COURSE DESCRIPTION

Approval: 24th Senate Meeting

Course number : CS502
Course Name : Compiler Design
Credit : 3-0-2-4
Prerequisites : CS202: Data Structures and Algorithms;
CS304: Formal Languages and Automata Theory
Intended for : BTech third and final year CSE and MS/PhD
Distribution : Elective for third and final year BTech CSE/EE, MS, PhD
Semester : Odd/Even
Comments : None

---

1. Preamble:

A compiler is a fundamental software necessary to translate computer programs to a form that can be executed on intended machines. Designing a compiler involves learning several aspects of computer science: logic, formalism, mathematics, data structures, algorithms, programming, and so on. This course is intended as a primer to the various stages typical in the design of standard compilers, starting with the front-end stages of compilation, and giving a peek into the back-end and some recent advancements in the area. At the end of the course, students should be able to appreciate the underlying concepts in compiler design, and be motivated to learn the art of analyzing and transforming programs for performance.

2. Learning outcomes:

After taking this course, the students will:

- be familiar with the front-end as well as back-end stages of compiler design
- understand the differences between constructing lexers/parsers by hand versus using automated generators
- get hands-on experience with generating intermediate representations, which in turn will let them appreciate the importance of designing simpler languages
- be able to appreciate the nuances of analyzing and transforming programs for performance
- get experience of working with relatively large programming environments, which will also inculcate a sense of good software design
- be motivated to explore different real-world applications that involve the concepts learnt as part of the course

3. Course modules:

- *Introduction and lexical analysis:* Introduction to language translators. Stages of compilation. Lexical analyzers: token specification and recognition. **[4 hours]**

- *Parsing:* Overview of context-free grammars. Parse trees and derivations. Left recursion and left factoring. Top-down and bottom-up parsing. **[8 hours]**

- *Semantic analysis:* Syntax-directed translation. Various intermediate representations. Intermediate code generation. Type checking. **[9 hours]**

- *Runtime environments:* Activation records. Heap management. Garbage collection. **[4 hours]**

- *Code generation and optimization:* Register allocation. Instruction selection and scheduling. Control-flow graphs. Data-flow analysis. Peephole optimizations. **[9 hours]**

- *Advanced topics:* Loop optimizations. Call-graph construction. Machine learning in compiler design. Just-in-time compilers. **[8 hours]**

4. Labs:

Learning the art of compiler design involves a nice mix of theory and practice. The course consists of at least four programming assignments from the following set (assignments will be rotated in every offering to contain plagiarism):

- Learning to use a lexer/parser generator such as Flex/Bison/JavaCC
- Intermediate code generation
- Type checking
- One forward/backward dataflow analysis
- Register allocation
- Assembly code generation

Towards this end, the students will learn various new software and program analysis tools. The labs will be conducted to train the students in using these tools, and preparing them for the main take-home assignment by asking them to solve related simpler tasks in two-hour sessions. The lab slots may also be used sometimes to conduct tutorials.

5. Textbooks:

1. Alfred V. Aho, Monica Lam, Ravi Sethi and Jeffrey D. Ullman. “Compilers: Principles, Techniques, and Tools”, Second Edition, Pearson Education, 2007.
2. Andrew W. Appel, Jens Palsberg. “Modern Compiler Implementation in Java”, Second Edition, Cambridge University Press, 2002.

4. Reference books:

1. Keith D. Cooper and Linda Torczon. "Engineering a Compiler", Second Edition, Morgan Kaufmann, 2011.
2. Y. N. Srikant and Priti Shankar. “The Compiler Design Handbook”, Second Edition, CRC Press, 2007.

5. Similarity Content declaration with existing courses: Nil

6. Justification of new course proposal if cumulative similarity content is >30%: N/A