Name:

CS 443 — Clojure to C Compiler

Mattox Beckman

Objectives

So you got to do the Lisp to Forth compiler? Great! Let's up the game a bit. This time, we are going to compile Clojure to C. If you don't know C, you are free to use Java or Haskell instead.

- Consider some issues in representing and manipulating expressions in multiple languages simultaneously.
- Consider how to provide a "standard environment" for your compiled program.
- Implement translator to convert Lisp style expressions to C style.

Given Code

Here's an example that shows how you might get started. It's just an idea, feel free to do this any way you want.

```
(ns lisp-to-c
  (require [clojure.edn :as edn]))

(defn compile-def [x]
  ;; your code here!
)

(defn compile-defn [x]
  ;; your code here!
)

(defn compile-exp [x]
  ;; your code here
)

(defn compile [x]
  ;; your code here
)
```

Here is a sample run:

```
lisp-to-c> (compile '((def x 10) (def y (* x 10))
"#include <stdio.h>
int x = 10;
int y = x * 10;

int main() {
  printf("%d\n",y);
};"
```

Your Work

- Build a compiler that will convert a `def` or `defn` into equivalent C.
- Be able to support `printf`, `if`, `let`, `+`, `-`, *, and `return` (on the C side).
- The output should be an executable that, when run, displays the value of the last variable that was `def`-ed.
- You are allowed to assume all variables will be assigned integers.
- You are allowed to output other functions and wrappers to make this work.