Course Analysis

Compilers and Execution Environments (ID2202)

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science (EECS)

Examiner: David Broman
Course responsible: David Broman
Course analysis carried out by: David Broman (dbro@kth.se)
Course edition: Fall 2022 (Period 2)
Level and credits: Master's level course, 7.5hp
Number of first registered students (according to Ladok): 46

1. Course Summary and Design

This section gives a brief overview of the main parts of the course design, including course objectives, learning activities, and examinations. The course content and course objectives are based on the official course syllabus, whereas the explanation of learning activities and examinations are parts of the course memo. Below, we give a summary of these parts. We also give a brief overview of the main changes compared to previous course editions.

Course Content

The course covers technologies for implementation of programming languages by means of compilers, both for real and virtual execution environments, technologies to read, understand, translate, improve as well as execute programs:

- To read programs: lexical analysis and syntax analysis. Finite state machines, regular expression context-free grammars, LL and LR-parsing.
- To understand programs: semantic analysis, type checking.
- To translate programs: machines and instructions.
- Intermediary code, choice of instructions, conventions for procedure calls.
- To improve programs: machine-independent optimizations; computer-oriented optimizations (register allocation, scheduling of instructions).
- To execute programs: virtual execution environments and runtime systems. Memory management, garbage collection, to load and link programs, just-in-time compilation.

Course Objectives

After passing the course, the student shall 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.

For higher grades, the student should design more complex components of a compiler.

Learning Activities

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

- *Module 1: 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: Code Generation and Runtime Environments.*
- The goal of module 2 is to compile complete unoptimized Cigrid programs to x86 machine code. Cigrid is a small language designed in this course for learning purposes.
- *Module 3: Program analysis and optimizations.*
  The goal of module 3 is to perform various optimizations of a compiler, including intermediate code optimization and register allocation.

Besides the three main modules, there is also a Module 0, consisting of a crash course in functional programming.

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

- *Lectures (LA).* Each module starts with three lectures. All lectures are given physically at KTH Kista, but are also streamed using Zoom.
- *Hacking sessions (LA).* Each week, there are two in-person hacking sessions at KTH Kista. During these sessions, the student is able to chat with fellow students, and to pose questions to teaching assistants.
- *Assignments (LA/ET).* During the 2-week module, the student works independently (not in groups) on different design and implementation tasks. It is possible to submit tasks continuously to the Git system developed for the course.
- *Peer reviewing (LA/ET).* At the beginning of the week, after a module, the student is given the task to peer review the code and the solutions of another fellow student. As part of this peer reviewing, the student provides comments and written questions about the other student's work.
- *Seminars (LA/ET).* The week after each module, after the peer-reviewing task, the students participate in a seminar, where they present tasks and answer questions from a teaching assistant and the peer reviewing student. There are 2-3 students presenting during each seminar. The seminars take place over Zoom.

Examination

The course has one course part (7.5hp) that is reported into Ladok when the course is completed. The grading scale is: A, B, C, D, E, FX, F.

There is no written exam (summative assessment) at the end of this course. Instead, the examination is *formative*, where the students both show their level of acquired knowledge and skills, and where they are given a possibility for learning.

For each of the three modules in the course (see above), the student receives one of the following grades: Satisfactory (S), Good (G), Very Good (VG), or Failed (F). The final course grade is then in the end computed based on the three grades received for the three modules.

There is also a so-called *deadline bonus,* which makes it easier to receive a higher grade if the module is finished on time, on the first deadline. This gives the incentive for the students to focus on the course, learn during the scheduled working weeks, and to communicate with teaching assistants during these scheduled slots. The deadline bonus only helps to get higher grades; not to pass the course itself.

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

- *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 is given the grade pass by a teaching assistant.
    - 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 his/her 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.

Note that the tasks include both programming tasks, where the students' solutions are automatically corrected (using a Git-based unit testing system developed for this course), as well as manual correction by teaching assistants (mainly focusing on theoretical exercises).

Changes Since the Last Edition

This is the third course edition for ID2202 that has been taught by the course responsible and examiner David Broman. Since the last year, there have been certain adjustments in the course, including:

- **OCaml crash course.** A new addition to last year's course round was that we have a one-week crash course on OCaml, to get everyone up to speed from the start. The focus is on typed functional programming and its usefulness when developing compilers. A change this year was that we made module 0 compulsory, which resulted in the students starting earlier with the implementation this year.
- **Workload.** To lower the workload, we removed some of the exercises in Module 1 and rearranged them to lower the workload, especially in the first parts of the course.

2. Course Evaluation Process

In this section, we briefly outline how we gathered course feedback from the students and how we adjusted, reacted, and used the received feedback.

Evaluation Activities

During the course, the following evaluation activities were planned and/or took place.

- All students were strongly encouraged to send emails to the examiner with feedback. In particular, in some instances, we asked explicit questions to the students that they later on answered. By having direct contact with most students who took the course actively, we were able to adapt and make the course better during the course.
- After the course, we sent out the usual web-based course evaluation form. As expected, we received a pretty low answer rate: 6 answers out of 46 registered students.

- We also had informal feedback meetings after lectures. I also received many emails with very constructive feedback and comments.

All students had the possibility to give feedback, either anonymously via the web-based course evaluation system, or directly via email or Zoom. In the beginning of the course, we asked to get volunteers for the course committee.

Meetings with Students

This year, we have not yet had a course committee meeting, but an extra meeting is planned for the fall 2023, before the next course edition. For the 2023 course edition, we will combine this course with the compiler course taught at the main campus. For this reason, we will have an extra course committee meeting with students from both courses to get feedback on the new development.

During the course, I had quite extensive communication with several students over email. Feedback from students has played a significant role when shaping both this version of the course, and as input for improvements for next year.

Gender, Diversity, and Disability Aspects

There were significantly more male than female students who took this course. It is, in general, an elective course that is available in many programs. Many students gave feedback during the course, both orally and via email. In the course memo, we have provided information about Funka students and that the students can contact the examiner for information about what kind of support we can provide regarding disability.

3. Outcomes

This section summarizes the students' results during this course round, the expected and experienced student workload, and a general summary of student responses. The responses reflect personal student feedback and anonymous feedback sent via the web-based course evaluation.

Student Results

In total, there were 46 students registered according to Ladok. 42 students submitted a solution to module 0 (auto-corrected), and out of these, 28 students submitted at least one complete module that included getting a pass on a seminar. In total, 22 students received a final course grade within this course round.

Besides the ID2202 students, one more Ph.D. student successfully finished the Ph.D. course version FID3006. The grade distribution (excluding FID3006) of the passed students was as follows (rounded numbers):

A: 9% (2 students)
B: 5% (1 student)
C: 5% (1 student)
D: 23% (5 students)
E: 59% (13 students)

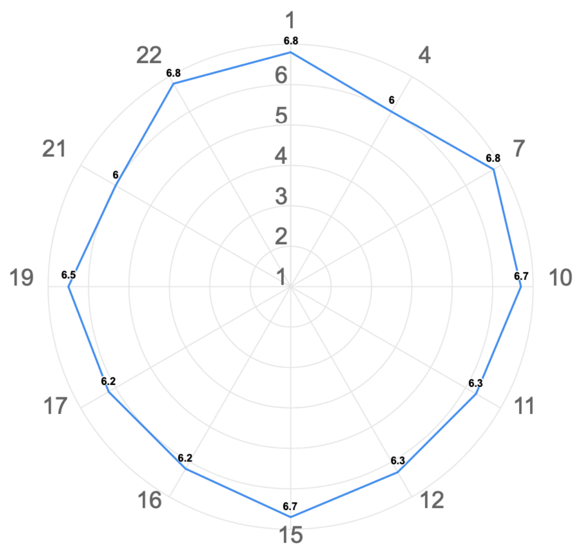
This year, the grades were slightly higher than the previous year.

Student Workload

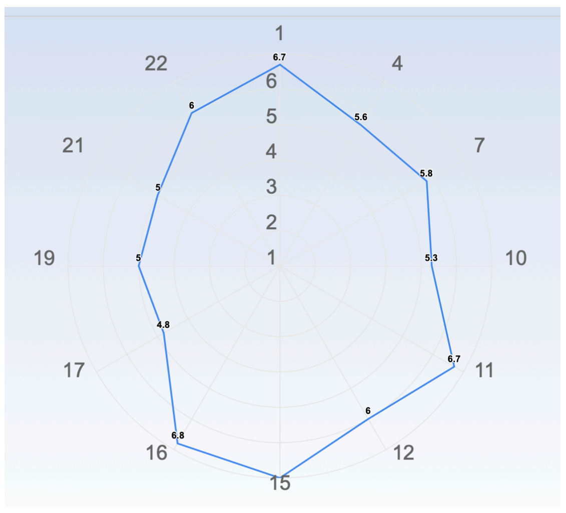
In the web-based course evaluation, the students answered that they worked between 12h per week to 38h per week. The students generally said that the workload was high, especially module 1. Note that this course is 7.5hp and studied during one period, that is, half-time studies. Some students thought it was too high, whereas other students thought it was as expected.

Student Responses

The web-based student responses resulted in the following Learning Experience Questionnaire (LEQ)-diagram for years 2021 and 2022:



Year 2021



Year 2022

The 2022 edition has changed slightly; some areas are lower than in previous years. Note, however, that this is a very small sample of the students. Only six students answered the LEQ questionnaire.

Below, we have summarized the main feedback from the student responses:

- Item 19 "The course activities enabled me to learn in different ways" is a bit lower than last year. The raw data show that all responding students are positive, except one negative student, who thought the structure was repetitive. On the other hand, most other students seem very positive.
- Item 17 "My background knowledge was sufficient to follow the course" is a bit lower this year. The main thing that some students say is that they did not know functional programming before. In general, several students do not have a computer science background, in which case the course can be challenging.
- Some students said that the structure of the course was excellent, enabling students to work at their own pace.
- The main comment was that the workload was high, especially at the beginning of the course. Some students suggested fewer and easier tasks.
- Many students answered that they liked the assignments and the possibility of creating a compiler of their own.

- Some students thought that we should have even more details in the lectures.
- Many students were pleased with the hacking sessions.
- Several students appreciated the videos.
- In general, it seems like most students were satisfied with the course.

4. Analysis and Planned Course Development

From the outcome given in the previous section, we can conclude that most students enjoy the course, but they also find it quite challenging. In this section, we discuss and analyze some of the key student feedback. We propose changes and improvements for the next year's course development for each of the different areas.

Background knowledge and workload
This year, we got similar responses as last year, that several students found the course challenging and required a lot of effort. Given the struggle some students found in the early programming exercises, it is still clear that many students do not have the expected computer science background that is actually required for the course.

We introduced both a crash course in OCaml and reduced the workload on some of the exercises in Module 1. Also, since the next year's edition will combine this course with the one taught at the main campus, we will make some changes and updates.

For next year, we plan to do the following main updates:

- **Crash course.** We will keep the crash course that we introduced this year. We will, however, offer the crash course both for OCaml and Scala. Students can then choose which language they will use. They can also use Java, C++, and more, but there will be no specific crash courses.
- **Videos.** We will continue to add more tutorial videos since this has been very appreciated as a complement to the lectures.
- **New content.** New content. We will extend the course with more material on JVM as a target language. There will also be slight changes in the lecture material. However, since the feedback has generally been very good over the years regarding lectures, seminars, exercises, and examinations, we will not make major changes.
- **Workload.** We will consider further reducing the workload, especially in module 1.

Conclusions

To conclude, the course has been well-received over the years. The main challenge for next year is to reduce the workload slightly while introducing some new material from the other course. We also note that the final grades achieved by the students were slightly higher this year. In summary, we are delighted that the course is appreciated, and we hope to make it even better next year.