PIR MEHR ALI SHAH ARID AGRICULTURE UNIVERSITY
University Institute of Information Technology

Compiler Construction (CS-636)

Credit Hours:	3(2-2)	**Prerequisites:**	Programming Fundamentals Automata Theory

Course Learning Outcomes (CLOs)

At the end of course the students will be able to:	**Domain**	**BT Level***
1. Understanding compiler structure, important components, their inputs and outputs	C	2,3
2. Understanding parsing algorithms including top-down and bottom-up parsing algorithms	C	4,5
3. Understanding data structures and their implementation details used by the compiler including symbol table and literal table	C	3
4. Learning how to optimize code to improve performance	C	6

*BT- Bloom's Taxonomy, C=Cognitive domain, P=Psychomotor domain, A=Affective domain

Course Contents:

Structure of Compiler Construction, Components their Inputs & Outputs, Lexical Analyzer, Syntax Analyzer, Semantic Analyzer, Parse Tree, Abstract Syntax Tree, Regular Expressions, Context Free Grammars, Attribute Grammars, Intermediate Code Representations, three-address code, Directed Acyclic Graphs, Runtime Environments, Symbol Table, Storage Management, Code Generator, Intermediate Code Optimizer, Target Code Optimizer

Course Objective:

- To teach students the basic concepts of compilers, their components and how they work together
- To get lexical analyzer and syntax analyzer implemented of any programming language

Teaching Methodology:

Lectures, Written Assignments, Semester Project

Courses Assessment:

Mid Exam, Home Assignments, Quizzes, Project, Presentations, Final Exam

Reference Materials:

1. Compiler Construction by Kenneth C. Louden and Glagotia
2. Modern Compiler Implementation in C, By Andrew W. Appel, Maia Ginsburg, Contributor Maia Ginsburg, Cambridge University Press, 2004.
3. Modern Compiler Design by Dick Grune, Henri E. Bal, Ceriel J. H. Jacobs, Koen G. Langendoen, 2003, John Wiley & Sons.
4. Compiler Construction by Dr Rafa E Al-Qutaish LAP Lambert Academic Publishing, 04-Feb-2011

Week/Lecture #		**Theory**	**Practical**
Week 1	Lecture-I	Introduction of Compiler and brief history	

	Lecture-II	Phases of Compiler, Cousins of compiler	
Week 2	Lecture-I	Lexical Analyzer, Overview and Role of Lexical Analyzer, Tokens, Lexemes	Implementation issues of Scanners
	Lecture-II	Patterns, Specification of tokens, recognition of tokens.	Algorithm to perform lexical analysis
Week 3	Lecture-I	Finite automata, NFA, DFA, Conversion from a regular expression to an NFA	
	Lecture-II	NFA problems, NFA & DFA Comparison, Design of a Lexical Analyzer.	Implementation of FAs and their simulations
Week 4	Lecture-I	Symbol Table Manager, **o**verview, Symbol Table organization, Classification of symbol table	
	Lecture-II	Symbol table operations, working, implementation.	Implementation of symbol table using hash tables
Week 5	Lecture-I	Syntax Analyzer, Role of Parser, Context Free grammars, writing a grammar	Understanding CFG of a high-level programming language
	Lecture-II	Types of Parsing, Top down parsing, Bottom up parsing	
Week 6	Lecture-I	Backtracking, Recursive Descent parsing, problems with RD parsing, Predictive parsing	Implementation of Recursive Descent algorithm
	Lecture-II	Transition diagrams for predictive parsers, Non Recursive Predictive Parsing	
Week 7	Lecture-I	Practical Examples and Parser Generators	Working with YACC
	Lecture-II	JFLex and Yacc case studies	Working with Lex
Week 8	Lecture-I	Semantic Analysis**,** Overview of Type Checking, specification of a simple type checker	
	Lecture-II	Equivalence of type expressions	
Midterm Exam			
Week 9	Lecture-I	Type conversions, type rules, type constructors	
	Lecture-II	Scope Rules, Defining scope rules	Defining scope rules for a high-level language
Week 10	Lecture-I	A simple type checker generator	
	Lecture-II	Case studies of type checker	
Week 11	Lecture-I	Intermediate Code Generator, Intermediate languages, declarations	
	Lecture-II	Three address code instructions and their representations	Representing three-address code in data

			structure
Week 12	Lecture-I	Directed Acyclic Graphs, Value Number representation of Directed acyclic graphs	
	Lecture-II	Intermediate Code Generator examples	Implementing DAG
Week 13	Lecture-I	Code Optimization, Overview, the principal of sources of optimization	
	Lecture-II	Optimization of basic blocks, loops, code improvement transformations.	
Week 14	Lecture-I	Role of a code generator, Issues in the design of a code generator	Computing cost of the target program
	Lecture-II	Runtime storage management, simple code generator.	
Week 15	Lecture-I	Detection of syntax errors by compilers and their recovery mechanism	
	Lecture-II	Overview of principles of programming languages. Criteria for selecting programming languages	
Week 16	Lecture-I	Representing concurrency, and analyzing concurrent designs	
	Lecture-II	Demos of semester projects	
Final term Exam			