**Mandsaur University**

**Bachelor of Technology (Computer Science and Engineering)**

**Semester-V**

L-2 T-1 P-2 C-4

**CSE340 TR1 : Compiler Design**

**Course Objectives**

1. To introduce the major concept areas of language translation and compiler design.
2. To introduce students to the concepts underlying the design and implementation of language processors.
3. To learn context free grammars, compiler parsing techniques, construction of abstract syntax trees, symbol tables, and actual code generation.

**Course Outcomes (CO)**

CO1. Analysis of different phases of the compiler.

CO2.Create different types of parsers i.e. Top-Down and Bottom-up parsers and construct LL, SLR, CLR, and LALR parsing table

CO3.Analyze data structures used for symbol table and runtime organization and errors in various phases

CO4.Create intermediate code for statements in high level language.

CO5.Apply code optimization and code generation techniques to create target code.

**Articulation Matrix**

*(Program Articulation Matrix is formed by the strength of correlation of COs with POs and PSOs. The strength of correlation is indicated as 3 for substantial (high), 2 for moderate (medium) correlation, and 1 for slight (low) correlation)*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO/PO/PSO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | **1** | **1** | **-** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **1** | **2** | **-** |
| **CO2** | **1** | **3** | **1** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **3** | **1** | **-** |
| **CO3** | **1** | **1** | **-** | **-** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **2** | **2** | **-** |
| **CO4** | **1** | **-** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **2** | **3** | **2** |
| **CO5** | **1** | **-** | **-** | **-** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **3** | **1** | **1** |

### High-3 Medium-2 Low-1

### **Unit-1 9 Hours**

Introduction of Compiler, Major data Structure in compiler, BOOT Strapping & Porting, Compiler structure: analysis-synthesis model of compilation, various phases of a compiler, Lexical analysis: Input buffering, Specification & Recognition of Tokens, LEX.

### **Unit-2 9 Hours**

Syntax analysis: CFGs, Top down parsing, Brute force approach, recursive descent parsing, Transformation on the grammars, predictive parsing, bottom up parsing, operator precedence parsing, LR parsers (SLR,LALR, LR),Parser generation. Syntax directed definitions: Construction of Syntax trees, Bottom up evaluation of S-attributed definition, L-attribute definition, Top down translation, Bottom Up evaluation of inherited attributes Recursive Evaluation, Analysis of Syntax directed definition.

### **Unit -3 9 Hours**

Type checking: type system, specification of simple type checker, equivalence of expression, types, type conversion, overloading of functions and operations, polymorphic functions. Run time Environment: storage organization, Storage allocation strategies, parameter passing, Dynamic storage allocation , Symbol table.

### **Unit -4 9 Hours**

Intermediate code generation: Declarations, Assignment statements, Boolean expressions, Case statements, Back patching, Procedure calls Code Generation: Issues in the design of code generator, Basic block and flow graphs, Register allocation and assignment, DAG representation of basic blocks, peephole optimization, generating code from DAG.

### **Unit-5 9 Hours**

Introduction to Code optimization: sources of optimization of basic blocks, loops in flow graphs, dead code elimination, loop optimization, Introduction to global data flow analysis, Code improving transformations ,Data flow analysis of structure flow graph Symbolic debugging of optimized code.

**Total: 45 Hours**

## Text books:

1. V. Aho, R. Sethi, and J. D. Ullman, “Compilers:Principles,Techniques and Tools”, Pearson Education

## Reference books:

1. Mak, writing compiler & Interpreters, Willey Pub.Louden, Compiler Construction: Principles and Practice, Cengage Learning.
2. A. C. Holub. Compiler Design in C , Prentice-Hall Inc., 1993.
3. Raghavan, Compiler Design, TMH Publication.
4. Louden. Compiler Construction: Principles and Practice, Cengage Learning.

**Mandsaur University**

**Bachelor of Technology (Computer Science and Engineering)**

**Semester- V**

L-0 T-0 P-2 C-1

**CSE340 PR1 : Compiler Design**

**Course Objectives**

1. To introduce the major concept areas of language translation and compiler design.
2. To introduce students to the concepts underlying the design and implementation of language processors.
3. To learn context free grammars, compiler parsing techniques, construction of abstract syntax trees, symbol tables, and actual code generation.

**Course Outcomes (CO)**

CO1. Analysis of different phases of the compiler.

CO2.Create different types of parsers i.e. Top-Down and Bottom-up parsers and construct LL, SLR, CLR, and LALR parsing table

CO3.Analyze data structures used for symbol table and runtime organization and errors in various phases

CO4.Create intermediate code for statements in high level language.

CO5.Apply code optimization and code generation techniques to create target code.

**Articulation Matrix**

*(Program Articulation Matrix is formed by the strength of correlation of COs with POs and PSOs. The strength of correlation is indicated as 3 for substantial (high), 2 for moderate (medium) correlation, and 1 for slight (low) correlation)*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO/PO/PSO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | **1** | **1** | **-** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **1** | **2** | **-** |
| **CO2** | **1** | **3** | **1** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **3** | **1** | **-** |
| **CO3** | **1** | **1** | **-** | **-** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **2** | **2** | **-** |
| **CO4** | **1** | **-** | **2** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **2** | **3** | **2** |
| **CO5** | **1** | **-** | **-** | **-** | **3** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **3** | **1** | **1** |

### High-3 Medium-2 Low-1

## List of Practicals:



1. Write a program in C/C++to check whether a string belongs to the grammar or not.
2. Write a program in C/C++to identify whether a given string is an identifier or not.
3. Write a program in C/C++ to check whether a given string is a keyword or not.
4. Write a program in C/C++ to implement the token separation operation**.**
5. Write a program in C/C++ to compute FIRST of non-terminals.
6. Write a program in C/C++ to compute FOLLOW(A).
7. Write a program to implement the Lexical analysis using C.
8. Write a program in C/C++ to implement recursive descendent parsing.
9. Write a program in C/C++ to calculate LEADING of non terminals.
10. Write a program in C/C++ to implement the Symbol table operation.
11. Write a program in C/C++ to implement operator precedence parsing.

**Total: 30 Hours**

## Text books:

1. V. Aho, R. Sethi, and J. D. Ullman, “Compilers:Principles,Techniques and Tools”, Pearson Education

## Reference books:

1. Mak, writing compiler & Interpreters, Willey Pub.Louden, Compiler Construction: Principles and Practice, Cengage Learning.

2. A. C. Holub. Compiler Design in C , Prentice-Hall Inc., 1993.

3. Raghavan, Compiler Design, TMH Publication.

1. Louden. Compiler Construction: Principles and Practice, Cengage Learning.