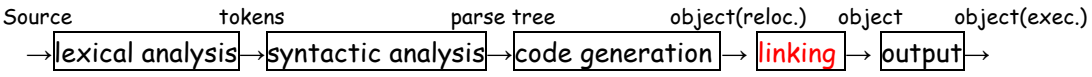
Práctica 5

Compiling the Whole Program and the Assembly Stage

This práctica is a continuation of the previous prácticas. Here you need to translate the complete program into an object program with **absolute** addresses. i.e. you have to write Prolog code to implement the **linking** component of the diagram:



The translation of the complete source program may be expressed by the following clause:

```
compile(Source, (Code; instr(halt,0); block(L)) ) :-
    encodestatement(Source,D,Code),
    assemble(Code,1,N0),
    N1 is N0+1,
    allocate(D,N1,N),
    L is N-N1.
```

The result of compiling the program **Source** is a sequence of instructions **Code** followed by a HALT instruction and then a block of storage for the variables used in **Source**. Stage 3 of the compilation (the code generation component of the diagram) is represented by the goal

encodestatement(Source,D,Code).

At the end of stage 3, **Code** still contains many free variables (representing the yet to be specified addresses of writeable locations and labeled instructions). The goal

assemble(Code,1,N0)

computes the addresses of labeled instructions and returns **N0**, the address of the end of **Code**. **N1** is therefore the address of the start of the block of storage locations. The goal

allocate(D,N1,N)

is responsible for laying out the storage required for the source language symbols contained in the dictionary D. It fills the corresponding addresses and returns N, the address of the end of the storage block. Finally, the length L of the storage block is calculated as N-N1.

Write code for assemble/3 and allocate/3. Note that **assemble(X,Y,Z)** means that **Y** is the start address and **Z** the end address of the sequence of instructions **X**.

First, in order to avoid several solutions for the dictionary, modify (add a cut to) the first clause of your definition for lookup to:

```
Lookup(Name,dic(Name,Value,_,_),Value) :- !.
```

For writing the code for 'assemble' notice that there are three cases:

```
assemble((Code1;Code2),N0,N1) :- …      % the code to assemble is a sequence of things
assemble(instr(_,_),N0,N) :- …          % the code to assemble is a single instruction
assemble(label(N),N,N).                 % the code to assemble is a label
```

The code for the procedure 'allocate' has a similar character:

```
allocate(void,N,N) :- !.
allocate(dic(Name,N1,Before,After),N0,N) :-
        allocate(Before,…),
        N2 is …
        allocate(After,…).
```

Sample outputs of the program may look like this (the last one is abbreviated by Prolog but using the graphical debugger you can see the detail):

```
?- compile(if(test(=,name(x),const(5)),assign(name(x),const(1)), assign(name(x),const(2))),C).
C = (((instr(load, 10);instr(subc, 5));instr(jumpne, 7)); (instr(loadc, 1);instr(store, 10));instr(jump,
9);label(7); (instr(loadc, 2);instr(store, 10));label(9));instr(halt, 0);block(1)

?- compile(while(test(=,name(x),const(5)),assign(name(x),expr(+,name(x),const(1)))),C).
C = (label(1); ((instr(load, 9);instr(subc, 5));instr(jumpne, 8)); ((instr(load, 9);instr(addc, 1));instr(store,
9));instr(jump, 1);label(8));instr(halt, 0);block(1)

?-
compile((read(name(v));assign(name(c),const(1));assign(name(r),const(1));while(test(<,name(c),name(v)),
assign( name(c),expr(+,name(c),const(1)));assign(name(r),expr(*,name(r),name(c))));write(name(r))),C).
C = (instr(read,21) ; (instr(loadc,1) ; instr(store,19)) ; (instr(loadc,1) ; instr(store,20)); (label(6) ;
(((instr(load,19) ; instr(sub,21)) ; instr(jumpge,16)) ; ((((instr(load,19) ; instr(addc,1)) ; instr(store,19)) ;
((instr(load,20) ; instr(mul,19)) ; instr(store,20))) ; (instr(jump,6) ; label(16))))) ; instr( load,20) ;
instr(write,0)) ;instr(halt, 0);block(3)
```

Implement a predicate to display your output in a more friendly way.

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 5- 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 Friday 16 March.