<table>
<thead>
<tr>
<th>Course Number</th>
<th>Course Title</th>
<th>Compiler Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 416</td>
<td>Compiler Construction</td>
<td></td>
</tr>
</tbody>
</table>

**Semester Hours** 3  
**Course Coordinator** SP 20 Khaled Ahmed

**Catalog Description**  
Introduction to compiler construction. Design of a simple complete compiler, including lexical analysis, syntactical analysis, type checking, and code generation.

**Textbooks**  

**References**  

**Course Learning Outcomes**

- To learn the principles of compiler design and implementation.

**Assessment of the Contribution to Student Outcomes**

<table>
<thead>
<tr>
<th>Outcome</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Prerequisites by Topic**  
CS 306 and 311 each with a grade of C or better or graduate standing.
Major Topics Covered in the Course

1. Basic ideas: phases of a compiler, compiler construction tools {2 classes}
2. Language and grammars: basic concepts, classification of grammars (type 0, 1, 2, and 3), reduced grammars and extended BNF notations, regular expressions {4 classes}
3. A simple one-pass compiler: syntax definition, scanner, parsing, syntax directed translation, symbol tables, semantics and code generation {3 classes}
4. Lexical analysis: regular expressions, finite state acceptors, conversion algorithms, token specification, scanner generator (LEX) {6 classes}
5. Syntax analysis: top down parsing, recursive descent and predictive parsers, LL(1) grammars, bottom-up parsing, simple and operator precedence grammars, simple LR parsing, introduction to LALR and canonical LR parsing {6 classes}
6. Type checking: a simple type checker, type conversions {3 classes}
7. Symbol tables: symbol table organization for both block structured and non block structured languages {3 classes}
8. Run-time storage organization: dynamic storage allocation strategies, access to nonlocal names, parameter passing, heap storage {4 classes}
9. Intermediate codes: intermediate languages, quadruples {3 classes}
10. Code generation: issues in code design, target machine, register allocation, simple code generator {6 classes}

NOTE: When course is taken as 500-level credit (CS 591 “Special Topics”), there will be additional requirements such as a research project.