

Compiler Construction					
Credit Hours:	2	Course Code:	COSC-4102	Prerequisites:	Theory of Automata
Course Learning Outcomes (CLOs):					
At the end of the course, the students will be able to:				Domain	BT Level*
1. Understand semantic analysis including static checking, intermediate representations and attribute grammars				C	3
2. Analyse different compiler optimization techniques				C	4
3. Perform lexical analysis				C	5
4. Perform parsing techniques using parsing generators				C	5
5. Design and implement a compiler for a small language				C	6
* BT= Bloom's Taxonomy, C=Cognitive Domain, P=Psychomotor Domain, A=Affective Domain					
#	SDG	Relevant CLOs			
1	SDG 4: Quality Education	CLO 1: Understand semantic analysis including static checking, intermediate representations and attribute grammars			
2	SDG 9: Industry, Innovation and Infrastructure	CLO 5: Design and implement a compiler for a small language			
Course Contents:					
Introduction to Compilers, The Phases of Compiler, Cousins of Compiler, Grouping of Phases, Compiler Construction Tools, Simple One-Pass Compiler, Syntax Definition and Translation, Parsing, A Translator for Simple Expression, Lexical Analysis, Incorporation of Symbol Table, Abstract Stack Machine, Synthesis of Techniques, Lexical Analyzer, Input Buffering, Token Specification and Recognition, Language Specifying Lexical Analyzers, Automata, Deterministic and Non-Deterministic Automata (NFA), Conversion from Regular Expression to NFA, Design of Lexical Analyzer Generator, Pattern Matching, Optimization of DFA-Based Pattern Matchers, Syntax Analysis, The Role of Parsers, Context Free Grammar, Writing Grammar, Top Down and Bottom-Up Parsing, Operator Precedence Parsing, LR Parsers, Removing Ambiguities in Context Free Grammar, Parser Generators, Problems Solving, Syntax Directed Definition, Construction of Syntax Trees, Bottom-Up Evaluation of S-Attributed Definitions, L-Attributed Definition, Top Down Translation, Bottom-Up Evaluation, Recursion, Type Checking, Type Systems, Specification of Simple Type Checker, Equivalence of Type Expression, Type Conversion, Intermediate Code Generation and Optimization					
Teaching Methodology:					
Lectures, Written Assignments, Practical labs, Semester Project, Presentations.					
Course Assessment:					
Mid Term Exam, Home Assignments, Quizzes, Project, Presentations, Final Exam.					
Reference Material:					
1. Compilers: Principles, Techniques and Tools, A. V. Aho, R. Sethi and J. D. Ullman, Addison-Wesley; 2 nd Edition 2006; ISBN-10: 0321486811					
2. Modern Compiler Design, D. Grune, H. E. Bal, C. J. H. Jacobs, K. G. Langendoen, John Wiley, 2 nd Edition, 2012; ISBN-10: 1461446988					
3. Modern Compiler Implementation in C by A. W. Appel, M. Ginsburg, Cambridge University Press, 2004. ISBN-10: 0521607655					

Compiler Construction - Lab					
Credit Hours:	1	Course Code:	COSC-4202	Prerequisites:	None
Course Learning Outcomes (CLOs):					
At the end of the course, the students will be able to:				Domain	BT Level*
1. Understand semantic analysis including static checking, intermediate representations and attribute grammars				C	3
2. Analyse different compiler optimization techniques				C	4
3. Perform lexical analysis				C	5
4. Perform parsing techniques using parsing generators				C	5
5. Design and implement a compiler for a small language				C	6
* BT= Bloom’s Taxonomy, C=Cognitive Domain, P=Psychomotor Domain, A=Affective Domain					
#	SDG	Relevant CLOs			
1	SDG 4: Quality Education	CLO 1: Understand semantic analysisincluding static checking, intermediate representations and attribute grammars			
2	SDG 9: Industry, Innovation and Infrastructure	CLO 5: Design and implement a compiler for a small language			
Course Contents:					
Introduction to C#/intended language, Recognition of operators/variables exercises, Recognition of keywords/constants exercises, Input Buffering Scheme implementation, Symbol Table Implementation, Lexical analyzer implementation, Finding the first set of a given grammar practically, Finding the follow set of a given grammar practically, DFA Implementation, Stack parser using SLR performance, Semantic Analyzer Implementation, Parser implementation, Lexical Analyzer & Symbol Table (Ph-1) Integration, Ph-1 and Parser (Ph-2) Integration, Ph-2 and Semantic Analyzer Integration, Complete Mini Compiler implementation.					
Teaching Methodology:					
Lectures, Written Assignments, Practical labs, Semester Project, Presentations.					
Course Assessment:					
Mid Term Exam, Home Assignments, Quizzes, Project, Presentations, Final Exam.					
Reference Material:					
1. Compilers: Principles, Techniques and Tools, A. V. Aho, R. Sethi and J. D. Ullman, Addison-Wesley; 2nd Edition 2006; ISBN-10: 0321486811					
2. Modern Compiler Design, D. Grune, H. E. Bal, C. J. H. Jacobs, K. G. Langendoen, John Wiley, 2nd Edition, 2012; ISBN- 10: 1461446988					
3. Modern Compiler Implementation in C by A. W. Appel, M. Ginsburg, Cambridge University Press, 2004. ISBN-10: 0521607655					