CS 132: Compiler Design, Fall 2007

Assignment 7
Register Allocation and Runtimes

Due: 5pm, Wednesday November 21

E-mailed questions about this assignment should be sent to `cs132help@cs.hmc.edu`.

1. Make a `svn copy` of your Assignment 6 solution, and copy in the contents of `src/a7`.

2. The first task of this assignment is to implement an optimistic register allocator in `color.sml`. A lot of support code has been provided: you do not have to build the interference graph, nor must you actually insert spill code. All you have to do is take the interference graph and return (1) a mapping from nodes to machine registers, and (2) a list of temporaries that will be spilled. If spilling is required, the given sources will insert the spill code and re-run register allocation.

   You also are not required to implement coalescing, though the interference graph representation does include information about move instructions.

   Hint: when you implement the simplify phase (picking an order to remove nodes from the interference graph) I *strongly* recommend you do not modify the graph. (You'll need the original graph back for the coloring phase.) One possibility — though certainly not the only one — is to keep a list of the nodes not yet removed from the graph, and a mapping telling for each node in the graph how many of its neighbors have not yet been removed. There's an implementation of lookup tables in the structure `Graph.M`.

3. Create a file `runtime.c` that implements the functions described in your `runtime.h`. Make sure that it compiles with `gcc -c`.

4. Finally, add to your `runtime.c` file a function `main` that performs the following tasks:

   - Calls `_initalizers()` (The support code you were given handles `Global`s by allocating space in the data segment, and defining a function `_initializers()` that initializes all the globals.)
   - Calls `Start__main` (or whatever label your compiler generates for the static function `Start.main`), passing it any appropriate arguments.

   You can then create executables by saying, e.g.,

   ```
   gcc -o myprog  myprog.bcs.s runtime.c
   ```