# CS 335 Semester 2023–2024-II: Project Description

26th Mar 2024

**Due** Your submission is due by Apr 17 2024 11:59 PM IST.

**General Policies**

- Do not plagiarize or turn in solutions from other sources. 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 project. For example, you can submit Milestone 3 one day and ten hours late without penalty, provided you have two free days left.

# Description

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

## Milestone 3

Implement a translator to generate x86_64 assembly from the three-address code output in Milestone 2. The generated assembly code should run via GAS on Linux. Study the assembly code of a few C programs and refer to x86_64 assembly while implementing runtime support for function calls.

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. You can (optionally) implement a local register allocation scheme for efficiency.

The input test cases for Milestone 3 will not use floating-point types. This constraint will avoid the complexities with floating-point instructions in the x86_64 ISA and should simplify register manipulation.

**Output.** Showcase end-to-end translation of the Python language features (listed in the project) with at least five new test cases for Milestone 3. Test the correctness and execution of the generated assembly code using the GNU assembler invoked using `as` or `gcc`.

```
$ cd <milestone3>/src
$ make
$ ./milestone3 --input=../tests/test1.3ac --output=test1.s
$ gcc -c test1.s -o test1.o; gcc test1.o -o test1;
$ ./test1
```

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

- Highlight the required and optional (if any) language features supported by your implementation,

- Provide command lines and test cases to demo end-to-end execution of the test cases,
- Modifications (if any) to the 3AC from Milestone 2 for implementing Milestone 3,
- Manual changes (if any) to the generated assembly file to run it successfully with `as` or `gcc`,
- List any required features that you could not support and the associated challenges,
- Effort sheet.

**Submission.** Submission will be through BOTH Canvas and CSE Git.

**Canvas** Create a zip file named "`cs335-project-<groupid>.tar.gz`" and upload it to Canvas. Alternate extensions like `.tar.xz` and `.zip` are allowed. Only one group member should submit the project on behalf of her project mates. The compressed file should contain a folder `milestone3` with the following contents:

- All your source files must be in `milestone3/src` directory.
- Use the directory `milestone3/tests` to include the five new test cases as described above.
- 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 `PDF` file under the `milestone3/doc` directory. The `PDF` file should include compilation and execution instructions, and the other instructions listed above. You SHOULD USE LaTeX typesetting system for generating the PDF file.

**Git** Create a tag called milestone3 BY THE deadline. This is the tag that will be checked out and evaluated by the TAs.

You can create and push tags using the following commands.

```
git tag milestone3 -a
git push origin milestone3
```

**Evaluation.**

- Your implementation should follow the prescribed steps so that the expected output format 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 inputs and test cases, so remember to test thoroughly.
  - Input test cases will use two spaces for indentation and will have a final newline
  - All input test cases will have an `if __name__ == "__main__":` block
- Our evaluation will mostly 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.