Prof. Dr. Reinhard Wilhelm
Prof. Dr. Sebastian Hack
Dr. Daniel Grund
Michael Jacobs, M.Sc.

Compiler Construction WS11/12

Exercise Sheet 11

Please hand in the solutions to the theoretical exercises until the beginning of the lecture next Friday 2012-02-03, 12:00. Please write the number of your tutorial group or the name of your tutor on the first sheet of your solution.

Exercise 11.1 Code Generation (Points: 1+1+6+2)

Consider the integrated method of instruction selection and register allocation as introduced in the slide set about code generation (slides 15 to 25).

- Explain briefly why this approach for register allocation can be considered as local.
- For simplicity, the algorithm uses a variable's name to refer to the memory address at which the variable is stored. Unfortunately, the code generated by the algorithm on the slides does not generate correct assembler code for memory accesses. Point out the mistakes in the algorithm that cause this misbehavior and fix them.
- Use the algorithm to generate the assembler instructions for the following assignment statement:

  a := (a + b) − ((c + (d − e)) + (f + (b − g)))

  Assume that the target hardware only provides you the two registers R_1 and R_2. Your solution should contain the following things:

  - The expression tree for the assignment.
  - The register needs for the leaves and inner nodes of the assignment's right hand side part.
  - An outline of the generation phase that describes with which parameters the procedure Gen_Code is called in which order. For each call you should specify the current state of the register stack.
  - The complete assembler program generated from the assignment.
- Assume that we do no longer have to create two-address instructions in our assembler target language. This means that the target register and the first operand register of an assembler assignment do not have to be identical. Under this assumption we can simplify the case of the algorithm that is printed on slide 23. Find a simple modification of this case that allows you to remove the exchange statements completely.

Exercise 11.2 Interference Graphs (Points: 2+3)

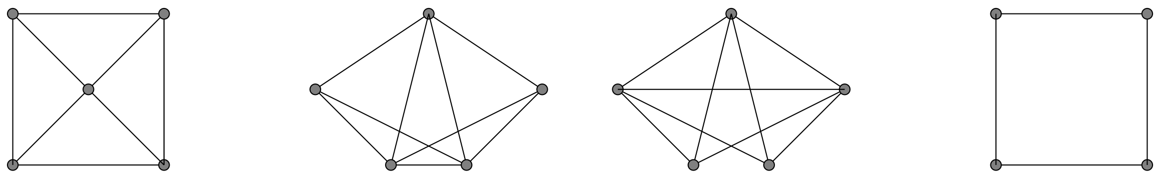
Consider the following code snippet.

```
s1 = 47;
s2 = 42;
s3 = s1 + s2;
do {
  s3 = s3 - s1;
  s4 = s3 + 2;
} while (s3 > s2);
s5 = s2 * s4;
s6 = s3 / s2;
write s6 - s5;
```

1. Draw the interference graph.
2. Assign actual registers to the symbolic registers by coloring the interference graph using Chaitin's local-colorability criterion. Assume an overall number of 4 registers to be available.

Exercise 11.3 Chordal Graphs (Points: 4)

Are the following graphs chordal or not? Justify your claims!



Exercise 11.4 Colorability (Bonus points: 6)

Show that Chaitin's algorithm finds a k-coloring for every k-colorable chordal graph.
Hint: Every chordal graph with at least 2 nodes has 2 simplicial nodes.