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 2021 (Period 2)
Level and credits: Master's level course, 7.5hp
Number of first registered students (according to Ladok): 36

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 examination. The course content and course objectives are based on the official course syllabus, whereas the explanation of learning activities and examination 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.

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).* During 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.

Besides the three main modules, this year also introduced a module 0, consisting of a crash course in functional programming in OCaml.

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) in 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 second-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:

- **Updated lectures**. The overall lecture structure is the same, but several of the lectures have been extended and improved. In particular, module 3 has been extended with in-depth presentations of dominators, loop optimization, and static single assignment (SSA). The last lecture also covered a research outlook.
- **Allow many languages, recommend one language.** Within the TA team and with the students from the previous year, we concluded that it is still an excellent idea to choose the implementation language. Still, we should have a default recommended language. As a consequence, we decided to use OCaml as the default functional language because it is relatively easy to learn (the functional subset), and it has the benefits of a typed functional language for compiler development.
- **OCaml crash course.** A new addition to this year's course round is 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 how it is useful when developing compilers.
- **Hybrid lectures.** All lectures were given in person, as well as broadcasted over Zoom. This year we decided not to have the actual lectures recorded.
- **Complementary videos.** Besides the lectures, additional videos were added that describe specific topics in more depth.
- **Compiler generator.** A major change in the assignment for module 1 was that all students used parser generators for creating the Cigrid parser, compared to manually developing their own recursive-descent parsers, as developed last year. The aim was to make the task a bit smaller and require less effort. The calc assignment still uses recursive-descent parsing, so that the students still learn fundamental top-down parsing.

2. Course Evaluation Process

In this section, we briefly outline the process on how we gathered course feedback from the students, and how we adjusted, reacted, and made use of 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.
- We performed a *battery evaluation* during one of the lectures, where students gave feedback about the course. We received in total 13 anonymous answers.
- 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 53 registered students. However, note that out of the 53 students, only 23 students participated in at least one seminar/peer review assignment, which means that the number of students who took the course actively was much lower than the number of registered students. This is a pretty common situation in many courses.
- During the course, we had one course committee meeting. The meeting was held in person and only one student signed up and participated.
- 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

There was one course committee meeting in the middle of the course. In this meeting, the student provided good ideas about what can be improved for next year. However, 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 student, and that the students can contact the examiner for information about what kind of support we can provide regarding disability.

3. Outcomes

In this section, we summarize 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 sent via email, course evaluation meetings, and anonymous feedback sent via the web-based course evaluation.

Student Results

In total, there were 53 students signed up for the course according to the Canvas system. However, the number of first-time registered students according to Ladok is 36. Out of these students, 23 students submitted and participated in at least one seminar. Out of these, 18 students received a final course grade within this course round (excluding PhD students who took the graduate level of the course, counting students finished by Feb 2022). 15 students got the deadline bonus (submitting on time). The grade distribution of the passed students was as follows (rounded numbers):

A: 0%
B: 0%
C: 5%
D: 28%
E: 68%

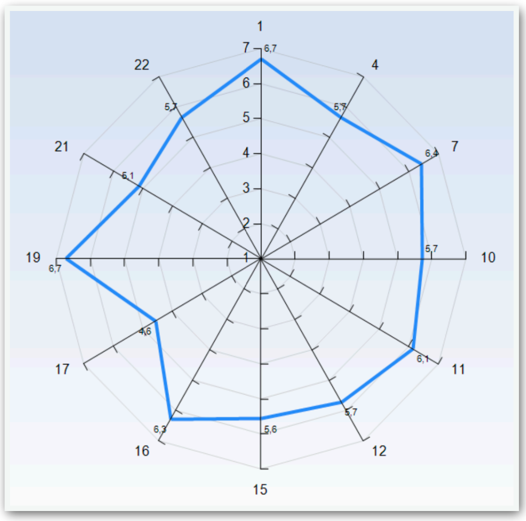
This year, the grades were lower than the previous year. However, several students indicated that they aim to improve their grades, since we allow resubmissions in March and June 2022.

Student Workload

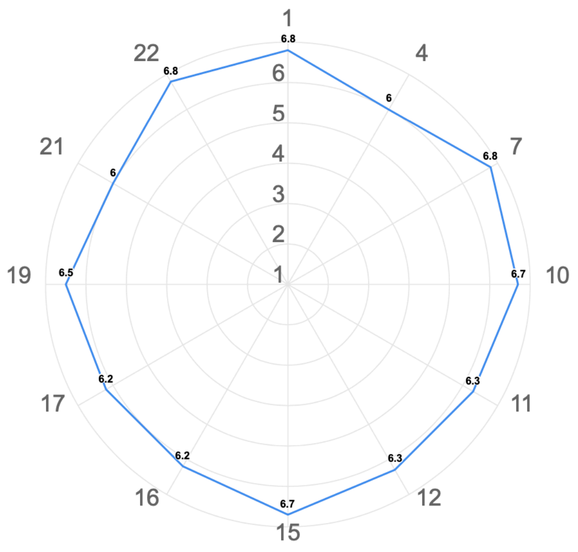
In the web-based course evaluation, the students answered that they worked between 15h per week to more than 41h per week with this course. In general, the students said that the workload was high, especially module 1. Note that this course is 7.5hp and studied during one period, that is, half-time studies. 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 2020 and 2021:



Year 2020



Year 2021

It is interesting to note that year 2022 is very high on basically all questions. In particular, questions 17 ("My background knowledge was sufficient to follow the course") and 21 ("I was able to learn by collaborating and discussing with others") have improved this year. One possible reason is that the OCaml crash course in the beginning helped the students to get into the assignments in a better way than last year.

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

- Many students liked the mix of theory and practice.
- Many students were very satisfied with the lectures and the way the course was structured.
- Several students were very satisfied with the work by the teaching assistants.
- There was some feedback that the workload was too high.
- Some students said that we can add more examples of how to use the parser generator tool in the first module, and that the connection to the calculator example should be clearer.
- The new OCaml tutorial in module 0 was very appreciated.
- One student thought that the grading levels were too strict.
- In general, most of the students were very satisfied with the course. Some students said that this was the best course they have taken at KTH and some of the best lectures they have experienced.

4. Analysis and Planned Course Development

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

Background knowledge and workload
Compared to last year, fewer students answered that they did not have the right background knowledge. However, 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. The experienced workload can also directly connect to the background knowledge in general, and for more advanced programming in particular. During this year, we introduced a crash course in OCaml the first week, which was very appreciated. We plan to keep this module also next year.

To improve the situation next year, we plan to do the following:

- **OCaml crash course.** We will keep the crash course that we introduced this year. However, this year, the assignments for the crash course were optional. Next year, we will also consider including them as part of the deadline bonus. We will also have more guidelines for using parser generators, something that was not included in module 0 this year.
- **Workload.** We will consider further reducing the workload, especially in module 1 and in the more advanced parts of module 2.

Lectures, seminars, and examination
In general, the feedback on the seminars, exercises, and examinations was very positive. We do not plan to make significant changes to the structure. We are really happy that the students liked the way we organized the formative assessment and the examination, and we plan to continue to use the same form of examination next year.

Conclusions

To conclude, we believe that the course was very much well-received. This year, we received top scores on basically all areas in the course evaluation. This is also the same feeling we received from students during oral sessions and via email. The main challenge for next year is to reduce the workload and still keep the same learning outcomes. Also, the final grades are lower than we expected, especially when compared to last year. These areas will be the focus for next year's edition. In summary, we are delighted that the course is so appreciated and we hope to make it even better next year.