CS 335 Semester 2019–2020-II: Project Description

13th Mar 2020

Due Your submission is due by Mar 29th 2020 11:59 PM IST.

General Policies

- Do not plagiarize or turn in solutions from other sources. We WILL check your submission(s) with plagiarism checkers. You will be PENALIZED if caught.
- Each group will get a total of FOUR FREE DAYS to help with your project submission deadlines. You can use them as and when you want throughout the duration of the project.

Description

The goal of this project is to implement a compilation toolchain, where the input is in Java language and the output is x86 assembly.

Milestone 3

Extend your 3AC IR from Milestone 2 to add runtime support for procedure calls. Your implementation of activation records MUST INCLUDE the following fields, the exact order of the fields depends on you and the conventions of the target x86 architecture.

- space for actual parameters,
- space for return value,
- space for old stack pointers to pop an activation,
- space for saved registers,
- space for locals.

You can optionally add other fields required by your implementation. Remember that you may need to define 3AC instructions corresponding to various features allowed in the input program (for e.g., `call`, `pushparam`, and `return` where the semantics is obvious from the name).

Output.
For correct input programs, your implementation should output a text dump of the 3AC of the input Java program with runtime support for activations.

The plan for the next and final Milestone is to generate correct x86 assembly from 3AC which can be run (say NASM or GAS on Linux). So it is important that you do a good job with runtime support for procedures that is tailored to the target assembly syntax.

We will discuss code generation shortly in class, but you might want to refer to Chapter 8 from the Dragon book to get a feel of the requirements for code generation.

Submission.

- Submission will be through Canvas.
- Create a zip file named "`cs335_project_ <roll>.zip`". The zipped file should contain a folder `milestone3` with the following contents:
  - All your source files must be in `milestone3/src` directory. You are free to choose your implementation language.
  - Submit ten nontrivial test cases that your implementation can handle.
  - Create a directory `milestone3/tests` for your test cases. Name the test files as "`test_ <serial number>.java`".
  - Use `Makefile` (or equivalent build tools like `ant`) for your implementation. You are free to use wrapper scripts to automate building and executing your compiler.
  - Your submission must include a `milestone3/doc` directory and a `PDF` file. The `PDF` file should describe any tools that you used, and should include compilation and execution instructions. Document all command line options.

    You SHOULD USE LaTeX typesetting system for generating the PDF file.

Evaluation

- Your implementation SHOULD FOLLOW THE PRESCRIBED STEPS so that the expected output format (if any) is respected.
- We will evaluate your implementations on a Unix-like system, for example, a recent Debian-based distribution.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.
- The TAs will meet with each group for evaluation. Make sure that your implementation builds and runs correctly. A typical evaluation session could be as follows.

```
$ cd <milestone3>/src
$ make
$ ./milestone3 ../tests/test5.java --output=test5.3ac
$ cat test5.3ac
```