Formal Languages and Compilers
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Written exam[1]: laboratory question
28/06/2008

SURNAME: .....
NAME: ..... Student ID: .....
Course: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Other:.....
Instructor: ○ Prof. Breveglieri ○ Prof. Crespi ○ Prof Morzenti

The laboratory question must be answered taking into account the implementation of the `Acse` compiler given with the exam text.

Modify the specification of the lexical analyzer (`flex` input) and the syntactic analyzer (`bison` input) and any other source file required to extend the `Lance` language with the ability to handle a new iterative construct *computed goto* resembling the one in the following sample.

```
int a;
...
label3: ...
...
goto a in label1,label2,label3;
...
label1: ...
...
label2: ...
...
```

The semantic of the construct is the following one: if the runtime value of the control variable (`a` in the example) is a positive integer n , then the goto structure jumps to the n-th label after the keyword `in` (e.g.: if $a = 1$ the control flow jumps to "label1" in the aforementioned code). If the control variable assumes either a value less than 1, or a value greater than the number of labels, no alterations in the control flow are made and the program continues its execution.

Your modifications have to allow the `Acse` compiler to both correctly analyze the syntactical correctness of the aforementioned construct and generate a correct translation in the `Mace` assembly language.

In order to correctly solve the question we suggest you to use the `getNewRegister` function contained in `axe_engine.[h,c]`, which is able to reserve a free register for your purposes. The prototype of the function is the following one.

[1] Time 45'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

```
/* get a register still not used. This function returns
 * the ID of the register found */
int getNewRegister(t_program_infos *program);
```

The integer returned by the function represents, in a non ambiguous way, the register which has been allocated.

1. Define the tokens and the Acse.lex and Acse.y declarations needed to achieve the required functionality. (3 points)

2. Define the syntactic rules needed to achieve the required functionality. (8 points)

3. Define the semantic actions needed to achieve the required functionality. (17 points)

4. Define the semantic actions needed to raise an error if there are undeclared labels within the goto statement. (5 points)