CS314 Spring 2014

Assignment 5

Due Tuesday, March 11, **before** class

1 Problem — Lexical/Dynamic Scoping

Assume variable names written as **capital** letters use **dynamic** scoping and variable names written as **lower case** letters use **static** (lexical) scoping. Assume that procedures return when execution reaches their last statement. Assume that all procedure names are resolved using static (lexical) scoping. Show the output of the entire program execution. Label the output with the location of the print statement (e.g.: (*2*): ...)).

```
program main()
{    int A, b;
     procedure f()
     {  int c;
        procedure g()
           {  int c;
              c = 30;
              ... = ...b...    //<<<-------- (*A*)
              print A,b,c; //<<<------ (*1*)
              end g;
           }
        print A,b; //<<<------------ (*2*)
        A = 0; b = 0; c = 0;
        call g();
        print c; //<<<------------ (*3*)
        end f;
     }
     procedure g()
     {  int A,b;
        A = 5; b = 7;
        call f();
        print A,b; //<<<----------(*4*)
        end g;
     }
  A = 2;  b = 3;
  print A,b; //<<<------------ (*5*)
  call g();
  print A,b; //<<<----------(*6*)
  end main;
}
```

2 Problem — Lexical/Dynamic Scoping

This problem is an extension of the previous problem.

1. Show the program with all lexically scoped variable names (**lower case**) replaced by their (level,offset) representation.

2. Show the runtime stack when execution reaches the point marked (*1*), (*3*), (*4*), and (*6*) in the code. Assume that program "main" has its own frame on the stack. Make sure you label all the stack frames with the corresponding program/procedure names and include the allocated local variables (and their particular values) within the frame. Include all control links and access links between the activation records (stack frames), and the value of the frame pointer FP by drawing an arrow to the corresponding location within the stack. **Use the frame layout as discussed in class**.

3. Give the RISC machine code for the non-local access to variable **b** at program point (*A*). The access will need to load the content of variable **b** into a register. Use the ILOC instruction format for your RISC code.