Formal Languages and Compilers
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Written exam[1]: laboratory question
05/02/2008

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 *foreach* resembling the one in the following sample.

```
int a[100];
int x;
int y;
...
for(x:a){
   y = y + x;
}
write( y );
...
```

This kind of construct doesn't explicitly point out the index of the vector. On the first iteration, x takes the value of a[0], on the second one is updated to a[1], on the third is updated to a[2] and so on for each element of the array.

Your modifications have to allow the Acse compiler to both correctly analyze the syntactical correctness of the aforementioned constructs and to 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.

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

1. Define the tokens and the Acse.lex and Acse.y declarations needed to achieve the required functionality.

2. Define the syntactic rules needed to achieve the required functionality.

3. Define the modifications to the data structures needed to achieve the required functionality.

4. Define the semantic actions needed to achieve the required functionality.

5. **Bonus** (to be solved only if you have already finished the previous points).Forbid any assigment to the induction variable (`x`, in the aforementioned example)in the loop body.