CS164: Written Assignment 8
(On Dataflow Analysis and Register Allocation)

Assigned: Friday, Nov 12, 2004
Due: Wed, Nov 17, 2004, noon, in course mailbox.
The solutions will be posted a few hours after the homework is due.

Note: Thursday, Nov 18 is the day of the second Midterm Exam.

Note: The submission of this homework is optional, in the following sense: we will drop one score (the weakest one) of all of your homework scores. So, you may want to submit this homework if you failed at least one of the previous homeworks.

Grading and Submission

Your answers must be brief and easy to understand. Your grade (credit/no credit) will depend partly on how easy it is for us to understand and verify your answer. If the written assignment contains multiple questions or multiple parts, **you must answer all questions to receive a "pass" grade.** Minor mistakes are acceptable. Submit your written assignments either in the classroom (before the lecture) or in 283 Soda. *No late homeworks are accepted.* **Please indicate your login name and Section number.**

Dataflow Analysis and Register Allocation

Consider the following program:

```
do {
   a := b + c
   d := d - b
   e := a + f
   if (...)  {
        f := a - d
   } else {
        b := d + f
        e := a - c
        if (...)  break
   }
   b := d + c
} while (...); print(c + e)
```

1. Draw the control flow graph (CFG) for the program.

   Note that you do not need to write the *if* statements into the basic blocks; these statements are included in the source code merely to "define" the CFG edges.

2. Use global data flow analysis to compute the liveness information for variables `a` to `f`. Assume that no variables are live past the *print* statement.

   Annotate the solution of the analysis onto the CFG, by indicating on the entry and exit of each *statement* the set of live variables.

   (While we ask you to show only the final solution of the analysis, you should arrive at the solution by applying the rules given in class notes.)

3. Allocate five processor registers to the six program variables.

   - Draw the register interference graph for the program.
   - Apply the graph-coloring heuristic: Show one possible order in which you can remove the nodes from the graph using the graph-coloring heuristic we covered in class.

     Note: If you need to spill a variable, then (i) spill the variable with alphabetically lowest name (among those variables that are candidates for spilling); (ii) show the interference graph after spilling the variable; and (iii) show one possible order in which you can remove the nodes from the graph (after spilling).
   - Assign registers: for the node order specified above, assign a register to each graph node. When assigning a register to a node, always use the lowest-numbered register available. Use register names r_1 to r_5.
   - Finally, show the resulting program (with references to variables rewritten to references to registers). Initialize registers where needed (i.e., insert statements like $r_2 := a$).