**ASSIGNMENT DOCUMENTATION**

**COMPILER CONSTRUCTION**

Hafiz Ahmad Hassan 15L-4359

Maryam Nadeem 15L-4107

**LEXICAL ANALYZER**

**LANGUAGE TOKENS**

* **Data types**: there are two data types recognized by the lexical analyzer int and char

1. ( int , kw )
2. ( char , kw )

* **Keywords**: there are six keywords recognized by the lexical analyzer and those are if, else, while, ret, in, and out

1. ( if , kw)
2. ( else , kw )
3. (while , kw)
4. ( ret , kw )
5. ( in , kw )
6. ( out , kw )

* **Arithmetic operators**: there are four arithmetic operators recognized by the lexical analyzer + , ‐ , \*, /

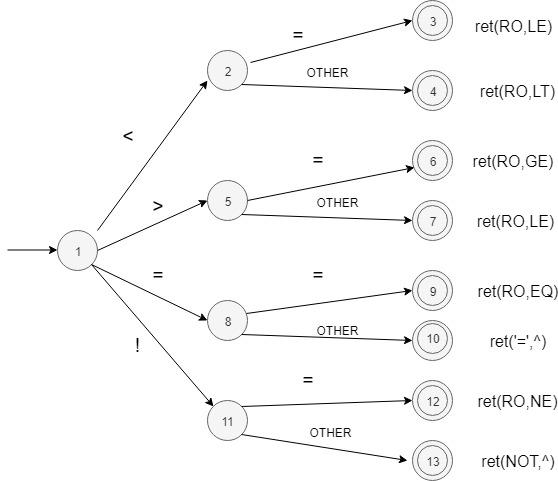
1. ( ‘+’ , ^)
2. ( ‘-‘ , ^ )
3. ( ‘\*’ , ^)
4. ( ‘/’ , ^)

* **Relational operators**: there our four relational operators in the lexical analyzer <, <=, >, >=, ==, !=

1. ( RO , <)
2. ( RO , <= )
3. ( RO , > )
4. ( RO , >=)
5. ( RO , ==)
6. ( RO , !=)

Regular definition: <| <= |> | >= | ==| !=

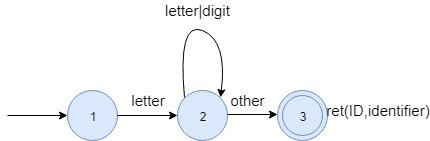
Transition Diagram:



* **Line comments**: (‘ # ’ , ^ )
* **Multi line comments**: (‘ ## ’ , ^ )
* **Identifier**: ( ID, identifier)

Regular definition: letter: a|b|…|z|A|B|…|Z| digit: 0|1|2|…|9 identifier: letter (letter|digit)\*

Transition Diagram:



* **Numeric constant:** (NUM, numeric constant)
* **Constants**: (CONST , constant)

Regular definition: letter: a|b|…|z|A|B|…|Z| Constant: ‘letter’

Transition Diagram:



* **Strings**: (STR,string)
* **Parenthesis:** ( ‘ ( ‘ , ^ ) ( ‘ ) ‘ , ^ )
* **Braces:** ( ‘ ( ‘ , ^ ) ( ‘ ( ‘ , ^ )
* **Square brackets:** ( ‘ [ ‘ , ^ ) ( ‘ ] ‘ , ^ )
* **Assignment operator:** ( ‘ := ’ , ^ )
* **Semi colon:** ( ‘ ; ’ , ^ )
* **Comma:** ( ‘ , ’ , ^ )

**PARSER**

**A1 =[sdt.addString(myLex());]**

Declaration -> Function|FunctionCall

FunctionCall -> id **[sdt.n['t']=myLex();]** AssignOp [ **sdt.n['s']= myLex();**]E

Fucntion -> FunctType id ( Parameter ) { Statements }

FunctType-> Type | ^

Parameter -> One More |^

One -> Type id

More -> , Type id More | ^

DeclareVariable -> Type VarId **A103[ sdt.n['o']=sdt.getStored();]** ;**A102[sdt.emit(sdt.n['o']);]**

VarId -> id **A1** MoreVar

MoreVar-> , id MoreVar | ^

InitializeVariable -> id **[A1]** AssignOP **[A1]** IniVar **A100[sdt.n[u]=getStored()]** ; **A101[emit [sdt[u]]**

IniVar -> id **[A1]** | Integer **[A1]**|LiteralConstant**[A1]** | E [AssignStored(E.n)]

OUT -> out **[A1]**OutType **A103[sdt.n[o]=getStored()]** ; **A102[emit [sdt[o]]**

OutType -> String **[A1]**| id **[A1]**|intege**r[A1]** | LiteralConstant**[A1]**

RET -> ret **[A1]** RetType **A103 ;A102**

RetType -> id**[A1]** | integer**[A1]** | LiteralConstant**[A1]**

IN -> in **A1** id **A1 A103[ sdt.n['o']=sdt.getStored(); ] ;A102[ sdt.emit(sdt.n['o']);]**

E -> T **[A10] sdt.n['R']=sdt.n['T'];**

R **[A44sdt.n['E']=sdt.n['R'];**

**sdt.storage=s[0]+"= "+sdt.n['E'];]**

R -> + T [**A11 sdt.emit(makeVar() + " = " + sdt.n['R'] + “+ ”+ sdt.n['T']);**

**sdt.n['R']= "t"+(vid-1);**

] R

R -> - T **[** **A11 sdt.emit(makeVar() + " = " + sdt.n['R'] + “- ”+ sdt.n['T']);**

**sdt.n['R']= "t"+(vid-1);** ]R

T -> F **[A9 sdt.n['T']=sdt.n['F'];]**

R’

R’ -> \* F **sdt.emit(makeVar() + " = " + sdt.n['R'] + “\* ”+ sdt.n['T']);**

**sdt.n['R']= "t"+(vid-1);** R’ | / F **sdt.emit(makeVar() + " = " + sdt.n['R'] + “/ ”+ sdt.n['T']);**

**sdt.n['R']= "t"+(vid-1);**

R’ F -> id **A21 [ F.n = id.lex]**

| integer | ( E )|id(OAL) **[ sdt.emit("call " + sdt.n['i']+" " +sdt.n['s'] + sdt.count +" " +makeVar());**

**sdt.emit(sdt.n['t']+" := " +" "+" t" + String.valueOf(vid-1));]**

OAL -> AL

AL->E AL’ **[E.n = id.lex ,sdt.count=1]**

AL’->,E AL’ **[emit (“param”+sdt.n[‘E’]) sdt.count=sdt.count+1]**

WHILE -> while ( Con ) **A401 [ sdt.backPath(Integer.valueOf(sdt.n['x'])-1, sdt.line);]** ConStatement **A402[ sdt.emit("goto" +" "+String.valueOf(Integer.valueOf(sdt.n['x'])-1));**

**sdt.backPath(Integer.valueOf(sdt.n['y'])-1, sdt.line);**

**]**

IF -> if ( Con )

**A106[ sdt.backPath(Integer.valueOf(sdt.n['x'])-1, sdt.line);]**

ConStatement

**A6 [ sdt.emit("goto"); sdt.n['z']= String.valueOf(sdt.getLine());]**

| if ( Con )  **A106[sdt.backPath(Integer.valueOf(sdt.n['x']), sdt.line);** ] ConStatement

**A6[ sdt.emit("goto"); sdt.n['z']= String.valueOf(sdt.getLine());]** else **A70[sdt.backPath(Integer.valueOf(sdt.n['y'])-1, sdt.line);]** [ **sdt.backPath(Integer.valueOf(sdt.n['y']), sdt.line); ]** ConStatment **A71[ sdt.backPath(Integer.valueOf(sdt.n['z']), sdt.line); ]**

Con -> P RO P  **A105[sdt.emit("if" + sdt.getStored() + " goto "); sdt.n['x']=String.valueOf(sdt.getLine());sdt.emit("goto");sdt.n['y']=String.valueOf(sdt.getLine());]**

P -> id A1 | integer A1 | LiteralConstant A1 | ( Con )

ConStatement -> S | ManyStatements

S -> DeclareVariable | Initialize Variable |E | OUT | RET | IN

ManyStatements -> { Statements }

Statements -> Statement Statements

Statement -> DeclareVariable | Initialize Variable |E | OUT | RET | IN | WHILE | IF

Type -> int A1| charA1

FC-> id AssignOP id(ME);

ME->,ME