Project C

Assistants: Antoine Van Muylder & Sander Huyghebaert

1 Instructions

The goal of project C will be to

1. polish your previous submissions (project A, B)
2. achieve milestone 5
3. extra: achieve milestone 6

If you are not able to finish the whole project, we encourage you to submit a partial solution. For example, if you submit a polished version of project A and B without solving milestone 5, this may be enough to pass the course project, albeit with a lower grade.

1.1 Practicalities

- The due date for project C is: *Sunday 30 May 2021 23:59 CET*.
- The project is individual (1 person = 1 project). Please remember that sending code or sharing code is strictly not allowed.
- You should send a zip file on canvas containing your solution for each milestone. Make sure you keep the *compiler.rkt* file provided by the skeletons, so we can easily access your implementation for every exercise.
- Make sure your code runs with the expected Racket and nasm versions by running it on the wilma server before submitting it.
- A short defence of your project will be held on the day of the final exam.

1.2 Milestones and readings

Project C consists of 2 milestones[1]

- "**Milestone 5:** Adding Call"
- "**Milestone 6:** Adding Return" is optional but will be taken into account in the grade, if achieved.

Each milestone specifies a set of "readings"[2] that are necessary to read in order to complete the exercises. Make sure to read and understand them!

Some milestones also contain **Challenge** exercises, these challenges are optional and solving them correctly will increase your grade for this project.

[1] `https://soft.vub.ac.be/compilers/milestone_top.html`
[2] `https://soft.vub.ac.be/compilers/book_top.html`

1.3 Testing

Make sure to thoroughly unit-test each compiler pass you implement (this will be graded as well). We would like you to write your own original tests. You can get some inspiration for tests by using the online interrogator. It is good practice to write your tests and design your code before implementing it. You can include your tests in the special `test` submodule

```
(module+ test
  (require rackunit)
)

...;;a compiler pass

(module+ test
  (check-equal ... )
  ;; tests about the pass
)
```

Note that the `module+` form allows you to split the definition of a module (here the test submodule) into several fragments.

1.4 Grading

The project assignment needs to be done individually. Sending code or sharing code is strictly not allowed. Your project assignment will be evaluated on the basis of 4 different criteria:

1. **Overall architecture of the code.** Is there a clear logical separation between the tasks you are trying to solve?
2. **Implementation.** Are the non-obvious aspects of your implementation properly documented in comments in the code? Does the code uses meaningful identifiers? Do you successfully avoid code duplication? Does it use the Racket standard library when possible? etc...
3. **Tests.** Did you write your own tests? Are edge cases tested? Are all the cases tested?
4. **Correctness.** Is your compiler correct? In other words, is there a difference between (1) intepreting ("running") the source code or (2) compiling the source code and interpreting the target? We ask you to provide tests for this as well.

1.5 Questions

You can resort to the online interrogator if you have doubts about a certain pass:

- **Milestone 5:** `https://soft.vub.ac.be/compilers/interrogator?an=a5`
- **Milestone 6**: `https://soft.vub.ac.be/compilers/interrogator?an=a6`

For any further questions you can always use the dedicated canvas forum: `https://canvas.vub.be/courses/20710/discussion_topics/95010`

Or contact `avmuylde@vub.ac.be`.