COMP 520 Compiler Design
Group Milestone #3

Code Generation for GoLite

Due: Sunday, March 26

Overview:

The purpose of this milestone is to develop the code generator for your compiler. You will do this in two phases (Milestone #3 and Milestone #4)

Question 1: *Computation Intensive Programs* (5 points)

Develop two example programs which perform some reasonably intense computation, something that you think will run for between 10 and 60 seconds when compiled (GoLite). They must perform a useful computation. That is, they must be of interest in a real world setting.

We will use these benchmarks to evaluate the efficiency of the code produced by the code generators.

Question 2: *Valid Programs* (10 points)

At this point your compiler should reject all programs for which code generation is not possible. For 5 constructs of the language as described in the syntax specification, provide a test program that you can use to verify your code generator. These test programs should ensure that all relevant semantics of the constructs are correctly implemented. For instance, your variable declaration test should allow you to check for initialization, and your equality test should ensure that `structs` are comparable. Assignments, short declarations, type declarations and identifiers are also good places to start.

There are some more advanced features of the Go language that are much more challenging to support entirely. We do not expect a perfect code generator, but you should strive to be as complete as possible.

Question 3: *Code Generation Status Report* (10 points)

In this milestone you will start implementing your code generator:

- Decide upon what sort of code you will generate. You can choose between low-level code such as Java bytecode, LLVM-IR, etc. or high-level code such as C, C++, JavaScript, Python, Java, etc.

- Design the code generation patterns for the key constructs of your language. Ensure that you are not choosing a strategy that will generate excessively slow code. For example, if you are generating C code, then attempt to generate code that the C compiler will be able to effectively optimize.
- Implement the code generation for a subset of your language. Choose some key language features. You should choose some simple expressions, statements and control constructs to start with.

For the report you should have:

- Your chosen target language and the associated advantages/disadvantages with your choice.
- A brief summary (less than 1 page) of what you have implemented so far.
- An outline of your 5 test programs from Q2. Explain which construct you are considering, and what use cases you are testing in your code generator.

What to hand in

You will be developing your project in your team's github repository. At each milestone you will create a tag before the due date, and the TAs will review the code associated with that tag. For this milestone you should create a tag called *milestone3*. Information about creating git tags can be found at: `http://git-scm.com/book/en/v2/Git-Basics-Tagging`.

Please ensure the following documents are in your git project.

```
programs/
    benchmarks/     (your two benchmarks programs)
    code/           (your 5 codegen construct tests)

doc/
    milestone3.pdf  (a summary (less than one page) of what you have
                     implemented and an outline of your 5 test programs
                     from Q2)
```