CS 331 Compilers **Fall 2017**

Written Assignment 4

Prof. Szajda *Due Friday, December 1, 11:59:59 pm*

This assignment asks you to prepare written answers to questions on code generation, operational semantics, optimization, register allocation, and garbage collection. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work. Written assignments can be turned in at the start of lecture. Alternatively, assignments can be turned in at my office by 11:59:59 PM on the due date.

1. Consider the following arithmetic expression: $7 * ((6 + 4)/2 - 3 + (5 - 4) * 3) + 1$.

   (a) You are given MIPS code that evaluates this expression using a stack machine with a single accumulator register (similar to the method given in class Lecture 12). This code is wholly unoptimized and does not perform transformations sucn as arithmetic simplification or constant folding. How many times in total will this code push a value to or pop a value from the stack (give a separate count for the number of pushes and the number of pops)?

   (b) Now suppose that you have access to two registers `r1` and `r2` in addition to the stack pointer. Consider the code generated using the revised process described in Lecture 12 starting on slide 45, with `r1` as an accumulator and `r2` storing temporaries. How many loads and stores are now required?

2. Consider the following basic block, in which all variables are integers, and ** denotes exponentiation.

```
a:= b + c
z := a ** 2
x := 0 * b
y := b + c
w := y * y
u := x + 3
v := u + w
```

Assume that the only variables that are live at the exit of this block are `v` and `z`. In order, apply the following optimizations to this basic block. Show the result of each transformation.

(a) algebraic simplification

(b) common subexpression elimination

(c) copy propagation

(d) constant folding

(e) dead code elimination

When you've completed part (e) the resulting program will still not be optimal. What optimizations, in what order, can you apply to optimize the result of (e) further?

3. Consider the following program.

```
L0: e := 0
    b := 1
    d := 2
L1: a := b + 2
    c := d + 5
    e := e + c
    f := a * a
    if (f < c) {
      goto L3
    }
L2: e := e + f
    goto L4
L3: e := e + 2
L4: d := d + 4
    b := b - 4
    if (b != d) {
      goto L1
    }
L5:
```

This program uses six temporaries, `a - f`. Assume that the only variable that is live on exit from this program is `e`.

Draw the register interference graph. (Drawing a control-flow graph and computing the sets of live variables at every program point may be helpful.)

4. Suppose that the following Cool program is executed.

```
class C {
  x : C; y : C;
  setx(newx: C) : C { x <- newx };
  sety(newy: C) : C { y <- newy };
  setxy(newx : C, newy : C) : SELF_TYPE { { x <- newx; y <- newy; self; } }; };
class Main {
  x : C;
  main() : Object {
    let a : C <- new C, b : C <- new C, c : C <- new C, d : C <- new C,
        e : C <- new C, f : C <- new C, g : C <- new C, h : C <- new C in {
      f.sety(g); a.setxy(e, c); b.setx(f); g.setxy(f, d); c.sety(h); h.setxy(e, a);
      x <- c;
 } }; };
```

(a) Draw the heap at the end of the execution of the program, identifying objects by the names to which they are bound in the `let` expression. Assume that the root is the `Main` object created at the start of the program, and that this object is not in the heap. If the value of an attribute is `void`, don't show that attribute as a pointer on the diagram.

(b) For each of the garbage collection algorithms discussed in class (Mark and Sweep, Stop and Copy, and Reference Counting), show the heap after garbage collection. When the pointers of an object are processed, assume that the processing order is the order of the attributes in the source program. Assume that the heap has space for at least 16 objects.