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 2020 (Period 2)
Level and credits: Master's level course, 7.5hp

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 and parsing.*
  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 and recorded using Zoom.
- *Hacking sessions (LA).* Each week, there are two online hacking sessions. During these online 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.

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 first course edition for ID2202 that has been taught by the course responsible and examiner David Broman. The course objectives have not changed to any larger extent compared to previous years, but both the learning activities and examination structure have changed completely. The main differences can be summarized as follows:

- Previously, there was a written (on paper) exam at the end of the course. This has been replaced with several examination tasks during the course, as parts of the three modules.
- Previous edition did not include any practical part, where the student had to write software and develop a compiler. In this new design, one of the key parts is to iteratively develop a small compiler for an educational source language called Cigrid (a small subset of C/C++).

Because of the pandemic during 2020 and that it was the first time the course was given with the new format, we continuously adapted the course to meet new demands.

- In the beginning of the course, all lectures were given in a hybrid mode. That is, the lectures were given in-person at KTH in the lecture room and it was also broadcasted using Zoom. Hence, students could attend either in-person or digitally via Zoom. The lectures were also recorded and could be viewed afterwards.
- As it turned out, module 1 took significantly more effort to complete for the students than first expected. For this reason, the deadlines for the modules were moved forward, and we added some extra retake opportunities during the spring 2021.

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, at 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 adopt and make the course better during the course.
- We planned to have a so called *battery evaluation* during one of the lectures, where students gave feedback about the course. However, since the course needed to change format because of Covid (the latter parts of the lectures were all online), we did not perform the usual battery evaluation. Instead, we had more extensive feedback and discussions via email directly with the students.
- After the course, we sent out the usual web-based course evaluation form. As expected, we received a pretty low answer rate: 7 answers out of 32 registered students. However, note that out of the 32 students, only 15 students handed in any exercise, 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.
- After the course, we had a course committee meeting. The meeting was held online via Zoom. Two students had the possibility to attend online during the Zoom meeting, and a third student provided written feedback after the meeting.

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 at the end of the course. In this meeting, the students 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. In the beginning when the lectures were in-person, we also had direct discussions during lecture breaks. 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. Students from both gender gave significant 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. In all lectures, we have recorded the information, subtitled, and used microphones in lecture rooms.

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 32 students signed up for the course according to the Canvas system. Out of these students, 15 students submitted at least one exercise. Out of these, 14 students received a final course grade within this course round. 8 students got the deadline bonus (submitting on time). The grade distribution of the passed students was as follows (rounded numbers):

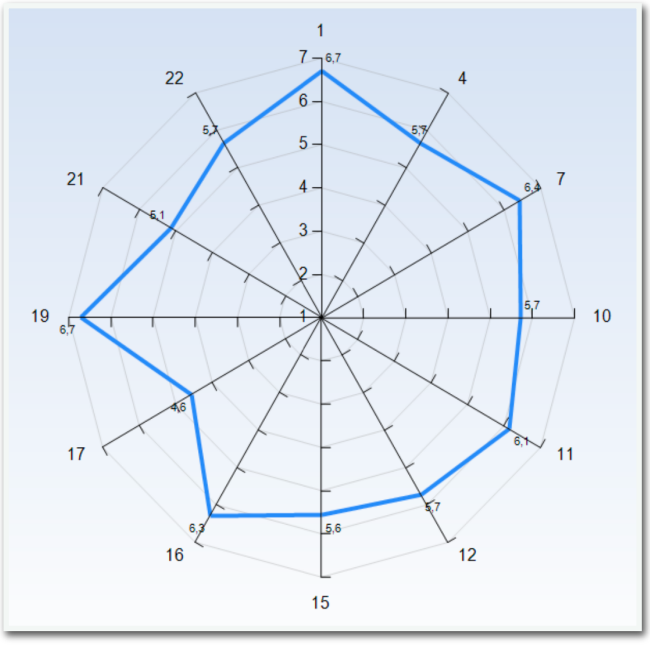
A: 21%
B: 21%
C: 0%
D: 21%
E: 35%

Student Workload

In the web-based course evaluation, the students answered that they worked between 18h per week to more than 41h per week with this course. In general, the students said that the workload was high. 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. Most students answered that module 1 had the highest workload because many students chose to learn a new programming language for the exercises. We will come back to this point later in the analysis.

Student Responses

The web-based student responses resulted in the following Learning Experience Questionnaire (LEQ)-diagram:



The scores are in general quite high, especially on question 1 (“I worked on interesting issues"), 7 (“the intended learning outcome helped me to understand what I was expected to achieve”), 11 (“Understanding of key concepts had high priority”), 16 (“The assessment on the course was fair and honest”), and 19 (“The course activities enabled me to learn in different ways”). These numerical scores also correspond very well with the oral feedback received during the course committee meeting.

The lowest grade was on 17 (“My background knowledge was sufficient to follow the course”) and 21 (“I was able to learn by collaborating and discussing with others”). We will come back to these issues of background knowledge and possibility to collaborate with outers in the analysis.

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

- Many students answered that the course was very stimulating, in a positive way and that they learned a lot. A few students said that the high workload and the difficulties with the exercises made it stressful.
- Many students expressed that they really liked the practical aspects of the course, that they learned to write a small compiler by themselves. Some students said that they did not think that the theory part was as engaging.
- Many students expressed that they really liked the way that the course was examined. By not having a final written exam, they said that it removed a lot of stress. Also, several students expressed that they really liked the way of having automatic correction when submitting code, as well as having peer and TA feedback during seminars.
- Some students did not find the course book that useful, whereas other students emphasized that the they really liked it and that they could learn more in-depth things from it.
- Some students said that we can give more hints on the design of the compiler, especially how to prepare for the design of module 3 already at module 2.
- Several students said that it was really demanding to learn a new language by themselves. Suggestions were made that we should focus on one language, which is explained to everyone.
- Some students suggested that we should also have small review session for some parts of the modules, especially for the calculator in module 1 and the assembler part in module 2.
- Some students thought that we should be stricter to enforce code comments, since reviewing of other students’ code is part of the course.
- Some students thought that one of the hardest parts was to figure out how to structure the code.
- Some students thought that it was a bit too much parsing working in module 1. It was time consuming.
- The students liked the lectures in general, and the in-person lectures in particular. Some of the lectures were delayed (because of changes and adapting the course) which most students thought was OK under the circumstances.
- Although the hacking sessions via Zoom did not fully result in their potential possibility for collaboration, several students emphasized that they liked it because they could ask questions to the TAs.
- The students liked the exercises and the seminars. They said that they learned a lot and that it was really good to have a close discussion with the teaching assistants.
- Several students really liked the examination (automatic grading, seminars etc.), sometimes using the term “amazing examination”. They said that it reduced stress and was motivating and fun. The deadline bonus system was also positively received.

- We received several comments that they liked the course a lot; that it was very hard, but also really fun.

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

Many students stated that they did not have the background knowledge for this course. A problem with the course setup is that some students are allowed to take this course, even if they have a very brief computer science background. It is naturally very hard to teach a rather complex subject, such as compilers, if the student's background is not adequate. However, what we also saw was that many students lacked programming experience in general, and especially in functional programming. Learning a new language, which many students did the first couple of weeks, was very time consuming and might have resulted in that some students spent too much time on this, compared to learning the core concepts of compilers. Since it was up to the students themselves to choose the language that they were going to implement the compiler in, some students chose a language such as Rust, which is rather hard to learn for a beginner, especially because of its quite sophisticated type system.

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

- **Allow many languages, recommend one language.** Within the TA team and together with students, we concluded that it is still a very good idea that students can choose language, but we should have a default recommended language. As a consequence, we decided to use OCaml as the default functional language, because it is rather 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 the next course round will be that we will have a one week crash course on OCaml (in parallel with the introduction to compilers), to get everyone up to speed from start. The focus will be on typed functional programming and how it is extremely useful when developing compilers.
- **Design guidelines.** In the new course edition, we will provide more design guidelines, to make it easier to create a compiler from scratch. There will also be more requirements to comment code, to make it easier for peer reviewing. We will also reconsider the workload for Module 1, and possibly simplify some of the parsing work, to make it more lightweight. In particular, we will add further guidelines to make the transition from the calc interpreter and the Cigrid parser easier.

Collaboration

As shown in the LEQ diagram, one of the lowest scores was on the possibility to collaborate with other people. In this course, the aim has been that each student should individually design their own compiler and write all their code themselves. This is intentional, since the learning outcome in this course is not team work, but personal engineering skills within the topic of compilers. However, the learning activity called hacking sessions was designed to enable discussions and collaboration. Due to the pandemic, this activity had to be online via Zoom, which drastically lowered the actual collaboration and discussions between students. Only few students spoke up at the Zoom meetings, and it is clear that online hacking sessions did not work as well as in-person hacking sessions, as we planned for.

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

- **Lectures and hacking sessions in-person.** If allowed, all lectures will be in-person, but also hybrid (broadcasting via Zoom). All hacking sessions will be in-person, to facilitate better opportunities for collaboration.
- **Online office hours.** In addition, there will also be online office hours, where teaching assistants are available to answer questions over Zoom.

Lectures, seminars, and examination
In general, the feedback on the seminars, exercises, and the examination was very positive. We do not plan to make major changes of the structure, except for the extra crash course in the beginning of the course. We are really happy that the students liked the way we organize 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, even if it was the first time it was given with a completely new structure, exercises, lectures, and examination. Students were very positive about the examination format, and the help given by both the examiner and the teaching assistants. The main areas of improvement are to focus on one implementation language, and to give a crash course on that language in the beginning of the course, to get everyone up to speed early on in the course. We will also look over the distribution of workload, especially for module 1.