Course Memo

Introduction

Welcome to the course in Compilers and Execution Environments!

In this course, you will learn how a compiler works. You will learn about compiler theory, for instance regular expressions, parsing theory, optimization algorithms, code generation strategies, garbage collection, and more. Moreover, you will gain significant practical experience in designing and implementing a complete compiler from scratch.

Learning Outcomes

The complete syllabus **can be found here** **(https://www.kth.se/student/kurser/kurs/ID2202?l=en)**. In summary, the learning outcomes are as follows.

After passing the course, the student should be able to

- use methods for lexical, syntactic and semantic analysis
- use methods for generation of machine code
- use methods for optimizing programs
- give an account of common components in execution environments

in order to

- obtain an understanding of how a programming language is implemented as well as for the general theories that are used and how these can be applied.

Modules

The course is divided into three modules. Each module is two weeks long and covers a specific area:

- **Module 1 (weeks 45 and 46): Lexical analysis, syntax analysis, and semantic analysis.** The goal of module 1 is to comprehend, design, and implement the front end of a compiler, including lexing, parsing, and interpretation.
- **Module 2 (weeks 47 and 48): Code generation and runtime environments.** The goal of module 2 is to compile complete unoptimized Cigrid programs to x86 machine code.
- **Module 3 (weeks 49 and 50): Program analysis and optimizations.** The goal of module 3 is to perform various optimizations of your compiler, including intermediate code optimization and register allocation.

The first week (week 44) will be a "getting started week", where the student gets familiar with the environments, gets an overview of compilers, and takes a crash course in typed functional programming.

Learning Activities and Examination

Each module consists of the following *learning activities* (marked with LA) and formative *examination tasks* (marked with ET):

1. **Lectures (LA).** Each module starts with a number of lectures. All lectures will be given physically at KTH Kista, but they will also be live streamed using Zoom. The lectures will *not* be recorded.
2. **Videos (LA).** There will be short videos available that explain certain specific areas of the course. The videos are not lecture recordings, but short videos dedicated to specific topics. This includes both theoretical concepts (algorithms, formalisms etc.) and practical aspects (programming etc.). Additional videos will be added on request.
3. **Hacking sessions (LA).** Each week, there are two in-person hacking sessions (please see the schedule). During these in-person sessions (not online), you will be able to chat with fellow students and to pose questions to teaching assistants.
4. **Online office hours (LA).** Each week, we will have online office hour possibilities, where students can contact teaching assistants, and get help over Zoom,
5. **Assignments (LA/ET)**. During the two weeks modules, you will work independently (not in groups) on different design and implementation tasks. You will submit your tasks continuously to the course Git system. The deadline for submitting all tasks is on Friday mornings, 9am, the last week of each module.
6. **Peer reviewing (LA/ET).** During the beginning of the week after a module, you will be given the task to peer review the code and the solutions of another fellow student. As part of this peer reviewing, you provide comments and written questions about the other student's work.
7. **Seminars (LA/ET)**. The week after each module, after the peer reviewing task, you will participate in a seminar, where you will present your tasks and answer questions from a teaching assistant and the peer reviewing student. There will be two to three students presenting during each seminar. The seminars will take place over Zoom and there will be individual bookings of these time slots.

There will be extra retake opportunities after Christmas.

Grading

According to the syllabus, there is one course part that is reported into LADOK.

- DAT1 - Computer based exam, 7.5 credits, Grading scale: A, B, C, D, E, FX, F

All examination will be done using: (i) the assignments (ii) peer reviewing, and (iii) seminars. Hence, there is no written exam at the end of the course.

Each of the three modules are graded with one of the following grades:

- *Satisfactory (S): The student has fulfilled the following for the module:*
  - *All assignments marked as (S) have been submitted to the Git correction system and all automatic tests pass.*
  - *A teaching assistant has corrected tasks for (S) that cannot be automatically corrected and the results are pass.*
  - *The student has sent in a peer reviewing assessment that has been given the grade pass by a teaching assistant (if applicable).*
  - *The student has participated in a complete seminar, where they have acted as an opponent and asked questions based on the peer review.*
  - *The student has presented their solution on a seminar, defended the implementation, and explained the solution in a satisfactory manner. The teaching assistant for the seminar has given the student pass on the oral presentation.*
- *Good (G): The student has fulfilled all criteria for (S) as well as submitted solutions to all assignments marked as (G). For all the tasks at level (G), the student has received a pass by either the automatic grading system or manually by a teaching assistant, or both.*
- *Very Good (VG): The student fulfilled all criteria for (S) and (G), as well as submitted solutions to all assignments marked as (VG). For all the tasks at level (VG), the student has received a pass by either the automatic grading system or manually by a teaching assistant, or both.*
- *Failed (F): The criteria for satisfactory (S) have not been fulfilled.*

The final grade (A, B, C, D, E, FX, F) is based on the overall grades of the three modules.

If a student submits solutions that are given pass on time *for all three modules* (meets the deadline of the last Friday at 9am in each module), then the student receives a *deadline bonus.* The final grade is computed as follows:

- **Grade A** if received VG on all three modules. With a deadline bonus, only two VGs and one G is needed.
- **Grade B** if received two VGs and one G. With a deadline bonus, only one VG and two Gs are needed.
- **Grade C** if received one VG and two G. With a deadline bonus, only three Gs are needed.
- **Grade D** if received at least two Gs and one S. With a deadline bonus, only at least one G and two Ss are needed.
- **Grade E** if received at least three S.
- **Grade FX** if F is received on at most one module. An FX cannot result in deadline bonus.
- **Grade F** otherwise.

After Christmas, the students have the possibility to complement their solutions for all three modules, and to receive higher grades on the different modules. Note that these submissions are optional. If complementary work is done to get higher grades, the deadline bonus is not valid anymore.

Example 1: Suppose a student submitted all assignments on time, and passed seminars and peer reviewing on all modules. Suppose also that they received grades G S S on the three modules. Hence, because of the deadline bonus, the student gets the final grade D.

Example 2: Suppose a student submitted everything on time and hence has the deadline bonus. The grades for the three modules were VG G S. Hence, they get the final grade D. However, the student is not satisfied with this grade and manages to upgrade the S to a G after week 50, resulting in VG G G. The deadline bonus is not valid anymore (after complementary work). As a consequence, the student gets grade C. To get a B during the complementary work, one of the Gs would have to be further upgraded to a VG (i.e., resulting in VG G VG or VG VG G). Note that if the student would have received VG G G on time and received the deadline bonus, the final grade would have been a B.

The grade will be assigned according to what has been reported into Canvas by the end of January 17. If a student receives grade FX, they will have to submit a new solution at latest by January 10, 9am. The peer reviewing and the seminar for the FX submission will take place in January on an individual basis.

There will be two retake occasions during the spring. It is then possible to complement your solutions and assignments for all three modules, to achieve a higher grade or to pass the course. The submission deadlines are January 10 (9am) and March 1 (9am). The peer reviewing and seminars will take place the weeks following the deadlines, and will be assigned on an individual basis. If you have not passed the course after the March deadline, you will have to retake the course the next fall.

Note also that the teacher may ask for individual examination and presentation of exercises, if there are unclarities in submitted solutions, unclarities during seminar presentations, or indications of cheating attempts.

Teachers

Examiner and Course Responsible

- **David Broman**   **(https://people.kth.se/~dbro/)**

Lectures

- **David Broman (mailto:dbro@kth.se)**

Hacking Sessions and Seminars

- **Linnea Ingmar**   **(https://www.kth.se/profile/lingmar)**
- **Viktor Palmkvist**   **(https://www.kth.se/profile/vipa)**
- **Lars Hummelgren**   **(https://www.kth.se/profile/larshum)**
- **Klas Segeljakt**   **(https://www.kth.se/profile/klasseg)**

Literature

The following is the recommended course book:

- Keith D. Cooper & Linda Torczon. **Engineering a Compiler.** Second Edition, Morgan Kaufmann, 2012.

The book is available for free online, using your KTH account. **See this link** **(https://www-sciencedirect-com.focus.lib.kth.se/book/9780120884780/engineering-a-compiler)** .

Disability

If you have a disability, you may receive **support from Funka (https://www.kth.se/en/student/studentliv/funktionsnedsattning)** .

Funka also recommends that you inform the teacher (in this case **David Broman (mailto:dbro@kth.se)**) regarding any needs you may have. Funka does not automatically inform teachers in courses.

Code of Honor

Please make sure that you are aware of and that you understand the **KTH EECS Code of Honor. (https://www.kth.se/en/eecs/utbildning/hederskodex)**

Policy against Plagiarism

Note that all forms of cheating and plagiarism will be reported. Please see KTH's **policy for handling plagiarism (https://canvas.kth.se/courses/17831/files/2446097/download)** .

- **Programming assignments.** You may discuss problems and solutions with anyone. You are not allowed to copy ANY code and submit it as part of your solution. You must create your own solutions from scratch.
- **Individual theory assignments.** You are not allowed to discuss or publish any of the assignments. You are not allowed to get help from anyone or find the solution on Internet.
- **Written text assignments.** You are not allowed to copy, cut, or paste any text into your report that is not produced by you. The only exception is if you quote text properly, and give a citation to the original source.

Use of GitHub

Since all students are performing the same exercises in the course, you should not make any of the solutions to the exercises and your compiler solution publicly available during or after the course. This includes making it public on Github, on websites etc. If you later on are applying for a job, it is of course OK to share it with a potential employer or to share it to other people privately (after the course, as long as it is not to future students). The key thing is to not make it publicly available.

Course Analyses and Feedback

The following course analyses are available:

- **ID2202, Fall 2020, Course Analysis (https://canvas.kth.se/courses/28707/files/4767414?wrap=1)**
  ⤓ **(https://canvas.kth.se/courses/28707/files/4767414/download?download_frd=1)**
- **ID2202, Fall 2021, Course Analysis (https://canvas.kth.se/courses/28707/files/5310425?wrap=1)**
  ⤓ **(https://canvas.kth.se/courses/28707/files/5310425/download?download_frd=1)**

We think it is very important to continuously improve our course, especially now when it is the first time that the course is given. Therefore, we really appreciate all constructive feedback that we can get. In this course, we collect and analyze feedback in the following ways:

Personal Feedback

We really appreciate if you give us direct and constructive feedback. Please, send me (David) an email (**dbro@kth.se (mailto:dbro@kth.se)**) with your feedback or let's meet up for a cup of coffee!

Battery Evaluation

At the break of one of the lectures, students write down (anonymously) pros and cons about the course. We call this a battery evaluation since the physical answer papers looks like batteries (+ and - lists on each part of the paper). The feedback is then collected, summarized, and analyzed by the course responsible. The analysis is presented at one of the following lectures. The benefit of this kind of early and direct feedback during the course is that actions can be taken for you, in this course edition.

Web-based Course Evaluation

We perform standard web-based course evaluations after the course. This feedback is used to improve the next year's course edition. The course analysis is based on this course evaluation.

Course Committee (Kursnämnd)

A course committee is a group of students that meet up with me at the middle and at the end of the course. The purpose is to give concrete oral feedback in an informal way. If you are interested to be part of the course committee, please let me know at the first lecture.

FID3006

FID3006 is the doctoral level version of the compiler course. It follows the same structure as ID2202, but has an additional project assignment, as well as specific requirements for passing the three course modules. More specifically, the differences are as follows:

- **Project assignment.** In addition to the course modules (as presented in the course memo for ID2202), FID3006 students need to do a small project assignments. In this project assignment, each student should individually do a small project related to compilers and programming languages. The exact topic is quite flexible, and it can preferably be related to the doctoral student's own research. It must be some sort of investigation, provide clear research questions or

problem statements, relate to state of the art, include a discussion, and explain the results. It can preferably include an implementation and experiments, but can also be of theoretical nature. Please talk to the examiner and discuss your topic before you start. As a result of the project work, you need to submit a term paper, 3-6 pages, excluding references, using the ACM **acmart format** **(https://www.sigplan.org/Resources/Author/)**, configured using sub format *sigplan* (two-column format). You submit the term paper by sending it attached in an email to **dbro@kth.se (mailto:dbro@kth.se)**. We will then decide on a time slot where you give a 20min oral presentation of your work. When both the term paper and the presentation are given a pass grade by the examiner, the project assignment has been passed.

- **Module grades for course modules 1, 2, and 3.** In addition to passing the project assignment, FID3006 students need to get grade VG (Very Good) on all three course modules. There is no deadline bonus, but we recommend that FID3006 students submit their work according to the schedule of the ID2202 course.

If nothing else is stated, to pass the course, both the project assignment and the course modules must have been finished before the next year's course round starts.