Project A

Assistants: Antoine Van Muylder & Sander Huyghebaert

1 Instructions

The goal of project A is to design and implement a first compiler having as source language a simple "value-oriented language", and as target `x86_64`. A value-oriented language is a language where operations consume and produce values directly, instead of referring to some underlying machine state that the programmer must always keep in mind and manually manipulate.

1.1 Practicalities

- The due date for project A is: *Sunday 28 Feb 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 2 racket files: one for each evaluated milestone (see below at "skeletons" for more info). Make sure your code runs with the expected Racket and nasm versions by running it on the wilma server (see below) 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 A is split into 3 independent milestones[1]: milestones 0, 1, 2. Only milestone 1 and 2 will be evaluated. Milestone 0 is a warm-up and allows you to setup the toolchain we will be using in the projects.

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!

1.3 Skeletons

Each milestone comes with a Racket skeleton. Make sure to rename this skeleton into `milestonei` (i=1,2) and to keep the `provide`'s of that file unchanged. The rest of the code can be modified to your liking.

- `https://github.com/antoinevanmuylder/compilers-skeletons-vub/tree/assignment-0`
- `https://github.com/antoinevanmuylder/compilers-skeletons-vub/tree/milestone-1`
- `https://github.com/antoinevanmuylder/compilers-skeletons-vub/tree/milestone-2`

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

1.4 Wilma

VUB students can access a Linux server named wilma (wilma.vub.ac.be) where racket and nasm are properly installed. You can claim a wilma account here `https://wilma.vub.ac.be/english/` (same credentials than VUB).

Once you have an acount you can:

- ssh into wilma with `ssh <vub-net-id>@wilma.vub.ac.be`. There you can debug/test your code
- transfer files using the scp protocol

```
$ scp username@from_host:file.txt /local/directory/ #remote-->local
$ scp file.txt username@to_host:/remote/directory/  #local-->remote
```

This should be possible on non Linux machines as well, using equivalents of the `scp` command.

1.5 Custom packages

The skeletons require the following Racket packages: `cpsc411/compiler-lib`, `cpsc411/2c-run-time`. Here is how to install them:

1. Complete the first 3 exercises of milestone 0.
2. Clone this repository on your dev machine (for instance, wilma!) `https://github.com/cpsc411/cpsc411-pub`.
3. Go to the `cpsc411-lib/` directory.
4. Run the command `raco pkg install`.

If the milestones or readings refer to some racket files, those should be in this directory: `cpsc411-pub/cpsc411-lib/cpsc411/`.

1.6 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.

1.7 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 make 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.

1.8 Questions

You can resort to the online interrogator if you have doubts about a certain pass. `https://soft.vub.ac.be/compilers/interrogator?an=a1` `https://soft.vub.ac.be/compilers/interrogator?an=a2`

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

Or contact `antoine.van.muylder@vub.ac.be` preferably in english.

Good work!