CS 335 Semester 2022–2023-II: Project Description

29th Mar 2023

Due Your submission is due by Apr 19th 2023 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. For example, you can submit Milestone 1 two days late without penalty, and you will have two free days remaining to potentially utilize for the rest of the semester.

Description

The goal of this project is to implement a compilation toolchain, where the input is in Java language and the output is x86_64 code.

Milestone 4

Implement a translator to generate x86_64 assembly instructions from the three-address code generated in Milestone 3.

You can assume that the input test programs will only use integral types. This constraint should not impact your Milestone 3 implementation, should simplify register manipulation, and will avoid the complexities with floating-point instructions in the x86_64 ISA.

Refer to Chapter 8 from the Dragon book (2nd edition) to get a feel of the requirements for code generation. You can use a local register allocation scheme for simplicity.

The primary evaluation criteria are correctness and completeness. For example, the assembly code generator should not fail while translating large expressions or for lack of available registers. Your implementation should, however, generate meaningful error messages for unsupported features.

Showcase end-to-end handling of optional Java language features for bonus marks. Include test cases to highlight the bonus features during the demo of Milestone 4.

Use the PDF file (part of your deliverables) to document the following.

- Highlight the basic features supported by your implementation,
- Highlight the optional features (if any) supported by your implementation,
- Provide command lines and test cases to demo end-to-end execution of the test cases,
- Extensions (if any) to the 3AC from Milestone 3 for implementing Milestone 4,
- Manual changes (if any) to the generated assembly file to run it successfully with `as` or `gcc`,
- List any basic features that you could not support and the associated challenges,
- Include a table with the following structure, one row for each member: #, Member Name, Roll Number, Member Email, Contribution (%).

Output.

Test the correctness and execution of the generated assembly code using the GNU assembler invoked using `as` or `gcc`.

Submission.

- Submission will be through Canvas.
- Create a zip file named "`cs335_project_ <roll>.zip`". The zipped file should contain a folder `milestone4` with the following contents:
  - All your source files must be in `milestone4/src` directory.
  - 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.
  - Use the directory `milestone4/tests` and your previously-designed test cases to showcase your implementation.
  - Your submission must include a `milestone4/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 is respected.
- We will evaluate your implementations on a recent Debian-based distribution.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.
- Our evaluation will only focus on correctness, the compilation time is not important.
- 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 <milestone4>/src
        $ make
        $ ./milestone4 --input=../tests/test5.3ac --output=test5.s
        $ gcc -c test5.s -o test5.o; gcc test5.o -o test5;
        $ ./test5
```