TDT4205 Compiler Construction
Assignment/Problem Statement 4

1 Theory 30%

The Factorial sequence is defined so that
$f(0) = 1$
$f(1) = 1 * f(0)$
$f(2) = 2 * f(1)$
$f(n) = n * f(n-1)$
Write an x86 64 assembler program that calculates f(8) and prints it on standard output.

2 Programming 70%

The VSL compiler in the provided archive is extended with a function 'generate program' in generator.c; this function is called from main.c, after syntax tree and symbol table construction. Implement this function so that it generates x86 64 assembly code for the following constructs:

2.1 Global string table

Strings should be given numbered labels in a data segment.

2.2 Global variables

Global variables should be given names corresponding to their declarations, prefixed with an underscore character ' ', so as to avoid names that clash with names from the system libraries.

2.3 Functions

Functions should be placed in the text segment, named in the same manner as global variables, and set up/remove a stack frame. Furthermore, they should initiate a recursive traversal of their syntax subtrees, so that the remaining constructs can be generated.

2.4 Function parameters

Function parameters should be expected to follow the standard calling conven- tion covered in lectures. Copies can be placed at the bottom of the function's stack frame, to make their run-time address computable from their sequence number, and liberate the registers for further function calls.

2.5 Arithmetic expressions

Arithmetic expressions should be translated so as to leave their result in the RAX register, and remove any intermediate calculations from the generated program's run time stack.

2.6 Assignment statements

Assignment statements should copy the result of an expression to the address of the assigned variable.

2.7 PRINT statements

PRINT statements can be translated into a sequence of 'printf' calls, with one call per item in the PRINT statement's list.

2.8 RETURN statements

RETURN statements should leave the result of their expression in the RAX register, remove the function's stack frame, and return control to the caller.

2.9

Implementing the following constructs in generator.c:

1. Local variables
2. Function calls

3. Conditionals (IF and relations)
4. While loops
5. Continue (null statement)