# Formal Languages and Compilers
# Proff. Breveglieri, Morzenti
# Written exam[1]: laboratory question
# 19/07/2017

SURNAME: ..........................................................................
NAME: ................................................. Student ID: ................
Course: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Other: ...
Instructor: ○ Prof. Breveglieri ○ 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 analyser (`flex` input) and the syntactic analyser (`bison` input) and any other source file required to allow the support for a software emulated multiplication operation in `Lance`
A software emulated multiplication operation should have the same functional behavior of a regular multiplication, save for the fact that it *no MUL operations* are emitted by ACSE to handle it. The software emulated multiplication employs the operator `[*]` instead of the common `*`. The proposed implementation should perform proper constant folding, taking into account the fact that ACSE will run on a machine which *has* support for multiplication. Therefore, the * may coexist with the operator [*], as in the following example.

```
int a=3,b=2;
if (a*b == a [*] b) {
 write (1);
} else {
 write(0);
}
```

The software emulated multiplication operator has the same precedence of the common multiplication and must provide the same semantics (proper treatment of signs included).

[1]Time 60'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (1 point)
2. Define the syntactic rules or the modifications required to the existing ones. (2 points)
3. Define the semantic actions needed to implement the required functionality. (18 points)
   The solution is in the attached patch.

4. Provide the Flex compliant description of a lexical translator which scans a text composed of a sequence of ASCII characters from stdin and

   (a) Replaces all the uppercase letters enclosed in square brackets with their corresponding lowercase ones

   (b) Replaces all the letters enclosed in curly braces with their corresponding ASCII code, printed as a decimal number

   (c) Leaves all the remaining characters untouched

   You may assume that no brackets nesting of any kind appears in the input. Any character which is not a letter is printed in output without change. For example, the string `This [StRING] {Is} processed` becomes `This [string] {73115} processed`. (7 points)

   The solution is provided in the attached lexer.l file.

5. (**Bonus**) Describe how the approach proposed to introduce the software emulated multiplication can be modified to provide an efficient (constant time) single machine word multiplication. To this end, recall that the `mace` machine is a 32 bit architecture.