Assignment 3

(due Fri, 14/12/2012)

Compiler Design I (Kompilatorteknik I) 2012

DISCLAIMER: In the code listings of the following exercises we use a syntax that closely resembles C. However the C language allows neither nested function definitions (which appear in Exercise 2) nor true by-reference parameter passing (which is used in Exercise 1).

1 Parameter passing

Suppose we have the following program:

```
1  int spam(int x, int y) {
2    y = x + y;
3    return 2 * y;
4  }
5
6  int egg(int b) {
7    int a = 42;
8    return spam(a, b);
9  }
10
11 int main() {
12   int c = 100;
13   return c - egg(c);
14 }
```

Assume that the compiler allocates all the variables on the stack.

If the arguments of `spam` and `egg` are passed *by-reference*, answer the following questions:

1. Write the (MIPS) assembly code that will be used to retrieve the values of variables `x` and `y` before the addition is performed on line 2.
2. Write the (MIPS) assembly code that will be used to fill in the argument slots of the activation records for the calls on lines 8 and 13.

2 Activation records and scoping

In the following listing we show an excerpt of a program in a language that allows nested definitions for functions, is *statically scoped* and uses *call-by-value*:

```
1  int foo(int a) {
2    return 2 * a
3  }
4
5  int bar(int b, int c) {
6
7    int baz(int d, int e) {
8      int f, g;
9      f = 2 * e + b;
10     g = foo(f + d);
11     return g;
12   }
13
14   int h;
15
16   h = baz(2 * b, c);
17   return h + 42
18 }
```

Suppose we have in our `main()` function a call to `bar(1, 2)`. Assume that the compiler allocates all the variables (named and temporary) on the stack.

1. What will the return value of this call be?
2. Give a diagram of the state of the stack right before the return of the call to function `foo` on line 10. Your diagram should contain at least the following information regarding the calls to `foo`, `bar` and `baz` functions (including the initial call):
   (a) boundaries of the activation records
   (b) location of the actual parameters for the calls to these functions
   (c) location of the return values
   (d) location of the return addresses
   (e) location and contents of the control links and access links
   (f) *Optional:* Location of temporary variables
   (g) *Optional:* Contents of actual parameters
3. Show where the values of the following variables are located on the stack and describe the code that should be generated by the compiler to access these values in the specified locations (including the use of any points of reference, e.g. the stack pointer). You may use as many registers as you like to calculate any addresses that are needed:
   (a) Variable `f` on line 10
   (b) Variable `e` on line 9
   (c) Variable `b` on line 9

3 Code generation

In the examples presented on the lecture slides, the condition of an `if...then...else` expression is just one comparison for equality. Consider the case where we want to also allow composite boolean expressions that use just one `and` or `or` operator:

```
if (<e1> == <e2>) or (<e3> == <e4>) then <e5> else <e6>

if (<e1> == <e2>) and (<e3> == <e4>) then <e5> else <e6>
```

Write a code generation function for each of the two cases. Feel free to decide whether the operators will be short-circuiting or not.

4 Local optimizations

Consider the following basic block, in which all variables are integers:

```
k := x + y
i := k
h := x + y
j := h + i
l := 0 * b
r := l + j
f := j + r
```

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.

1. algebraic simplification
2. common sub-expression elimination
3. copy propagation
4. constant folding
5. dead code elimination

When you have completed part 5, the resulting program will still not be optimal. What optimizations, in what order, can you apply to optimize the result of 5 further?

5 Register allocation

Consider the following program.

```
while (d > 0) {
    a = b + c;
    d = d - b;
    e = a + f;
    if(e != 0) {
        f = a - d;
    } else {
        b = d + f;
        e = a - c;
    }
}
```

1. Construct the control-flow graph and calculate the set of live variables on entry to each statement, assuming that `b,c,d,e` and `f` are live on exiting the loop.
2. Compute the register interference graph for the program.
3. Colour the register interference graph using the minimum number of colours.

Instructions

As in Assignment 1 and 2, you are allowed to work in pairs. There are two ways to submit this assignment:

1. Submit a physical copy of your answers in my mailbox (Andreas Löscher, 99) on the 4th floor of building 1, opposite the 'fika' room.
2. Send an email with an electronic copy of your answers to `andreas.loscher@it.uu.se`.

Good luck!