic Programming Code Generation .....                                 | 6-28 |
| 5.13.1   | Compile-Time Layout of Local Data.....           | 5-40 | 6.9    | Local Optimization.....                                                   | 6-29 |
| 5.14     | Activation Records .....                         | 5-40 | 6.10   | Global Optimization .....                                                 | 6-30 |
| 5.15     | Storage Allocation Strategies .....              | 5-43 | 6.11   | Machine Dependent Issues .....                                            | 6-31 |
| 5.15.1   | Static Allocation.....                           | 5-43 | 6.11.1 | Assignment and Use of Registers .....                                     | 6-31 |
| 5.15.2   | Stack Allocation.....                            | 5-44 | 6.11.2 | Instruction Cost .....                                                    | 6-32 |
| 5.15.3   | Heap Allocation .....                            | 5-45 | 6.11.3 | Rearrangement of Quadruples for<br>Code Optimization.....                 | 6-32 |
| 5.16     | University Questions and Answers .....           | 5-46 | 6.12   | University Questions and Answers .....                                    | 6-33 |

## UNIT VI

### Chapter 6 : Code Generation and Optimization

6-1 to 6-34

#### Syllabus :

- Code Generation : Code generation Issues. Basic blocks and flow graphs, A Simple Code Generator.
- Code Optimization : Machine Independent: Peephole optimizations: Common Sub-expression elimination, Removing of loop invariants, Induction variables and Reduction in strengths, use of machine idioms, Dynamic Programming Code Generation.
- Machine dependent Issues : Assignment and use of registers, Rearrangement of Quadruples for code, optimization.