AMRITA VISHWA VIDYAPEETHAM

Academic year: **2022-23** Semester: **ODD**

Department: **CSE**  Programme: **B.Tech (VII Sem)**

Course Code: **19CSE401** Course Title: **Compiler Design**

L-T-P-C: **2-0-3-3** Evaluation Pattern: **70:30**

**Course Overview**

* This course aims at introducing the major areas of programming language translation and compiler construction.
* The course emphasizes techniques that have direct application to construction of compilers.

**Syllabus**

# Unit – 1

Compiler: Definition, Objectives, Structure – Overview of translation. Scanners: Table-driven. Parsers: LL(1), LALR(1).

# Unit - 2

Context-Sensitive Analysis: Attribute grammar – Ad Hoc Syntax Directed Translation. Intermediate Representations: Abstract Syntax Tree, Three Address code. Symbol Tables: Hash Table.

# Unit - 3

Procedure Abstraction: Access Links. Optimization: Local Value Numbering, Superlocal Value Numbering, Liveness Analysis.

# Text and Reference Books

1. **Cooper, Keith, and Linda Torczon, Engineering a Compiler, Second Edition, Morgan Kaufman, 2011. (Major Textbook followed)**
2. Parr T. Language implementation patterns: create your own domain-specific and general programming languages. Pragmatic Bookshelf; First Edition, 2010.
3. Mak R. Writing compilers and interpreters: a software engineering approach. John Wiley & Sons; Third Edition, 2009.
4. Appel W Andrew and Jens Palesberg, Modern Compiler Implementation in Java, Cambridge University Press, Second Edition, 2002.
5. Aho, Alfred V., Monica S. Lam, Ravi Sethi, and Jeffrey Ullman, Compilers: Principles, Techniques and Tools, Prentice Hall, Second Edition, 2006.

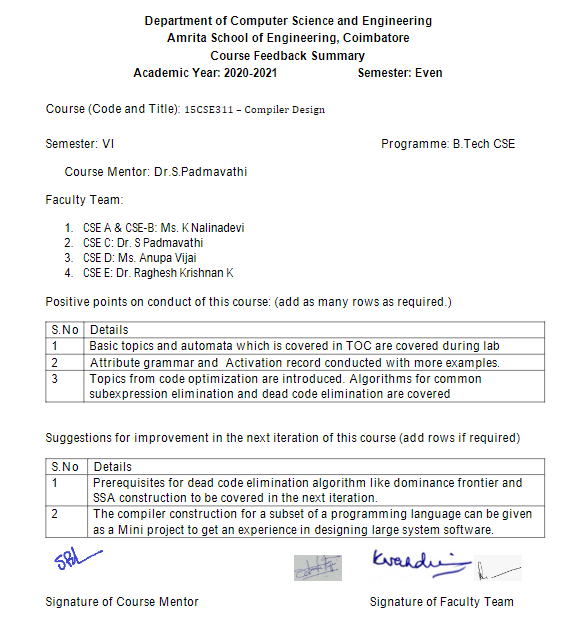
**Course Outcomes**

|  |  |
| --- | --- |
|  | **Course Outcome** |
| CO 1 | Apply theoretical concepts for the analysis of program structure. |
| CO 2 | Apply theoretical concepts and ad hoc techniques to translate high level structures to intermediate representations. |
| CO 3 | Analyze the design of data structures for compile-time code generation. |
| CO 4 | Analyze the design of data structures for run-time code generation. |
| CO 5 | Apply algorithms to improve the performance of the translated code. |

**CO-PO Mapping**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PS O1 | PS O2 |
| CO1 | 2 | 1 | 2 | 1 | 2 |  |  |  |  |  |  |  | 3 | 2 |
| CO2 | 2 | 3 | 2 |  | 2 |  |  |  |  |  |  |  | 3 | 2 |
| CO3 | 2 | 3 | 3 |  |  |  |  |  |  |  |  |  | 3 | 2 |
| CO4 | 2 | 3 | 3 |  |  |  |  |  |  |  |  |  | 3 | 2 |
| CO5 | 2 | 2 | 1 | 1 |  |  |  | 2 |  |  |  |  | 3 | 2 |

# Previous Course Committee Feedback Summary of 15CSE311 – Compiler Design



**Course Plan**

**Tools used in the lab**: ANTLR for front end development.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lecture No(s)** | **Topics** | **Key-words** | **Objectives** | **Sections and pages**  **(italics)**  **from Text**  **Book \*** | **Remarks** |
| 1-3 | Introduction | Compiler Structure, Overview of translation | To know the overall structure of the compiler | *Chapter-1, textbook* | PPT available |
| Lab -1 | Installation of ANTLR and Design of lexical Analyser | Regular Expression, NFA, DFA, Lexer in ANTLR | To design a lexical analyser in ANTLR | *ANTLR manual* |  |
| 4-5 | Syntax analysis:   * RE vs CFG * Top-down parsing vs bottom-up parsing | Derivation, Reduction, Ambiguous grammar, Sentential Form, Parse Tree | To analyse the structural correctness of the source program and generates parse tree as output | *Chapter-3, textbook* | PPT available |
| Lab-2 | ANTLR Lexer practice+ LL(1) parser design (theory) | RDP, Left recursion, Left factoring, LL(1). | To learn the construction and working of top-down parser | *Chapter-3, textbook* | PPT available |
| 6-7 | Bottom-up parsing: LR(0) | Handle, LR(0) | To learn the construction and working of bottom-up parser | *Chapter-3, textbook* | PPT available |
| Lab-3 | ANTLR Parser practice+ LALR(1) | LR(1), Error types LALR(1), Hierarchy of parsers | To learn the construction and working of bottom-up parser.  To construct a parser using ANTLR. | *Chapter-3, textbook* | PPT available |
| 8-9 | Attribute Grammar | Synthesized and inherited grammar, evaluation of attribute grammar and circularity. | To perform type checking and coercion on the variables in source program | *Chapter-4, textbook* | PPT available |
| Lab-4 | Lab evaluation -1 | Lexer + parser design and implementation for a programming language of the students’ choice in ANTLR | To check the working of Lexer and Parser design in ANTLR for an academic language of student’s choice | *Assessment* |  |
| Periodical Test -1 | | | | | |
| 10-11 | Intermediate Representation + Symbol table | Abstract Syntax Tree, three address code, Hash Table, stack, hash table with stack | To construct symbol tables and represent parse tree as IR codes. | *Chapter-5 in textbook* | PPT available |
| Lab-5 | Procedure Abstraction | Scoping rules and Namespace structure, Activation record on stack | To create, access, store and delete activation records for procedures. | *Chapter-6 in textbook* | PPT available |
| 12 | Procedure Abstraction | communicating between procedures using access links | To create, access, store and delete activation records for procedures. | *Chapter-6 in textbook* | PPT available |
| 13-15 | Code optimization | Definition, typical transformations, Scope of optimization, Local optimization using LVN | To transform the code to achieve optimal code generation. | *Chapter-8 in textbook* | PPT available |
| Lab-6 | Ad Hoc Syntax directed translation in ANTLR | Semantic analysis | To evaluate expression, perform type verification, IR code conversion in ANTLR. | *Practice* |  |
| 16-19 | Code optimization | Global Optimization - finding uninitialized variable | To transform the code to achieve optimal code generation. | *Chapter-8 in textbook* | PPT available |
| Lab-7 | Lab evaluation-2 | Implementation of a semantic analyser in ANTLR for the programming language chosen by the students’ | To evaluate expression, perform type verification in ANTLR | *Practice* |  |
| Lab-8 | Intermediate code generation in ANTLR + Regional Optimization using SVN and loop unrolling | IR in ANTLR | To evaluate expression, perform type verification, IR code conversion in ANTLR. | *Chapter-8 in textbook + Lab Practice* | PPT available |
|  | | Periodical Test -2 | |  |  |
| 20-25 | Iterative Data flow Analysis | Dominance, Live variable Analysis, SSA, Dead code elimination algorithm | To perform optimization of the code for a single thread of control. | *Chapter – 9, 10 in textbook* | PPT available |
| Lab-9 |
| Lab-10 | Lab evaluation-3 | Implementation of a Intermediate code generation in ANTLR for the programming language chosen by the students’ | Intermediate code generation in ANTLR | *Assessment* |  |
| Lab-11 | Self study of Code Optimization in LLVN integration with ANTLR | Implementation of a Code optimizer in LLVN integrated with ANTLR | To implement Code optimization algorithms in LLVN | *Practice* |  |
| Lab-12 | Lab evaluation-4 | Implementation of a Code optimization algorithm in LLVN integrated with ANTLR for the programming language chosen by the students’ | To implement Code optimization algorithms in LLVN for the programming language. | *Assessment* |  |
|  | | **End Semester Examination** | |  |  |

**Evaluation Pattern: 70:30**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | **Weightage** | **Exam Name/type** | **Max marks** |
| Periodical 1 | **10** | Periodical 1 | **50** |
| Periodical 2 | **10** | Periodical 2 | **50** |
| Continuous Assessment (Theory) | **10** | Quiz-1: Parser  Quiz -2: Attribute Grammar  Quiz -3: IR and symbol Table  Quiz - 4: Access link  Quiz -5: Optimization | **10**  **10**  **10**  **10**  **10** |
| Continuous Assessment (Lab) | **40** | Lab Evaluation -1: ANTLR scanner and Parser  Lab evaluation - 2: ANTLR syntax directed translation  Lab evaluation - 3: ANTLR semantic analyser  Lab evaluation - 4: LLVN code optimization | **10**  **10**  **10**  **10** |
| End Semester | **30** | End Semester | **100** |

Lab Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lecture No(s)** | **Topics** | **Key-words** | **Objectives** | **Sections and pages**  **(italics)**  **from Text**  **Book \*** | **Remarks** |
| Lab -1 | Installation of ANTLR and Design of lexical Analyser | Regular Expression, NFA, DFA, Lexer in ANTLR | To design a lexical analyser in ANTLR | *ANTLR manual* |  |
| Lab-2 | ANTLR Lexer practice+ LL(1) parser design (theory) | RDP, Left recursion, Left factoring, LL(1). | To learn the construction and working of top-down parser | *Chapter-3, textbook* | PPT available |
| Lab-3 | ANTLR Parser practice+ LALR(1) | LR(1), Error types LALR(1), Hierarchy of parsers | To learn the construction and working of bottom-up parser.  To construct a parser using ANTLR. | *Chapter-3, textbook* | PPT available |
| Lab-4 | Lab evaluation -1 | Lexer + parser design and implementation for a programming language of the students’ choice in ANTLR | To check the working of Lexer and Parser design in ANTLR for an academic language of student’s choice | *Assessment* |  |
| Lab-5 | Procedure Abstraction | Scoping rules and Namespace structure, Activation record on stack | To create, access, store and delete activation records for procedures. | *Chapter-6 in textbook* | PPT available |
| Lab-6 | Ad Hoc Syntax directed translation in ANTLR | Semantic analysis | To evaluate expression, perform type verification, IR code conversion in ANTLR. | *Practice* |  |
| Lab-7 | Lab evaluation-2 | Implementation of a semantic analyser in ANTLR for the programming language chosen by the students’ | To evaluate expression, perform type verification in ANTLR | *Practice* |  |
| Lab-8 | Intermediate code generation in ANTLR + Regional Optimization using SVN and loop unrolling | IR in ANTLR | To evaluate expression, perform type verification, IR code conversion in ANTLR. | *Chapter-8 in textbook + Lab Practice* | PPT available |
| Lab-9 | Iterative Data flow Analysis | Dominance, Live variable Analysis, SSA, Dead code elimination algorithm | To perform optimization of the code for a single thread of control. | *Chapter – 9, 10 in textbook* | PPT available |
| Lab-10 | Lab evaluation-3 | Implementation of a Intermediate code generation in ANTLR for the programming language chosen by the students’ | Intermediate code generation in ANTLR | *Assessment* |  |
| Lab-11 | Self study of Code Optimization in LLVN integration with ANTLR | Implementation of a Code optimizer in LLVN integrated with ANTLR | To implement Code optimization algorithms in LLVN | *Practice* |  |
| Lab-12 | Lab evaluation-4 | Implementation of a Code optimization algorithm in LLVN integrated with ANTLR for the programming language chosen by the students’ | To implement Code optimization algorithms in LLVN for the programming language. | *Assessment* |  |