Principles of Compiler Design

Course Title	Principles of Compiler Design
Course Code	PPSCSMAJM101
Total Number of Lectures	60
Credits	04
Introduction	● Provide an understanding of the fundamental principles in compiler design. ● Provide the skills needed for building compilers for various situations that one may encounter in a career in Computer Science. ● Learn the process of translating a modern high-level language to executable code required for compiler construction. ● To apply the optimization techniques to have a better code for code generation.
Course Outcomes	● Understand the theoretical foundations and concepts underlying the design and implementation of compilers. ● Acquire knowledge about the different phases of the compilation process ● Learn how to design and implement lexical analyzers and parsers ● Gain hands-on experience in building semantic analyzers ● Understand intermediate code generation and Implement optimization techniques ● Gain practical experience in code generation ● Familiarity with runtime environments and Develop skills in error handling and debugging ● Explore advanced topics in compiler design and Apply knowledge to practical projects
Units	Given Below

UNITS

Unit Number	CONTENT	NUMBER OF LECTURES
I	**Unit 1:** Introduction to Compilers o Compilers and translators o Why do we need translators? The structure of a compiler, Lexical analysis Syntax analysis, Intermediate code generation, Optimization , Code generation , Error handling , Compiler writing tools, Getting started	12L
II	**Unit 2: Front end of Compiler** **Introduction to Compiler Design**: Role and importance of compilers, Phases of compilation process, Compiler architecture and components **Lexical Analysis**: Role of lexical analyzer, Regular expressions and finite automata, Lexical analyzer generators (e.g., Lex) **Syntax Analysis:** Role of parser, Context-free grammars, Top-down parsing (LL parsing) Bottom-up parsing (LR parsing), Syntax analyzer generators (e.g., Yacc/Bison)	12L

	Semantic Analysis: Role of semantic analyzer, Symbol table management, Type checking and type systems, Attribute grammars	
III	**Unit 3: Back end of Compiler** **Intermediate Code Generation**: Intermediate representations (IR), Three- address code generation, Quadruples and triples, Syntax-directed translation **Code Optimization**: Data flow analysis, Common subexpression elimination, Constant folding and propagation, Loop optimization techniques **Code Generation:** Code generation techniques, Target machine description, Register allocation, Instruction selection and scheduling **Runtime Environments:** Activation records and stack management. Heap memory management, Call and return mechanisms, Exception handling	12L
IV	**Unit 4**: **Introduction to Compiler Tools, Techniques and Advanced Topics in Compiler Design:** Lexical and syntax analyzer generators, Code generation frameworks (e.g., LLVM), Debugging and testing compilers, Just-in-time (JIT) compilation, Parallel and concurrent programming support, Compiler optimization frameworks, Domain-specific language (DSL) compilation **Lexical and Syntax Error Handling:** Error recovery strategies **Error reporting and handling**: Error detection and recovery Errors, Lexical-phase errors, Syntactic-phase errors, Semantic errors	12L

Principles of Compiler Design Practical
Practical code PPSCSMAJM1P1 and No. of Credits 02

Sr. No.	Title of the Practical
1	Write a program to generate tokens for given lexeme
2	Write a c program to find whether the string is parsing or not.
3	Write a program to implement simple lexical analyzer using c language.
4	Write a program to generate syntax tree.
5	Write a program to construct nfa for the given regular expression
6	Write a program to construct dfa for the given regular expression
7	Write a program to implement symbol table
8	Write a program to find first & follow from a grammar.
9	Write a program to implement construction of operator precedence parse table
10	Write a c program to implement simple lr parsing algorithm

Reading List (Books)

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman *Compilers: Principles, Techniques, and Tools*" by 2nd Edition, Pearson Publication, 2006 ISBN-13: 978 0321486813
2. Dick Grune, Henry E. Bal, Cariel T. H. Jacobs, "*Modern Compiler Design*"

Reference book

1. Andrew W. Appel."*Modern Compiler Implementation in C*". 3rd Edition, Cambridge University Press, 2020, ISBN-13: 978-1108426631
2. D. M. Dhamdhere."*Principles of Compiler Design*", 2nd Edition Publisher: McGraw-Hill Education, 2017, ISBN-13: 978-9339204608

NoSQL Technologies

Course Title	NoSQL Technologies
Course Code	PPSCSMAJM102
Total Number of Lectures	60
Credits	04