UNIVERSITY OF GOTHENBURG

COMPUTER SCIENCE AND ENGINEERING

DIT300 Compiler Construction, 7.5 credits

Compiler Construction, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by The IT Faculty Board on 2006-11-17 and was last revised on 2017-06-16 by Department of Computer Science and Engineering to be valid from 2017-08-20, autumn semester of 2017.

Field of education: Science 100%
Department: 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) Game Design & Technology Master's Programme (N2GDT), 4) Software Engineering Master's Programme (N2SOM), 5) Computer Science, Bachelor´s Programme (N1COS) and 6) No translation available (NDATM)

Main field of studies	*Specialization*
Computer Science	AXX, Second cycle, in-depth level of the course cannot be classified
Computer Science-Algorithms and Logic	AXX, Second cycle, in-depth level of the course cannot be classified

Entry requirements

The requirement for the course is to have successfully completed two years of studies within the subject Computer Science or equivalent. Specifically, the course DIT230 Programming Languages or equivalent is required.

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

Course content

Theory and implementation of compilers, with an emphasis on the back-end phases of code generation and static analysis.

The project is the central part. The project task is to implement a complete compiler for a small imperative programming language, possibly with different backends for different flavours of virtual machine.

Form of teaching

Language of instruction: English

Assessment

The course is examined by a programming project and by an individual oral exam. The project work is carried out individually or in groups of normally two students. The programming project involves writing a compiler and consists of several submissions. A higher grade can be achieved by adding extra features to the compiler.

A student who has failed two examinations on the same material has the right to request a change of examiner. Such a request must be submitted to the Department in writing and shall be granted unless there are particular reasons not to do so.

In cases where a course has been discontinued or has undergone major changes, students must be guaranteed at least three examination opportunities (including the regular opportunity) based on the previous content of the course for a period of at least one year.

Grades

The grading scale comprises: Pass with Distinction (VG), Pass (G) and Fail (U).
To pass the course with the grade G, the student must obtain the grade G (at least) for both the project and the oral exam. To pass the course with the grade VG, the student must obtain the grade VG on the project and obtain the grade G on the oral exam.

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.