UNIVERSITY OF GOTHENBURG

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DIT301 Compiler Construction, 7.5 credits

Compiler Construction, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by Department of Computer Science and Engineering on 2022-12-16 to be valid from 2024-01-15, spring semester of 2024.

Field of education: Technology 100%
Department: Department of Computer Science and Engineering

Position in the educational system

The course is a part of the Computer Science Master's programme and a single subject course at University of Gothenburg. The course is also offered as an elective course in the Computer Science Bachelor's Programme.

The course can be part of the following programmes: 1) Computer Science, Master's Programme (N2COS), 2) Applied Data Science Master's Programme (N2ADS), 3) Computer Science, Bachelor's Programme (N1COS), 4) Game Design & Technology Master's Programme (N2GDT) and 5) Software Engineering and Management Master's Programme (N2SOF)

Main field of studies	*Specialization*
Computer Science	A1F, Second cycle, has second-cycle course/s as entry requirements

Entry requirements

To be eligible to the course, the student should have successfully completed 120 credits of studies in Computer Science or equivalent. Specifically, the following course: DIT231 Programming Language Technology or equivalent.

Applicants must prove knowledge of English: English 6/English B or the equivalent level of an internationally recognized test, for example TOEFL, IELTS.

Learning outcomes

After completion of the course the student is expected to be able to:

1. Knowledge and understanding

•define lexical and syntactic structure of a programming language using regular expressions and grammars, respectively

•define type systems for typical programming languages

•recognize the characteristics of both stack machines and register machines and how common high level constructs in imperative languages are mapped to machine code

•recognize the general idea of data flow analysis for static analysis of properties of programs

•recognize the main issues of compiling functional and object-oriented languages

2. Skills and abilities

•use standard tools to generate a lexical analyzer and a parser for a compiler

•define suitable abstract syntax data types

•implement type checking, including simple type inference, with error reporting and decoration of syntax trees with type information

•design and implement compilation schemes that generate intermediate or assembly code from a source program

•implement simple data flow analysis

•implement simple memory management for heap-based data

3. Judgement and approach

•judge the difficulty of implementing various programming language features for different architectures

Knowledge and understanding

- On completion of the course, students will have both theoretical and practical insigth of how end-to-end compilers are given specifications, designed, implemented and tested.

Competence and skills

- know the basic principles of run-time organization, parameter passing, and memory management, and implement them as a part of the compiler;
- design and implement compilation schemes that generate intermediate or assembly code from a source program;
- design and implement extensions of a simple imperative language, and know the main issues of compiling object-oriented languages;
- use data flow analysis to implement register allocation and code optimizations.

Judgement and approach

- Students will be able to judge the maintainability, scalability and efficiency of different ways of designing, implementing and testing compilers.

Course content

The aim of the course is to develop an understanding of the whole process of compiler construction, starting from lexical analysis and finishing with machine code generation. As the course laboration, the students build a complete compiler for a simple imperative language.

Theory and implementation of compilers, with an emphasis on the backend phases of code generation runtime environments.

Sub-courses

1. **Assignments** *(Inlämningsuppgifter),* 7.5 credits
   Grading scale: Pass with distinction (5), Pass with credit (4), Pass (3) and Fail (U)

Form of teaching

The teaching consists of lectures and a project, as well as individual supervision in connection with the project. The project is the central part; it involves implementing a complete compiler for a small imperative programming language. The lectures provide knowledge that is needed in the laboration, as well as an overview of the theoretical foundations and possible extensions of the compiler.

Language of instruction: English

Assessment

Forms of examination: programming project and oral presentation. The oral presentation is pass or fail. Students who fail the oral presentation fail the course. Students who pass the oral presentation receive a grade that is directly determined by the extent of their submissions for their programming project. The programming project is done in groups, but the oral presentations are individual.

If a student, who has failed the same examined element on two occasions, wishes to change examiner before the next examination session, such a request is to be submitted to the department in writing and granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In the event that a course has ceased or undergone major changes, students are to be guaranteed at least three examination sessions (including the ordinary examination session) over a period of at least one year, though at most two years after the course has ceased/been changed. The same applies to work experience and VFU, although this is restricted to just one additional examination session.

Grades

The grading scale comprises: Pass with distinction (5), Pass with credit (4), Pass (3) and Fail (U).
Students who pass the oral presentation receive a grade that is directly determined by the extent of their submissions for their programming project.

Course evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire is used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

Additional information

The course is a joint course together with Chalmers and Gothenburg University.

Course literature to be announced the latest 8 weeks prior to the start of the course.

The course replaces the course DIT300, 7.5 credits. The course cannot be included in a degree which contains DIT300. Neither can the course be included in a degree which is based on another degree in which the course DIT300 is included.