15-411 Compiler Design: Lab 6 - LLVM
Fall 2008

Instructor: Frank Pfenning
TAs: Rob Arnold and Eugene Marinelli

Compilers due: 11:59pm, Thursday, December 4, 2008
Term Paper due: 11:59pm, Thursday, December 11, 2008

1 Introduction

The main goal of the lab is to explore advanced aspects of compilation. This writeup describes the option of retargeting the compiler to generate LLVM code; other writeups detail the option of implementing garbage collection or optimizing the generated code. The language *L5* does not change for this lab and remains the same as in Lab 5.

2 Requirements

You are required to hand in two separate items: (1) the working compiler and runtime system, and (2) a term paper describing and critically evaluating your project.

3 Tests

You are not required to hand in new tests. The autograder will use a subset of the tests from the previous labs to test your compiler.

4 Compilers

Your compilers should treat the language *L5* as in Lab 5, including `extern` declarations. While we encourage you to continue to support both safe and unsafe compilation, but it complies with the specification if the potentially unsafe implementation is simply the safe one.

When generating code for the LLVM, given file *name*`.l5`, you compiler should generate at least two files: *name*`.ll`, which is in the LLVM human-readable assembly language, and *name*`.s`, the x86-64 assembly language. Translation from the former to the latter is likely in at least two stages, first generating an intermediate byte code file *name*`.bc` with `llvm-as` and then *name*`.s` by using `llc` (see the LLVM documentation).

5 What to Turn In

On the Autolab server, the hand-in and status pages for the optimization and garbage collection projects are separated, since different drivers will be employed.

Compiler Files (due 11:59pm on Thu Dec 4)

As for all labs, the files comprising the compiler itself should be collected in a directory `compiler/` 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.

Issuing the shell command

```
% make l5c
```

should generate the appropriate files so that

```
% bin/l5c --safe --llvm <args>
% bin/l5c --unsafe --llvm <args>
% bin/l5c --safe --x86_64 <args>
% bin/l5c --unsafe --x86_64 <args>
```

will run your *L5* compiler in safe and unsafe modes, generating LLVM or direct x86-64 native code, respectively. For backwards compatibility, the default is `--unsafe --x86_64`

The command

```
% make clean
```

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

Using the Subversion Repository

The recommended method for handout and handin is the course subversion repository.

The handout files for this course can be checked out from our subversion repository via

```
% svn checkout https://cvs.concert.cs.cmu.edu/15-411/<team>
```

where `<team>` is the name of your team. You will find materials for this lab in the `lab6llvm` subdirectory. Or, if you have checked out `15-411/<team>` directory before, you can issue the command `svn update` in that directory.

After first adding (with `svn add` or `svn copy` from a previous lab) and committing your handin directory (with `svn commit`) to the repository you can hand in your tests or compiler by selecting

```
S5b - Autograde your code in svn repository
```

from the Autolab server menu. It will perform

```
% svn checkout https://cvs.concert.cs.cmu.edu/15-411/<team>/lab6llvm/compiler
```

to obtain the files directories to autograde, depending on whether you are handing in your test files or your compiler.

If you are submitting multiple versions, please remember to commit your changes to the repository before asking the Autolab server to grade them! And please do not include an compiled files or binaries in the repository!

Uploading tar Archives

A deprecated method for handout and handin is the download and upload of tar archives from the Autolab server.

For the test cases, bundle the directory `tests` as a tar file `tests.tar` with

```
% tar -cvf tests.tar tests/
```

to be submitted via the Autolab server.

For the compiler, bundle the directory `compiler` as a tar file `compiler.tar`. In order to keep the files you hand in to a reasonable size, please clean up the directory and then bundle it as a tar file. For example:

```
% cd compiler
% make clean
% cd ..
% tar -cvf compiler.tar --exclude CVS compiler/
```

to be submitted via the Autolab server. Please do not include any compiled files or binaries in your hand-in file!

Term Paper (due 11:59 on Thu Dec 11)

You need to describe your implemented compiler and critically evaluate it in a term paper of about 10 pages. You may use more space if you need it. The recommended outline varies depending on your project. Submit a file `<team>-llvm.pdf` via email to the instructor at `fp@cs`.

Your paper should follow this outline.

1. Introduction. This should provide an overview of your implementation and briefly summarize the results you obtained.

2. Comparison. Compare compilation to LLVM, followed by native code generate with direct native code generation. How does the structure of your compiler differ? How does the generated code differ? If you are applying optimizations at the LLVM level, describes those optimizations and their rationale.

3. Analysis. Critically evaluate the results of your compiler via LLVM, which could include size and speed of the generated code. You might find the driver for the optimization lab `lab6opt` to be useful for this purpuse. Also provide an evaluation of LLVM: how well did it serve your purpose? What might be improved?

6 Notes and Hints

- Apply regression testing. It is very easy to get caught up in writing a back end for a new target. Please make sure your native code compiler continues to work correctly!

- Read the assembly code. Just looking at the assembly code that your compiler produces will give you useful insights into what you may need to change.