Práctica 4

Part 2: Compiling the Other Statements

The If Statement: let us start by considering the if statement which has the form:

if <test> then <then> else <else>

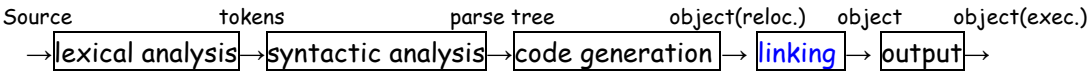
The code for this take the form:

```
        testcode;
        thencode;
        JUMP label2;
label1:
        elsecode;
label2:
```

where testcode causes a jump to label1 if the test proves false. We have used labels to indicate the instructions whose addresses are label1 and label2. The Prolog formulation of this is:

```
encodestatement(if(Test,Then,Else),D,
                (Testcode; Thencode; instr(jump,L2); label(L1); Elsecode; label(L2)) ) :-
        encodetest(Test,D,L1,Testcode),
        encodestatement(Then,D,Thencode),
        encodestatement(Else,D,Elsecode).
```

Notice that the clause does not fix the addresses L1 and L2 but only constraints its values through labelinf the object code. One can think of the output from 'encodestatement' as being relocatable code. The output will contain free variables L1 and L2 whose values will not be fixed until the linking stage of compilation.



This is an example of the use of the logical variable to delay specifying certain parts of a data structure. In the code for encodetest(Test,D,L,Testcode), The test has the form test(Op,Arg1,Arg2) and L is the address to jump to if the test fails. Op is either =, < or >.

Thus, the definition of the predicate encodetest is something as follows:

encodetest(test(Op,A1,A2),D,L, (Ecode; instr(J,L))) :- …

The While Statement: Write the code for compiling the while statement.

The Read and Write Statement: write the code for the definition handling the read (of a variable) and the clause handling the write (of an expression) statement. Assume the instruction instr(write,0) outputs the data in the accumulator into the screen. This is:

encodestatement(read(name(X)),D, instr(read,Addr)) :- lookup(…).
encodestatement(write(Expr),D, (Ecode; instr(write,0))) :- encodeexpr(…).

The Sequence Statement: write the code for handling the sequence statement

Stat1; Stat2

which which is treated as just another statement type. This is write code defining the clause

encodestatement((S1;S2),D, (Code1;Code2)) :- …

Target Language

Remember the target language instructions are as follows:

Arithmetic literal: ADDC, SUBC, MULC, DIVC, LOADC
Arithmetic memory: ADD, SUB, MUL, DIV, LOAD, STORE
Control transfer: JUMPEQ, JUMPNE, JUMPLT, JUMPGT, JUMPGE, JUMP
Input output: READ, WRITE, HALT.

For an example (to compute factorials) of the compiler function see Practica 3.
Sample outputs of the program so far may look like this:

```
?- encodestatement(if(test(=,name(x),const(5)), assign(name(x),const(1)), assign(name(x),const(2))),D,X).
D = dic(x, _G921, _G925, _G926)
X = ((instr(load, _G921);instr(subc, 5));instr(jumpne, _G897)); (instr(loadc, 1);instr(store, _G921));instr(jump,
_G892);label(_G897); (instr(loadc, 2);instr(store, _G921));label2(_G892)

?- encodestatement(while(test(=,name(x),const(5)),assign(name(x),expr(+,name(x),const(1)))),D,X).
D = dic(x, _G878, _G882, _G883)
X = lable(_G845); ((instr(load, _G878);instr(subc, 5));instr(jumpne, _G859)); ((instr(load, _G878);instr(addc,
1));instr(store, _G878));instr(jump, _G845);label(_G859)

?- encodestatement((read(name(x));write(expr(+,name(x),const(1)))),D,X).
D = dic(x, _G682, _G686, _G687)
X = instr(read, _G682), ((instr(load, _G682);instr(addc, 1));instr(write, 0))
```

Submitting your answer

The práctica can be solved in teams of two people (1 submission for team). Submission is by email (to rafael.ramirez@upf.edu), the subject of the email must be *'Practica 4- nombre1 apellido1-nombre2 apellido2*, and the message body should contain the programs for 1) the exercises and 2) some tests to your programs. Deadline is the exactly ONE week after the session of this práctica. Late submissions may have a penalty.