CS 415 Compilers: Problem Set 1
Due date: Friday, February 2, 11:59pm

Problem 1 – ILOC programming

Appendix A in our textbook (EaC) discusses ILOC, a linear assembly code for a simple abstract RISC machine. Here, you will also be able to use an additional instruction that allows you to print out a variable value: outputAI *register*, *constant* (print MEM[CONTENT(register) + constant]).

The code that you are writing follows the memory layout as discussed in class. Byte addresses smaller than 1024 are reserved and should not be used for program variables. Program variables are addressed through offsets from address 1024. The reserved register r_0 needs to contain this value. Given this code shape, the simple example program

```
// no aliasing
a := 6;
b := 10;
c := a + b;
PRINT c;
```

can be written in ILOC as

```
//
//   Simple example program
//
// Assign STATIC_AREA_ADDRESS to register "r0"
         loadI 1024     => r0
// Store value 6 into variable a with @a = 0;
         loadI 6 => r1
         storeAI r1 => r0, 0
// Store value 10 into variable b with @b = 4;
         loadI 10 => r2
         storeAI r2 => r0, 4
// Compute a + b
         loadAI r0, 0 => r3
         loadAI r0, 4 => r4
         add  r3, r4 => r5
// Store value in r5 into variable c with @c = 8;
         storeAI r5 => r0, 8
// Print the value of c
         outputAI r0, 8
```

Write (by hand) ILOC code that corresponds to the following high-level program written in pseudo code.

factorial:

```
a := 5; // input value; must be > 0 -- not tested
result := 1;
WHILE a > 1 DO
  result := result * a;
  a := a - 1;
END WHILE
PRINT result;
```

Your code has to work correctly for changing values of variable “a”. Think of “a” as an input parameter.

For the example program, give two versions: one that uses a register-register model where every value that can safely reside in a register is assigned a “new” virtual register, and one that uses a memory-memory model where all values reside in memory and are only loaded into registers just before they are used. Compare the number of instructions executed in both cases.

You can execute your ILOC code using the ILOC simulator *sim*. You can execute your ILOC code in file “test.i” by saying “*./sim < test.i*”. The simulator is available on the ilab cluster at (∼uli/cs415/spring18/ILOC_Simulator). NOTE: Our ILOC simulator *sim* uses the `br <label>` instruction instead of the `jumpI <label>` instruction.

We will use the ilab cluster for our programming projects. The ”http://ilab.rutgers.edu” ilab cluster page contains the listing of valid hostnames available for this homework. *You have the same home directory across all machines of the ilab cluster.*

Problem 2 – Feasible Registers for ILOC

What is the number of feasible registers in ILOC, i.e., the minimal number of physical registers to execute any ILOC code. Give an argument and use examples.

Problem 3 – Top-down register allocation

Source code

```
program main;
    var a, b, c, d, e, f, g, h: integer;
    begin
      a := 1;
      b := 2;
      c := b - 4;
      d := a + b;
      e := d + 1;
      f := e - c * e;
      g := (d + e) + f;
      h := g + a;
      writeln(h)
    end.
```

1. Show the ILOC code for the above source program assuming a register-register model. Code shape: Once a value of a variable is loaded into a virtual register, it stays in this register throughout its lifetime. Use the memory model as discussed in class, i.e., use the dedicated register r0 to hold the base address 1024. Addresses above 1024 are used for spilling registers.
2. Give the live ranges for all virtual registers (ignore r0). What is MAX_LIVE for your ILOC code?
3. Assume that there are two registers in the feasible set, called f1 and f2, i.e., F = {f1, f2}, and there is the dedicated machine register r0 that we do not consider for register allocation, i.e., is not part of the available register set.
   (a) Show the ILOC code that would be generated by the top-down algorithm discussed in EAC, i.e., no MAX_LIVE consideration, for (MAX_LIVE - 1) available registers for allocation (k-F registers are available). Describe the heuristic that you are using.
   (b) Show the ILOC code that would be generated by the top-down algorithm discussed in class, i.e., with MAX_LIVE consideration, for (MAX_LIVE - 1) available registers for allocation (k-F registers are available). Describe the heuristic that you are using.