Compiler Construction WS07/08

Solution Sheet 1

1 Static vs. Dynamic Information

- addresses of variables: both static derived and dynamic.
  - static derived: global variables, offset to stack pointer for local variables
  - dynamic: absolute position of local variables
- scopes of variables: static primary, because they can be read off the program text.
- sizes of structures: static derived, because the size of a structure depends on the final memory layout (padding), which is done by the compiler.
- values of pointers: dynamic
- types of variables: static primary, which corresponds to scopes of variables.

2 Statements and Expressions

- \textbf{loada} \langle\text{address of } i\rangle
  \textbf{dup}
  \textbf{loadc} 1
  \textbf{add}
  \textbf{storea} \langle\text{address of } i\rangle
  \textbf{pop}
  \textbf{pop}

- if \( y \) is a global variable: \textbf{loadc} \langle\text{address of } y\rangle
  if \( y \) is a local variable: \textbf{loadrc} \langle\text{address of } y\rangle

- \textbf{HEAD}: \quad \textit{code}_R \ v_1 \ \rho
  \quad \textbf{jumpz} \ \text{END}
  \quad \textit{code} \ s_1 \ \rho
  \quad \textit{code}_R \ v_2 \ \rho
  \quad \textbf{not}
  \quad \textbf{jumpz} \ \text{HEAD}
  \quad \textit{code} \ s_2 \ \rho
  \quad \textbf{jump} \ \text{HEAD}

- \textbf{END}: \quad \ldots
• HEAD:  codeR e1 ρ
    jumpz END
    code s ρ
    codeR e2 ρ
    not
    jumpz END
    jump HEAD
END:  ... 

3 Code Generation for CMa

; push result and i on stack
loadc 1
loadc 1

; loop head
.HEAD
loada 2
loadc 10
leq
jumpz .END

; compute result = result*i
loada 1
loada 2
mul
storea 1
pop

; increment i
loada 2
loadc 1
add
storea 2
pop
jump .HEAD

; pop i from stack
.END
pop