FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Review report of a final thesis

Reviewer:	Ing. Konrad Siek, Ph.D.
Student:	Bc. Michal Vlasák
Thesis title:	x86-64 native backend for TinyC
Branch / specialization:	System Programming
Created on:	21 August 2023

Evaluation criteria

1. Fulfillment of the assignment

▸ [1] **assignment fulfilled**
[2] assignment fulfilled with minor objections
[3] assignment fulfilled with major objections
[4] assignment not fulfilled

The student was tasked with designing and implementing a TinyC compiler. This involves attaining a familiarity with the existing TinyC front-end, designing and implementing a complete compiler for x86 as well as runtime with memory and I/O support. The design is meant to be educational in nature, showcasing features of the NI-GEN course and accessible to students. Overall, the student completes the task satisfactorily: the thesis provides an in-depth description of the state of the art and the design of the compiler, which is then implemented through the submitted code.

2. Main written part

95 /100 (A)

Summary:

Overall the thesis is very good. The thesis is well written and structured in a way that makes it understandable. It is reasonably well edited, with only minor editing errors, and without any serious or factual errors that I could find. The thesis uses citations correctly and is the best-sourced Master's thesis I have seen so far, containing some 50 papers, textbooks, and technical manuals cited throughout.

Details:

The thesis follows a relatively standard structure, consisting of:

- an introduction laying out the background relating to NI-RUN and the overall goals,
- a chapter on the x86 architecture and a chapter on compiler design, both of which serve as a good introduction to the topic, conveying a tremendous amount of information within a (relatively) small number of pages,
- a chapter describing both design and implementation:
- specific requirements for the designed compiler,
- explains design choices, including language and framework choices,
- the implementation of compiler internals,
- description of each stage of the compiler (SSA deconstruction, register allocation, lowering, peephole optimizations, etc.)
- description of the middle-end and runtime system.
- a short evaluation chapter describing known problems and comparing the generated output against GCC, and
- conclusion and future work.

The structure flow logically and is easy to understand. My one objection is that the design and implementation chapter is very long and would benefit from being split into a design chapter outlining what the student wanted to accomplish and a separate implementation chapter describing the resulting compiler---these elements are all there, but they are jumbled together.

I have not discovered any inaccuracies or factual errors in the thesis.

The work is not absent editing errors, but these are minor in nature (mostly punctuation). I mention them only for completeness.

The relevant sources are cited correctly and this might be the best-sourced Master's thesis I have seen so far, containing some 50 cited sources, including papers, textbooks, and technical manuals. There are some assumed general knowledge statements that are not cited but this can be forgiven (e.g. "such representations are considered state-of-the-art," pg. 63). The formatting of the citations and the bibliography is good.

To the best of this reviewer's understanding, software and other copyrighted works are used in accordance with their license terms, if any.

3. Non-written part, attachments 100/100 (A)

Summary:

The review evaluates the design in terms of use by students for education. As such, the design conforms to expectations and the implementation is very clean and explicit.

Details:

Since the compiler is meant to be interacted with by undergraduates, the review focuses on how well the compiler reflects a state-of-the-art compiler implementation, and how easy it is to understand the code. The review is not interested in the implementation beating a state-of-the-art compiler in any way.

The student demonstrated the implementation, after which I used it independently to compile toy examples. I experienced no problems compiling and running the code.

The structure of the code is easy to follow, with files reflecting specific stages of the compilation (lower, peephole, etc.) or architectural elements (environment, arena, table).

The code is clean and follows good C-coding standards throughout. It is generally explicit and avoids hacks, which is in line with its educational nature.

While I would have liked to see systematic comments providing descriptions of data structures, function, etc. throughout the code, there are comments where they are necessary to explain things to the reader.

The design is straightforward and conforming to expectations. It consists of a parser, middle-end, runtime, and back-end, the latter-most comprised of a number of a number of stages based around CFG transformations: critical edge splitting, SSA deconstruction, line range splitting, lowering, peephole optimization, and register allocation.

4. Evaluation of results, publication outputs and awards 95 /100 (A)

Summary:

The thesis is meant as an educational tool. In the opinion of this reviewer the work can be used for the course for which it was designed with only minor adjustments.

Details:

The submitted code is easy to follow and modified by students. The implementation is advanced enough as to show off the features of the course (peephole optimizations etc), while simultaneously retaining a simple enough design that students could read, understand, and modify it.

The thesis also serves as a reasonable introduction to many of the core concepts in the course and could be used by the instructor as companion material. I would advise some restructuring of the chapters to make the design and implementation chapter easier to reference.

The overall evaluation 100 /100 (A)

The student approached the task thoughtfully and executed it very well.

Instructions

Fulfillment of the assignment

Assess whether the submitted FT defines the objectives sufficiently and in line with the assignment; whether the objectives are formulated correctly and fulfilled sufficiently. In the comment, specify the points of the assignment that have not been met, assess the severity, impact, and, if appropriate, also the cause of the deficiencies. If the assignment differs substantially from the standards for the FT or if the student has developed the FT beyond the assignment, describe the way it got reflected on the quality of the assignment's fulfilment and the way it affected your final evaluation.

Main written part

Evaluate whether the extent of the FT is adequate to its content and scope: are all the parts of the FT contentful and necessary? Next, consider whether the submitted FT is actually correct – are there factual errors or inaccuracies?

Evaluate the logical structure of the FT, the thematic flow between chapters and whether the text is comprehensible to the reader. Assess whether the formal notations in the FT are used correctly. Assess the typographic and language aspects of the FT, follow the Dean's Directive No. 52/2021, Art. 3.

Evaluate whether the relevant sources are properly used, quoted and cited. Verify that all quotes are properly distinguished from the results achieved in the FT, thus, that the citation ethics has not been violated and that the citations are complete and in accordance with citation practices and standards. Finally, evaluate whether the software and other copyrighted works have been used in accordance with their license terms.

Non-written part, attachments

Depending on the nature of the FT, comment on the non-written part of the thesis. For example: SW work – the overall quality of the program. Is the technology used (from the development to deployment) suitable and adequate? HW – functional sample. Evaluate the technology and tools used. Research and experimental work – repeatability of the experiment.

Evaluation of results, publication outputs and awards

Depending on the nature of the thesis, estimate whether the thesis results could be deployed in practice; alternatively, evaluate whether the results of the FT extend the already published/known results or whether they bring in completely new findings.

The overall evaluation

Summarize which of the aspects of the FT affected your grading process the most. The overall grade does not need to be an arithmetic mean (or other value) calculated from the evaluation in the previous criteria. Generally, a well-fulfilled assignment is assessed by grade A.