Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

Summer Semester 08
Vesal Vojdani
vojdanig@in.tum.de

Compiler Construction & Virtual Machines

Exercise Sheet 1

Deadline: 23. April 2008, 12:00, during the lecture or in room 02.07.053

Exercise 1: *Code generation* — *8 Points*

Consider the following instruction sequence.

```
z = 1;
while (n > 0) {
    j = 1;
    y = x;
    while (2 * j <= n) {
        y = y * y;
        j = j * 2;
    }
    z = y * z;
    n = n - j;
}
```

a) What does the instruction sequence compute?

b) Generate CMa code for the instruction sequence.
Use the address environment $\rho = \{n \mapsto 0, j \mapsto 1, x \mapsto 2, y \mapsto 3, z \mapsto 4\}$!

Exercise 2: *Registers* — *12 Points*

We extend the CMa machine by adding an unbounded number of registers $R_0, R_1, \ldots$. To improve efficiency, expressions are evaluated by storing intermediate values in registers instead of on the stack. For example, to evaluate $x * y + 2$ and to store the result in R_{42}, we first put the address of x in R_{42}, put $S[R_{42}]$ in R_{42}, put the address of y in R_{43}, put $S[R_{43}]$ in R_{43}, put $R_{42} * R_{43}$ in R_{42}, put 2 in R_{43} and then put $R_{42} + R_{43}$ in R_{42}.

a) Choose a set of new CMa instructions for doing this translation.

b) Give the translation scheme for evaluation of expressions and assignment statements. For this purpose, extend the functions $code_L$ and $code_R$ to now take an additional argument i, such that R_i is the register in which to store the result of evaluation. Think of i as a static stack pointer: we assume the invariant that all registers R_j with $j \geq i$ are free for our use in evaluating the given expression. The generated code should produce instructions involving concrete registers, such as R_7 and R_{42}, not R_{R_0} or R_{SP}.