<table>
<tr><td>Course Title</td><td>Compiler Design Practice</td><td>Course No</td><td colspan="5"></td></tr>
<tr><td rowspan="2">Department/ Specialization</td><td rowspan="2">Computer Science and Engineering</td><td rowspan="2">Credits</td><td>L</td><td>T</td><td>P</td><td colspan="2">C</td></tr>
<tr><td>0</td><td>0</td><td>4</td><td colspan="2">2</td></tr>
<tr><td>Faculty proposing the course</td><td>Faculty, Department of CSE</td><td>Status</td><td>Core</td><td>■</td><td>Elective</td><td colspan="2">□</td></tr>
<tr><td>Offered for</td><td>B.Tech</td><td>Type</td><td>New</td><td>■</td><td>Revision</td><td colspan="2">□</td></tr>
<tr><td>To take effect from</td><td>July 2021</td><td rowspan="2">Submitted for approval</td><td colspan="5" rowspan="2">44th Senate</td></tr>
<tr><td>Prerequisite</td><td>Nil</td></tr>
<tr><td>Learning Objectives</td><td colspan="7">The objective of this course is to train students to design various phases of compiler such as Lexical analyzer, syntax analyzer, semantic analyzer, intermediate code generator, code optimizer and code generator. Students are also exposed to design compiler construction tools such as Lexical Analyzer generator and parser generator. Applications of finite state machine and pushdown automation in compiler design are also taught in this course.</td></tr>
<tr><td>Learning Outcomes</td><td colspan="7">● At the end of the course, students will be able to design a programming language and compiler for the same.
● Students will also be able to write large programs.</td></tr>
<tr><td>Course Contents (with approximate breakup of hours for lecture/ tutorial/practice)</td><td colspan="7">Lexical analyzer implementation in C - Lexical analyser implementation using LEX tool Recursive descent parser implementation in C for an expression grammar - YACC and LEX based implementation for an expressions grammar - YACC implementation of a calculator that takes an expression with digits, + and * and computes and prints its value - Front end implementation of a compiler that generates the three address code for a simple language- Back end implementation of a compiler which takes the three address code (output of previous exercise) and results in assembly language instructions - Implementation of peephole optimization in C.</td></tr>
<tr><td>Essential Reading</td><td colspan="7">1. Alfred Aho, Ravi Sethi and Jeffrey D Ullman, Compilers Principles, Techniques and Tools, Pearson Education, 2003. ISBN: 9780321491695</td></tr>
<tr><td>Supplementary Reading</td><td colspan="7">1. Levine J.R, Mason T, Brown D, Lex & Yacc, OReilly Associates, 1992 ISBN: 9781565920002.
2. Allen I. Holub, Compiler Design in C, Prentice Hall, 2003. ISBN: 9780131550452</td></tr>
</table>