Compilers	Code & No:	CS 330
	Credits:	3(3,1,0)
	Pre-requisite:	CS 270
	Co-requisite:	None
	Level:	7

Course Description:

In this course students will develop a deeper understanding of modern compiler techniques applied to general-purpose programming languages. It will give students a working knowledge of the foundations, tools, and engineering approaches used in developing formal language translators. Major topics include:

Topics to be Covered:

1) Introduction
2) Lexical analysis, scanning
3) Syntax analysis, parsing
4) Parser generators,
5) Semantic Analysis
6) Intermediate Representation
7) Intermediate code generation
8) Code Optimization
9) Error Detection and Recovery

Course Aims:

1. Understand how the design of a compiler requires most of the knowledge acquire during their study
2. Develop a firm and enlightened grasp of concepts learned earlier in their study like higher level programming, assemblers, automata theory, and formal languages, languages, languages specifications, data structure and algorithms, operating systems and com
3. Develop an in–depth knowledge of major topics in compiler design.
4. Develop a fundamental understanding of various stages of compiling.

Student Outcomes (SOs):

☒(a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

☒(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

☒(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

☐(d) An ability to function effectively on teams to accomplish a common goal

☐(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

☐(f) An ability to communicate effectively with a range of audiences

☐(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

☐(h) Recognition of the need for and an ability to engage in continuing professional development

☒(i) An ability to use current techniques, skills, and tools necessary for computing practice.

☐(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. [CS]

☐(k) An ability to apply design and development principles in the construction of software systems of varying complexity. [CS]

☐(j) An ability to use and apply current technical concepts and practices in the core information technologies of human computer interaction, information management, programming, networking, and web systems and technologies. [IT]

☐(k) An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation, and administration of computer-based systems. [IT]

☐(l) An ability to effectively integrate IT-based solutions into the user environment. [IT]

☐(m) An understanding of best practices and standards and their application. [IT]

☐(n) An ability to assist in the creation of an effective project plan. [IT]

Course Learning Outcomes (CLOs):

1. Understand the principles of compilers construction
2. Understand the basic components of a compiler (e.g. lexical analysis, top-down, bottom-up parsing, context-sensitive analysis, and intermediate code generation)
3. Design and implement a simple compiler
4. Use automatic tools in the development of compilers (e.g. Lex and Yacc)

SOs and CLOs Mapping:

CLO/SO	a	b	c	d	e	f	g	h	i	j	k	l	m	n
CLO1	√													
CLO2	√													
CLO3	√	√	√						√					
CLO4	√	√	√						√					

No.	Topics	Weeks	Teaching hours
1	Introduction to Compilers: language translation, comparison of interpreters and compilers, language translation phases	2	6
2	Lexical Analysis: regular expressions role in lexical scanners, comparison of hand-made scanner and automatically generated scanners, formal definition of tokens, use of finite state automata.	3	9
3	Syntax Analysis: formal definition of grammars, BNF, bottom-up vs. top-down parsing, tabular vs. recursive-descent parsers, error handling	3	9
4	Parsers Implementation: tabular parsers, symbol tables, the use of tools in support of the translation process	3	9
5	Semantic Analysis: data types, type-checking models, type-checking algorithms.	1	3
6	Intermediate Representation, Code Generation: intermediate and object code, intermediate representations, implementation of code generators, tree walking; context sensitive translation	1	3
7	Code optimization: data-flow analysis; loop optimizations	1	3
	Total	**14**	**42**

Textbook:

- A V Aho, R Sethi and J D Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley, 2015, 2nd reprint edition, ISBN-13: 978-0321486813

Essential references:

- Keith Cooper, Linda Torczon, Enginnering a Compiler, 2nd Edition, Elseview, 2012, ISBN-13: 978-0120884780