Technische Universität München *Summer Semester 08*
Fakultät für Informatik *Vesal Vojdani*
Prof. Dr. H. Seidl *vojdanig@in.tum.de*

Compiler Construction & Virtual Machines

Exercise Sheet 3

Deadline: 5. May 2008, at the lecture, in room 02.07.053, or by e-mail.

Exercise 1: *Variable initialization* *6 Points*

Extend the translation function *code* to initialization constructions, such as `int x = 5`. If no initializer is given, then you should set the variable to its null value, i.e., set the entire allocated area to zeroes. Structs and arrays initializers are given as a list of constant expressions, e.g., `int a[3] = {1, 2, 4}`.

Exercise 2: *Break and Continue* *4 Points*

Modify the scheme for translation of loops, to take care of the `break` statement, for the immediate termination of a loop, and `continue` statements for skipping to the next iteration of the loop. For this purpose, you may wish to extend the code generation function to take a further argument, a pair of labels (l_b, l_c), which characterize the labels that exit and continue the loop. Your solution need only give the translation function for one of the looping constructs, e.g., the `while`-statment.

Exercise 3: *Code generation: Function calls* *8 Points*

Generate the code for the following. Make sure it runs on the CMa of the VAM system. It should run for a fairly long time, 111 recursive calls. Send the solution by e-mail as a nicely commented .cma file.

```
int f(int n) {
  if (n % 2 == 0) return n/2;
  else return (3*n + 1);
}

int c(n) {
  if (n <= 1) return 1;
  else return c(f(n));
}

int main() {
  c(27);
}
```

Exercise 4: *Call-by-name* *6 Points*

Change the way paramters, even integers, are passed to procedures to adopt a call-by-name translation scheme. This is how reference types in C++ and non-primitive types in Java are handled. Consider the following example:

```
int x;
int f(int i) {
  i = 27;
}

int main() {
  f(x);
}
```

The value of x should be 27 after the function call.

a) Give the translation-scheme for function calls.
b) Write a VAM executable .cma file for this example using your new translation function.

Exercise 5: *Varargs* *No Points*

Try to come up with a scheme to translate *variadic* functions, such as the one in the following piece of C code.

```
#include<stdio.h>
#include<stdarg.h>

double avg(int n, ...) {
  double sum = 0;
  int i = 0;
  va_list e;

  va_start(e, n);
  while (i++ < n)
    sum += va_arg(e, int);
  va_end(e);
  return (sum/n);
}

int main() {
  printf("%f\n", avg(3, 1,3,7));
  return 0;
}
```

This is for discussion during the lab, it is not graded.