Blekinge Institute of Technology
Department of Computer Science

Revision: 2
Reg.no: BTH-4.1.14-0194-2023

COURSE SYLLABUS

Kompilator- och översättarteknik

Compiler Design and Translation Technique

6 credits (6 högskolepoäng)

Course code: DV1655
Main field of study: Computer Science, Technology
Disciplinary domain: Technology
Education level: First cycle
Specialization: G1F - First cycle, has less than 60 credits in first cycle course/s as entry requirements

Language of instruction: Swedish but teaching in English may occur.
Applies from: 2023-08-28
Approved: 2023-01-30

1. Decision

This course is established by Dean 2021-04-16. The course syllabus is approved by Head of Department of Computer Science 2023-01-30 and applies from 2023-08-28.

2. Entry requirements

Admission to the course requires 18 credits completed in programming , including computer structures and algorithms.

3. Objective and content

3.1 Objective

Most of the programs that we write are written using a high-level programming language, whereas the hardware understands only low-level code (binary instructions). Compilers are software program that can translate a program written in one language (usually a high-level language, that is called a source language) into an equivalent program in another language (usually a lower-level language such as assembly, known as target language). The assembler then is responsible to generate the binary code from an assembly code. It is common for compilers to be used for translation from one high-level (source) language to another high-level (target) language, such compilers are called source-to-source compilers. Understanding how different compilation phases are designed and implemented (including lexing, parsing, type-checking, code generation, interpretation) and the application of different translation techniques is essential for the advanced programmer that might be willing to perform any extensions to the existing programming languages or optimalizations to existing compilers. Additionally, this course guides students through the steps needed to design new domain-specific languages and develop their corresponding compilers. Furthermore, the techniques used in translation and compiling are also applicable in many other areas, where the knowledge of this subject is particularly useful.

3.2 Content

The course comprises of the following:
• Basic theory for formal languages, language grammars, and translation techniques
• Principles for lexical, syntactic (top-down and bottom-up parsing), and semantic analysis (type-checking)
• Introduction to various tools for designing and implementing different compiler phases including lexical analyzer and syntax analyzer.
• Design and implementation of graph-based intermediate representation and graph traversal techniques.
• Principles for designing code generation and interpretation
• Design and implement all phases of the compiler, including the lexer, parser, type-checker, intermediate representation, code generation, and interpretation.

4. Learning outcomes

The following learning outcomes are examined in the course:

4.1 Knowledge and understanding

On completion of the course, the student will be able to:
• Understand, describe, and employ basic theory for formal languages and language expressions, language grammars, and translation techniques
• Understand and employ the principles for lexical, syntactic, and semantic analysis
• Understand and employ principles for intermediate representation, code generation, and interpretation

4.2 Competence and skills

On completion of the course, the student will be able to:

- Design and develop different phases of the compiler, including
- Scanner (Lexical Analysis),
- Parser (Syntax Analysis),
- Type checker (Semantic Analysis),
- Intermediate representation,
- Code generator, and Interpreter.
- Design and develop test suite that ensures language coverage.

4.3 Judgement and approach

On completion of the course, the student will be able to:

- Assess the impact of choices in representation and algorithms on the implementation of the compiler.

5. Learning activities

The course uses a mixture of lectures, supervised lab sessions, and project work. The lectures introduce the theoretical material, the labs are used for tutorial support on the project work. The project is split in three parts to allow extensive feedback early in the process.

6. Assessment and grading

Modes of examinations of the course

Code	Module	Credits	Grade
2206	Project assignment (part 1)	1.5 credits	AF
2215	Project assignment (part 2)	2 credits	AF
2225	Project assignment (part 3)	2.5 credits	AF

The course will be graded A Excellent, B Very good, C Good, D Satisfactory, E Sufficient, FX Fail, supplementation required, F Fail.

The information before a course occasion states the assessment criteria and make explicit in which modes of examination that the learning outcomes are assessed.

An examiner can, after consulting the Disability Advisor at BTH, decide on a customized examination form for a student with a long-term disability to be provided with an examination equivalent to one given to a student who is not disabled.

7. Course evaluation

The course evaluation should be carried out in line with BTH:s course evaluation template and process.

8. Restrictions regarding degree

The course can form part of a degree but not together with another course the content of which completely or partly corresponds with the contents of this course.

9. Course literature and other materials of instruction

Main course book: Lam, M., Sethi, R., Ullman, J. D., & Aho, A. (2006). Compilers: Principles, techniques, and tools. Pearson Education.

Other literature: Levine, John. Flex & Bison: Text Processing Tools. " O'Reilly Media, Inc.", 2009.

10. Additional information

This course replaces the course DV1585