

**Chapter 1 : Overview of the Compiler and its Structure**

**1-1 to 1-15**

**Syllabus** : Language processor, Applications of language processors, Definition-Structure-Working of compiler, The science of building compilers, Basic understanding of interpreter and assembler. Difference between interpreter and compiler. Compilation of source code into target language, Cousins of compiler, Types of compiler.

<b>1.1</b>	<b>Language Processor .....</b>	<b>1-1</b>
1.1.1	Applications of Language Processor .....	1-1
<b>1.2</b>	<b>Definition-Structure-Working of Compiler .....</b>	<b>1-2</b>
1.2.1	The Science of Building Compilers.....	1-3
<b>1.3</b>	<b>Basic Understanding of Interpreter and Assembler.....</b>	<b>1-3</b>
1.3.1	Difference between Interpreter and Compiler .....	1-4
<b>1.4</b>	<b>Compilation of Source Code into Target Language .....</b>	<b>1-5</b>
1.4.1	The Analysis of a Source Program .....	1-5
<b>1.5</b>	<b>Phases of Compiler with Example.....</b>	<b>1-7</b>
<b>1.6</b>	<b>Cousins of compiler.....</b>	<b>1-11</b>
<b>1.7</b>	<b>Types of Compiler.....</b>	<b>1-14</b>

**Chapter 2 : Lexical Analysis**

**2-1 to 2-23**

**Syllabus** : The Role of the Lexical Analyzer, Specification of Tokens, Recognition of Tokens, Input Buffering, elementary scanner design and its implementation (Lex), Applying concepts of Finite Automata for recognition of tokens.

<b>2.1</b>	<b>Introduction of the Lexical Analysis.....</b>	<b>2-1</b>
<b>2.2</b>	<b>Specification of Token .....</b>	<b>2-2</b>
<b>2.3</b>	<b>Recognition of Token.....</b>	<b>2-3</b>
2.3.1	Input to Lexical Analyzer .....	2-4
2.3.2	Transition Diagram.....	2-4
2.3.3	Hand Coded Lexical Analyzer.....	2-6
<b>2.4</b>	<b>Input Buffering.....</b>	<b>2-7</b>
2.4.1	Elementary Scanner design and its Implementation (Lex).....	2-8
2.4.2	Non-deterministic Finite Automata .....	2-12
2.4.3	DFA (Deterministic Finite Automata) .....	2-12
2.4.4	Construction of an NFA from Regular Expression .....	2-13
2.4.5	Conversion of NFA to DFA .....	2-14

**Chapter 3 : Syntax Analysis**

**3-1 to 3-62**

**Syllabus :** Understanding Parser and CFG(Context Free Grammars), Left Recursion and Left Factoring of grammar Top Down and Bottom up Parsing Algorithms, Operator-Precedence Parsing, LR Parsers, Using Ambiguous Grammars, Parser Generators, Automatic Generation of Parsers. Syntax-Directed Definitions, Construction of Syntax Trees, Bottom-Up Evaluation of S-Attributed Definitions, L-Attributed Definitions, syntax directed definitions and translation schemes.

<b>3.1</b>	<b>Introduction.....</b>	<b>3-1</b>
3.1.1	The Role of the Parser (Syntax Analyzer).....	3-2
<b>3.2</b>	<b>Context Free Grammar .....</b>	<b>3-2</b>
3.2.1	Left Recursion.....	3-4
3.2.2	Left Factoring.....	3-5
3.2.3	Top Down and Bottom Up Parsing Algorithms .....	3-5
<b>3.3</b>	<b>Top-Down Parser .....</b>	<b>3-6</b>
3.3.1	Types of Top Down Parser .....	3-7
3.3.2	Non Recursive Predictive Parsing (PP1 Model).....	3-13
3.3.3	Construction of the Table.....	3-14
3.3.4	LL(1) Grammars .....	3-17
<b>3.4</b>	<b>Bottom-Up Parser .....</b>	<b>3-21</b>
<b>3.5</b>	<b>LR Parser .....</b>	<b>3-22</b>
3.5.1	LR parser structure and algorithm.....	3-23
3.5.2	SLR Parsing Table.....	3-24
3.5.3	Constructing Canonical LR Parsing Table.....	3-29
3.5.4	Constructing LALR Parsing Table .....	3-33
3.5.5	Difference between SLR, Canonical and LALR.....	3-33
<b>3.6</b>	<b>Using Ambiguous Grammars .....</b>	<b>3-41</b>
<b>3.7</b>	<b>Operator - Precedence Parsing .....</b>	<b>3-43</b>
<b>3.8</b>	<b>Parser Generator (YACC).....</b>	<b>3-46</b>
<b>3.9</b>	<b>Introduction to SDD and SDT .....</b>	<b>3-50</b>
3.9.1	Syntax-Directed Definitions and Attribute Grammar.....	3-50
<b>3.10</b>	<b>Construction of Syntax Trees.....</b>	<b>3-56</b>
<b>3.11</b>	<b>Bottom-Up Evaluation of S Attributed Definitions.....</b>	<b>3-58</b>
<b>3.12</b>	<b>L- Attributed.....</b>	<b>3-60</b>
<b>3.13</b>	<b>Translation Schemes.....</b>	<b>3-61</b>

**Chapter 4 : Error Recovery**

**4-1 to 4-5**

**Syllabus** : Error Detection & Recovery, Ad-Hoc and Systematic Methods.

<b>4.1</b>	<b>Introduction.....</b>	<b>4-1</b>
<b>4.2</b>	<b>Overview of Error Handling .....</b>	<b>4-1</b>
<b>4.3</b>	<b>Error Detection .....</b>	<b>4-2</b>
4.3.1	How Errors are Detected ? .....	4-2
4.3.2	Where Errors are Detected ? .....	4-3
<b>4.4</b>	<b>Error Recovery.....</b>	<b>4-3</b>
4.4.1	Ad-Hoc Recovery Mechanisms .....	4-3
4.4.2	Systematic Methods (Formalized General Approach) .....	4-4
4.4.2(A)	Syntax Directed Recovery.....	4-4
4.4.2(B)	Secondary Error Recovery .....	4-4
4.4.2(C)	Context Sensitive Recovery.....	4-5

**Chapter 5 : Intermediate Code Generation**

**5-1 to 5-24**

**Syllabus** : Variants of Syntax Trees, Three-Address Code, Types and Declarations, Translation of Expressions, Type Checking, Syntax Directed Translation Mechanisms, Attributed Mechanisms and Attributed Definition.

<b>5.1</b>	<b>Introduction.....</b>	<b>5-1</b>
<b>5.2</b>	<b>Intermediate Languages.....</b>	<b>5-1</b>
5.2.1	High Level Representation .....	5-1
5.2.2	Low Level Representation.....	5-1
5.2.3	Graphical Representations.....	5-1
5.2.4	Three Address Code .....	5-2
5.2.5	Types of Three-Address Statements .....	5-3
<b>5.3</b>	<b>Different Intermediate Forms .....</b>	<b>5-4</b>
<b>5.4</b>	<b>Syntax Directed Translation Mechanisms and Attributed Mechanisms and Attributed Definition .....</b>	<b>5-8</b>
5.4.1	Declaration.....	5-8
5.4.2	Assignment Statements .....	5-9
5.4.3	Addressing Array Elements .....	5-10
5.4.4	Boolean Expressions.....	5-14

5.4.4(A) Short-Circuit Code.....	5-15
5.4.4(B) Flow of Control Statements.....	5-16
5.4.4(C) Control-Flow Translation of Boolean Expressions.....	5-17
5.4.5 Case Statements .....	5-19
5.4.6 Procedure Calls.....	5-20
<b>5.5 Type Checking .....</b>	<b>5-20</b>
5.5.1 Type Systems.....	5-20
5.5.2 Type Expressions.....	5-21
<b>5.6 Writing a Simple Type Checker.....</b>	<b>5-22</b>
5.6.1 Type Checking of Expression .....	5-22
5.6.2 Type Checking of Statements .....	5-23
<b>5.7 Type Conversions.....</b>	<b>5-23</b>
5.7.1 Coercions.....	5-23

**Chapter 6 : Runtime Environments**

**6-1 to 6-18**

**Syllabus :** Source Language Issues, Storage Organization. Stack Allocation of Space, Access to Nonlocal Data on the Stack, Heap Management.

<b>6.1 Introduction.....</b>	<b>6-1</b>
<b>6.2 Source Language Issues .....</b>	<b>6-1</b>
6.2.1 Procedures .....	6-1
6.2.2 Activation Trees .....	6-2
6.2.3 Control Stacks .....	6-4
6.2.4 The Scope of a Declaration .....	6-4
6.2.5 Binding of Names.....	6-4
<b>6.3 Storage Organization .....</b>	<b>6-5</b>
6.3.1 Subdivision of Run-Time Memory .....	6-5
6.3.2 Compile-Time Layout of Local Data .....	6-6
<b>6.4 Storage Allocation of Space .....</b>	<b>6-6</b>
6.4.1 Static Allocation .....	6-7
6.4.2 Stack Allocation.....	6-8
6.4.3 Heap Allocation.....	6-9

6.4.4	Dangling References.....	6-9
6.4.5	Comparison between Static Allocation, Stack Allocation and Heap Allocation .....	6-9
<b>6.5</b>	<b>Access to Nonlocal Data on the Stack .....</b>	<b>6-10</b>
6.5.1	Static Scope or Lexical Scope for Block Structured Language .....	6-10
6.5.2	Lexical Scope without Nested Procedures .....	6-12
6.5.3	Lexical Scope with Nested Procedures .....	6-12
6.5.4	Nesting Depth.....	6-13
6.5.5	Access Links.....	6-13
6.5.6	Procedure Parameters .....	6-14
6.5.7	Displays .....	6-14
6.5.8	Dynamic Scope .....	6-16
<b>6.6</b>	<b>Heap management.....</b>	<b>6-17</b>

**Chapter 7 : Code Generation and Optimization**

**7-1 to 7-25**

**Syllabus :** Issues in the Design of a Code Generator, The Target Language, Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, A Simple Code Generator, Machine dependent optimization, Machine independent optimization Error detection of recovery.

<b>7.1</b>	<b>Issues in the Design of a Code Generator.....</b>	<b>7-1</b>
7.1.1	Input to Code Generation Phase (or Input to Code Generator).....	7-1
7.1.2	Target Programs .....	7-3
7.1.3	Memory Management.....	7-4
7.1.4	Instruction Selection .....	7-4
7.1.5	Register Allocation and Assignments.....	7-5
7.1.6	Choice of Evaluation Order .....	7-5
7.1.7	Approaches to Code Generation .....	7-5
<b>7.2</b>	<b>The Target Language .....</b>	<b>7-6</b>
<b>7.3</b>	<b>Addresses in the Target Code .....</b>	<b>7-6</b>
7.3.1	Addressing Modes.....	7-6
7.3.2	Instruction Cost.....	7-7
<b>7.4</b>	<b>Basic Blocks and Flow Graphs.....</b>	<b>7-7</b>
7.4.1	Introduction.....	7-7
7.4.2	Basic Block and Algorithm to find Basic Block.....	7-7

7.4.3	Transformations on Basic Blocks.....	7-10
7.4.3(A)	Structural Presuming Transformations.....	7-10
7.4.3(B)	Algebraic Transformations on Basic Block.....	7-11
7.4.4	Flow Graph.....	7-11
7.4.5	Concept of Loop with Respect to Basic Block and Flow Graph.....	7-13
<b>7.5</b>	<b>Optimization of Basic Blocks .....</b>	<b>7-13</b>
7.5.1	Elimination of Common Subexpression .....	7-13
7.5.1(A)	Elimination of Global Common Sub Expression.....	7-14
7.5.1(B)	Algorithm for Elimination of Global Common Subexpression.....	7-15
7.5.2	Loop Invariant Code Motion.....	7-16
7.5.2(A)	Algorithm for Code Motion.....	7-16
7.5.3	Removal of Induction Variable.....	7-18
7.5.4	Reduction in Strength.....	7-19
7.5.4(A)	Algorithm for Strength Reduction.....	7-19
<b>7.6</b>	<b>A Simple Code Generator.....</b>	<b>7-20</b>
7.6.1	Code Generation Algorithm .....	7-21
7.6.2	The Function getreg( ).....	7-21
<b>7.7</b>	<b>Machine Dependent Optimization .....</b>	<b>7-24</b>
7.7.1	Machine Independent Optimization Error Detection of Recovery .....	7-24

**Chapter 8 : Instruction-Level Parallelism**

**8-1 to 8-23**

**Syllabus** : Processor Architectures, Code-Scheduling Constraints, Basic-Block Scheduling, Pass structure of assembler.

<b>8.1</b>	<b>Introduction.....</b>	<b>8-1</b>
<b>8.2</b>	<b>Processor Architectures .....</b>	<b>8-1</b>
8.2.1	Instruction Pipelines and Branch Delays.....	8-1
8.2.2	Pipelined Execution.....	8-2
8.2.3	Multiple Instruction Issue.....	8-2
<b>8.3</b>	<b>Code-Scheduling Constraints.....</b>	<b>8-3</b>
8.3.1	Data dependence .....	8-3
8.3.2	Finding Dependences Among Memory Accesses.....	8-3
8.3.3	Trade-off between Register Usage and Parallelism .....	8-4

---

8.3.4	Phase Ordering Between Register Allocation and Code Scheduling.....	8-4
8.3.5	Control Dependence.....	8-4
8.3.6	Speculative Execution Support.....	8-4
8.3.7	A Basic Machine Model.....	8-5
<b>8.4</b>	<b>Basic-Block Scheduling.....</b>	<b>8-5</b>
8.4.1	Data-Dependence Graphs.....	8-5
8.4.2	List Scheduling of Basic Blocks.....	8-5
8.4.3	Prioritized Topological Orders.....	8-6
<b>8.5</b>	<b>Pass structure of Assembler.....</b>	<b>8-6</b>
8.5.1	Single Pass Assembler.....	8-6
8.5.2	Design of Two Pass Assembler.....	8-14
8.5.3	Intermediate Code Forms (Representations) for an Assembler.....	8-17
8.5.4	Comparison between Variant I and Variant II.....	8-22
8.5.5	PASS II.....	8-22

---