

Cobo - Assignment 6

S.R.W. van Kampen en R.A.J. Wacanno

March 22, 2020

6.1 Code Generation

a, b)

Below is the manually-generated assembly, with the assembly below a given comment line pertaining to that line in the source. Comments are indicated with semicolons.

```
; int factorial (int x) {
factorial:
; int res;
    esr 1                ; enter subroutine with 1 local variable
; if (x <= 1) res = 1;
    iload 0              ; push argument variable x on the stack
    iloadc_1             ; push int value 1 on the stack
    ile                 ; apply <= on first two stack items
    branch_f if_elseblock ; jump to else block if results is false
    iloadc_1             ; push int value 1 on the stack
    ireturn              ; return the top of the stack
; else res = x * factorial(x - 1);
if_elseblock:
    iload 0              ; push argument variable x on the stack
    isrg                 ; initiate global subroutine
    iload 0              ; push argument variable x on the stack
    iloadc_1             ; push int value 1 on the stack
    isub                 ; apply - on first two stack items
    jsr 1 factorial      ; jump to subroutine with 1 arg variable
    imul                 ; apply * on first two stack items
; return res;
    ireturn              ; return the top of the stack
;}
```

c)

```
factorial:                ; number of bytes:
    esr 1                  ; 2
    iload 0                ; 2
    iloadc_1               ; 1
    ile                    ; 1
    branch_f if_elseblock ; 3
    iloadc_1               ; 1
    ireturn                ; 1
if_elseblock:
    iload 0                ; 2
    isrg                   ; 1
    iload 0                ; 2
    iloadc_1               ; 1
    isub                   ; 1
    jsr 1 factorial        ; 4
    imul                   ; 1
    ireturn                ; 1
```

```

factorial:
    esr 1
    iload 0
    iloadc_1
    ile
    branch_f 5
    iloadc_1
    ireturn
if_elseblock:
    iload 0
    isrg
    iload 0
    iloadc_1
    isub
    jsr 1 -18
    imul
    ireturn

```

6.2 Compilation Schemes Revisited

We define two compilation schemes, one main and one auxiliary, the main initialises the auxiliary scheme with a prefix of a single underscore. Assuming this has a single scope, the prefixes ensure no variables are redefined.

$$\mathcal{C}[\textit{Code}] \Rightarrow \mathcal{CX}[\textit{Code}][_]$$

$$\mathcal{CX} \left[\begin{array}{l} \textit{for} (\textit{int } \textit{Var} = \textit{Start}, \textit{Lim}) \\ \{ \textit{Body} \} \\ \textit{Rest} \end{array} \right] \left[\left[\textit{Prefix} \right] \right] \Rightarrow \begin{array}{l} \textit{int } \textit{PrefixVar} = \textit{Start}; \\ \textit{while} (\textit{PrefixVar} < \textit{Lim}) \\ \{ \\ \quad \mathcal{CX}[\textit{Body}][_i_Prefix] \\ \quad \textit{PrefixVar} = \textit{PrefixVar} + 1; \\ \} \\ \mathcal{CX}[\textit{Rest}][_o_Prefix] \end{array}$$