Table 1 < Grammar >

```
prog      ::=   word "("   ")" block ;
decls     ::=   decls decl
            | ;
decl      ::=   vtype word ";" ;
vtype     ::=   int | char
            | ;
block     ::=   "{" decls slist "}"
            | ;
slist     ::=   slist stat
            |   stat ;
stat      ::=   IF cond THEN block ELSE block
            |   word "=" expr ";"
            |   EXIT expr ";"
            | ;
cond      ::=   expr ">" expr ;
expr      ::=   expr "+" fact
            |   fact ;
fact      ::=   num
            |   word ;
word      ::=   ([a-z] | [A-Z])* ;
num       ::=   [0-9]*
```

Table 2 < instruction set>

LD Reg#1, addr(or num)	Load var (or num) into the Reg#1
ST Reg#1, addr	Store value of Reg#1 into var
ADD Reg#1, Reg#2, Reg#3	Reg#1 = Reg#2 + Reg#3
LT Reg#1, Reg#2, Reg#3	1 if (Reg#2 < Reg#3), 0 otherwise , store into Reg#1
JUMPF Reg#1 label	Jump to label if Reg#1 contains 0
JUMPT Reg#1 label	Jump to label if Reg#1 contains Non-0
JUMP label	Jump to label without condition