15-411 Compiler Design, Fall 2021
Lab 6: Create Your Own Adventure

Seth and co.

Compiler Due: Tuesday, December 14, 2021
Term Paper Due: Tuesday, December 14, 2021

1 Introduction

The main goal of this project is to explore advanced aspects of compilation. Your team will have the opportunity to implement a novel compiler feature you find interesting.

2 Requirements

You are required to hand in three separate items:

- The working compiler and support materials (runtime, etc.) that implement your proposed project.
- Any additional tests and framework to test your project.
- A term paper describing and critically evaluating your project.

2.1 Possibilities

From writing the project proposals you submitted a week ago, you should have a good idea of what you want to work on. Be sure to read and address feedback from your proposal before starting, including meeting with the course staff if necessary. Hopefully, you have had time to think through the implementation and consider the feasibility of your ideas in the last few days! If (after thorough consideration) you decide you'd like to significantly modify your proposed adventure, please alert the course staff as soon as possible.

Of course, there are also two preset options you can choose (although we encourage you to go with your proposed project instead, which is likely more fun and interesting): extending your compiler to `C1` or adding garbage collection. Although the deliverable requirements, grading scheme, and deadlines for a preset option are the same as create-your-own-adventure, we provide separate instructions that you may view on the course website.

2.2 Tests and Measurement Tools

You need to demonstrate that your compiler is correct. How you do this will obviously depend heavily on the project you propose. You may need to write new tests to exercise the Lab 6 portions

of your compiler; you may need to construct a customized testing framework. Before proposing a project, you should give some thought to your testing approach. If there will be no way to check whether your compiler meets the goals of your project, we will probably not be satisfied with your project!

2.3 Term Paper

Your paper should follow this outline.

1. Introduction. This should provide an overview of the project you completed, give a sketch of your implementation, and briefly summarize results.

2. Project. Give a specification for your project including metrics for success. If you chose one of the preset options, recap the requirements given in the handout and explain the "extra mile" you added.

3. Implementation. Describe, with a reasonable level of detail, the modifications made to your compiler to meet the project goals, including syntax, semantics, data structures, and algorithms. Describe also any runtime system required for your project.

4. Testing Methodology. Describe the design of your testing approach. Include any relevant information such as the criteria you used as you selected or designed your tests, how you constructed your testing system, and how your testing approach verifies the functionality of your compiler. What you put in this section will, of course, depend on your project, but you should make sure to give example snippets/programs that demonstrate your work.

5. Analysis. Critically evaluate your compiler and sketch future improvements one might make to your current implementation.

The term paper will be graded at the end of the semester. There is no hard limit on the number of pages, but we expect that you will have at least 3–4 pages of concise and interesting analysis to present.

3 Deliverables and Deadlines

All your code should be placed in subdirectories of the `lab6` directory as before. Be sure to create a branch or tag named `lab6` for the course staff to read when you're done. The autograder will run regression tests against your own tests and the same tests it used in Lab 5`--`, but these will *not* directly contribute to your grade (in particular, you may intentionally implement features that are not backwards-compatible). We will grade you based on the code and `README` file(s) you have checked in at the deadline.

3.1 Compiler Files

As for all labs, the files comprising the compiler itself should be collected in the `lab6/` directory which should contain a `Makefile`. **Important:** You should also update the `README` file and insert a roadmap to your code. This will be a helpful guide for the grader. In particular, since there are

likely to be many different projects undertaken, do introduce the project at the very top of the README.

Issuing the shell command

```
% make
```

should generate the appropriate files so that

```
% bin/c0c --exe <args>
```

will run your compiler and produce an executable. It is not necessary to continue supporting any compiler flags besides `-t`.

After running `make`, issuing the shell command

```
% make test
```

should run your own tests and print out informative output. The command

```
% make clean
```

should remove all binaries, heaps, and other generated files.

If it is reasonable, you should modify the driver from Lab 5 to test your extended compiler. If there are any special instructions we need to follow in order to be able to run the driver on your compiler and test it, specify these instructions in your README file.

Runtime Environment

Because you may have implemented a new runtime, your compiler should have an additional flag `--exe`. If your compiler is given a well-formed input file `foo.l1` or `foo.l2` as a command-line argument and is also given the `--exe` argument, it should generate a target file called `foo.l1.s` or `foo.l2.s` (respectively) in the same directory as the source file, and should *also* compile the runtime and link it with your generated assembly to create an executable `foo.l1.exe` or `foo.l2.exe` (respectively). We will test your compiler against the regression suite, but the grade reported by the autograder won't directly contribute to your grade.

3.2 Tests and Measurement Tools

In a directory called `lab6/tests/`, include all the tests that you selected or wrote for the purpose of testing your project. If they are to be used in a different way than a vanilla L4 test, you should include a README file explaining exactly how to use your tests, and `make test` should run your tests.

If you also do any performance testing in the same vein as Lab 5, include the necessary files in `bench/`.

3.3 Term Paper

Submit your term paper in PDF form via Gradescope before the stated deadline. Early submissions are much appreciated since it lessens the grading load of the course staff near the end of the semester. **You may not use any late days on the term paper!**

4 Notes and Hints

- Discuss your ideas with the course staff to get feedback on feasibility and scope of the project before embarking on it.
- Try to identify some intermediate goals in case your overall project turns out to be too ambitious. You should have some intermediate goals to draw on from your proposal.
- Apply regression testing. It is very easy to get caught up in new features. Please make sure that the L4 portion of your compiler continues to work correctly (keeping in mind that not all changes are expected to be backward-compatible)!