Course Analysis

Compilers and Execution Environments (ID2202)

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

Examiner: David Broman
Course coordinators: Philipp Haller and David Broman
Course analysis carried out by: Philipp Haller (phaller@kth.se)
Course edition: Fall 2023 (Period 2)
Level and credits: Master's level course, 7.5hp
Number of first registered students (according to Ladok): 52

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 (either in Kista or on Campus), 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 first course edition for ID2202 that has been taught collaboratively by David Broman and Philipp Haller who both served as course coordinators. David Broman served as the examiner. Since the last year, the following changes were implemented:

- **Scala crash course.** Since this year, Scala is one of the supported languages for implementing the course project. In order to provide an introduction to typed functional programming for students who select Scala but who don't know OCaml, a Scala crash course was developed, with contents similar to the existing OCaml crash course.
- **Video series on compiler engineering using Scala.** The videos cover (a) constructing a bottom-up parser using the Scala-Bison bottom-up parser generator, and (b) constructing a recursive-descent parser manually.
- **New videos on Liveness Analysis and Register Allocation.** These videos cover material for module 3.
- **More material on JVM as a target language.** This is a result of merging material from the previous course DD2488 (designed and taught by Philipp Haller) into ID2202.

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 (LEQ). As expected, we received a pretty low answer rate: 13 answers out of 59 registered students (22%).
- We also had informal feedback meetings after lectures. We also received many emails with very constructive feedback and comments.
- We organized a course evaluation meeting on Wednesday 20 March, 2024, where two students participated, as well as four TAs (see below).

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

A course evaluation meeting took place on Wednesday 20 March, 2024, 17:00-18:00, via Zoom. The participants included the two course coordinators, David Broman (who is also examiner) and Philipp Haller, the student course representative and another student representative, as well as four TAs (all PhD students). During the course meeting, the results of the course evaluation (LEQ) were discussed, as well as feedback from the student representatives and the TAs.

During the course, we also had extensive communication with several students over email.

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 52 students first registered according to Ladok. This number is 13% higher than last year. An increase was expected, since this is the first course edition for ID2202 where it was merged with the previous course DD2488. In total, 37 students received a final course grade within this course round, an increase of 68% compared to the previous year. The percentage of registered students who received a final course grade was 63% (rounded), which is higher than last year when that percentage was 48% (22/46 students).

Besides the ID2202 students, 3 more Ph.D. students registered for the Ph.D. course version FID3006 according to Ladok. (None of the students has completed FID3006, yet.) The grade distribution (excluding FID3006) of the passed students was as follows (rounded numbers):

A: 24% (9 students)
B: 5% (2 students)
C: 5% (2 students)
D: 27% (10 students)
E: 38% (14 students)

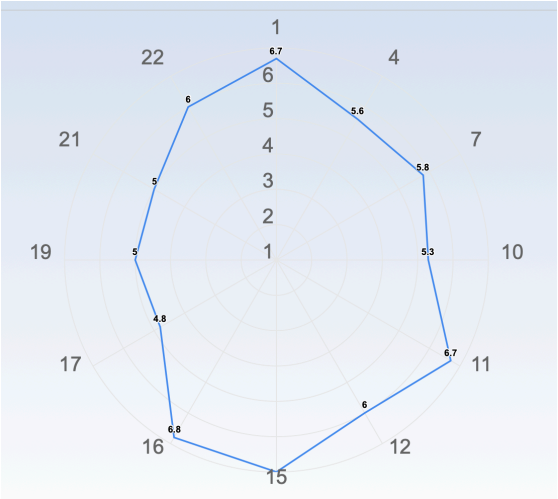
This year, the grades were slightly higher than the previous year. This year 24% of the passed students achieved grade A, whereas last year only 9% of the passed students received grade A.

Student Workload

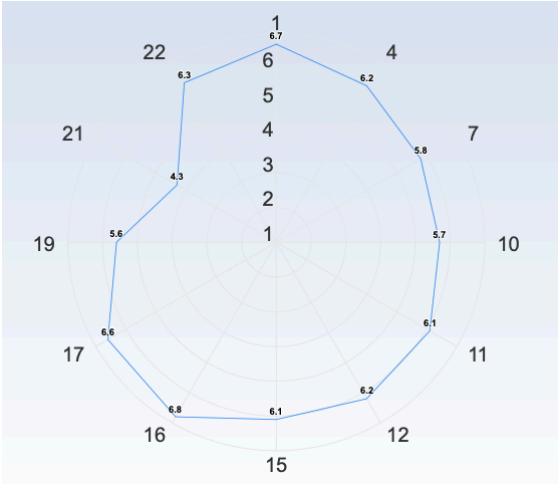
In the web-based course evaluation, the students answered that they worked between 4h per week to 32h per week. The students generally said that the workload was high, especially module 1 and 3. Note that this course is 7.5hp and studied during one period, that is, half-time studies. Some students thought 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 2022 and 2023:



Year 2022



Year 2023

As expected, there are small changes in the 2023 edition. Some areas are lower than in the previous year and some areas are higher. Note, however, that this is a very small sample of the students. Only 13 students answered the LEQ questionnaire.

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

- Item 15 "I could practice and receive feedback without being graded" is a bit lower than last year. The raw data show that all responding students are positive, except one negative student. Most students seem very positive with one student pointing out that the feedback from the cloud-based autograder was very helpful.
- Item 21 "I was able to learn by collaborating and discussing with others" is also a bit lower than last year. 5 students gave a negative score. One student answered that the peer reviews were less helpful than they had hoped, since the reviewed solutions used different languages. The same student also suggested solving the assignments in pairs to enable learning from each other. On the other hand, 61% of the respondents were neutral or positive, suggesting that most students did not perceive this area as negative in any way.
- Item 17 "My background knowledge was sufficient to follow the course" is significantly higher than last year. There were no negative scores. One student thought that the courses in the first 3 years of the 5-year computer science program (CDATE) were a suitable preparation. In total, 12 of the passed students were enrolled in the CDATE program. This higher participation from CDATE students is due to the merger with DD2488, which is new.

- A student said that the course was well managed and well structured, and the module divisions with regular hand-ins was appreciated.
- Many students answered that they liked the assignments and the possibility of creating their own compiler.
- Several students wrote that the lectures were very good ("very detailed and closely connected to reality") and that "the topics were introduced in a clear and logical way".
- Students also appreciated the grading levels, enabling them to choose how much time to devote to the course.
- Positively mentioned were also the autograder and support for a multitude of programming languages.
- Some students mentioned suggestions for improvement, but none of the suggestions was mentioned more than once.
- In general, it seems 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 some students 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

One difference that can be observed between this year and last year is that for item 17 "My background knowledge was sufficient to follow the course" there was not a single negative score. This means that this year most students had the expected computer science background required for the course. We also provided two complete crash courses for OCaml and Scala which might have helped with the early programming assignments. This year, we also made more videos available, (a) for later parts of the course, and (b) for students selecting Scala to solve the assignments. Regarding the workload, there were still some students who answered that the workload was too high, especially for modules 1 and 3.

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

- **Videos.** We will continue to add more tutorial videos since this has been very appreciated as a complement to the lectures.
- **Workload.** We will consider further reducing the workload, especially in modules 1 and 3. For example, the lexical and syntactic specification of Cigrid could be simplified to make creating the lexer and parser less time-consuming.
- **Assignments.** In the course evaluation meeting, the student representatives mentioned that later parts of the course, such as optimizations and garbage collection, could perhaps be connected to the labs/the compiler project.

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 keeping the time and space for the crash courses. We also note that the final grades achieved by the students were slightly higher this year. Partially, this might be due to the fact that more students from the CDATE program participated due to the merger with the previous course DD2488. In summary, we are delighted that the course is appreciated, and we will try to make it even better for the next iteration of the course.