TAC Grammar for A2

LEXER TOKENS

```
"retval"        -> RETVAL
"global"        -> GLOBAL
[ \t\n]         -> (ignore whitespace)
[!><=]=         -> RELOP
=               -> EQ
"("             -> OP
")"             -> CP
"goto"          -> GOTO
"if"            -> IF
"return"        -> RETURN
"param[0-9]+"   -> PARAM
"call"          -> CALL
\"[^"]*\".      -> STR

"t[0-9]+"       -> TEMPORARY // Temporary variables like t1,  t2, etc.

"L[0-9]+:"                  -> LABEL // Labels like L1:, L2:, etc.

"L[0-9]+"                    -> GOTO_LABEL // Goto labels like L1, L2..

[a-zA-Z_][a-zA-Z_0-9]*:     -> F_IDENTIFIER // Function identifiers
                               ending  with ':'
[a-zA-Z_][a-zA-Z_0-9]*      -> IDENTIFIER // General identifiers for
variables/functions
[0-9]+                      -> NUMBER // Numeric constants
[+\-*/%]                    -> AOP
"&&"                        -> AND
"||"                        -> OR
.                           -> (error, unexpected character)
```

GRAMMAR

```
------------------
start : program
-------------------------
program : globals functions
;
-------------------------
globals : globaldecl globals
| ;
-------------------------
globaldecl : GLOBAL IDENTIFIER
| TEMPORARY EQ NUMBER
| TEMPORARY EQ TEMPORARY
| IDENTIFIER EQ TEMPORARY
;
-------------------------
functions : F_IDENTIFIER decls RETURN  | ;
-------------------------
decls : paramdecls fundecls retvaldecl  | ;
-------------------------
paramdecls : paramdecl paramdecls
| ;
-------------------------
paramdecl : IDENTIFIER EQ PARAM
-------------------------
fundecls : fundecl fundecls
| ;
-------------------------
fundecl : assignmt
| func_call
| if_statement
| GOTO GOTO_LABEL
| LABEL
-------------------------
assignmt : direct
| indirect
-------------------------
direct : IDENTIFIER EQ TEMPORARY
| TEMPORARY EQ NUMBER
| TEMPORARY EQ TEMPORARY
| TEMPORARY EQ IDENTIFIER
| TEMPORARY EQ STR
;
```

```
-------------------------
indirect : TEMPORARY EQ expressions
-------------------------
expressions : expression expressions | ;
-------------------------
expression : logical_expression
           | arithmetic_expression
           | relational_expression
           ;
-------------------------
logical_expression
    : TEMPORARY logical_expression_tail
    ;
-------------------------
logical_expression_tail
    | AND TEMPORARY  // AND represents '&&' in TAC
    | OR TEMPORARY   // OR represents '||' in TAC
    | ;
-------------------------
arithmetic_expression
    : TEMPORARY arithmetic_expression_tail
    ;
-------------------------
arithmetic_expression_tail
    | AOP TEMPORARY
    | ;

// AOP represents binary arithmetic operators (+, -,  *, /, %)
-------------------------
relational_expression
    : TEMPORARY relational_expression_tail
    ;
-------------------------
relational_expression_tail
    | RELOP TEMPORARY
    | ;

// RELOP represents relational operators (==, !=, <, >, <=, >=)

-------------------------
if_statement : IF OP TEMPORARY CP GOTO GOTO_LABEL
-------------------------
func_call   : func_params CALL LABEL
            ;
-------------------------
func_params : func_param func_params | ;
```

```
--------------------------

func_param  : PARAM EQ TEMPORARY
            | PARAM EQ IDENTIFIER
            | ;
--------------------------
retvaldecl  : RETVAL EQ TEMPORARY
            | RETVAL EQ IDENTIFIER
            ;
```