<<<< USERNAME:tt FILENAME: maslex.l >>>>

```
%{
//-----------------------------------------------------------------------
// File maslex.l defines a prototype scanner for the MARIE Assembler Language.
// The scanner definition is a lex specification.
//-----------------------------------------------------------------------
// Author: Thomas R. Turner
// E-Mail: trturner@uco.edu
// Date:   October, 2016
//-----------------------------------------------------------------------
//Copyright October, 2016 by Thomas R. Turner.
//Do not reproduce without permission from Thomas R. Turner
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// Standard C and C++ Library Include Files

//-----------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <cstdio>
#include <map>
using namespace std;
//-----------------------------------------------------------------------
// Application Includes
//-----------------------------------------------------------------------
#include "maslex.h"
#include "mastkn.h"
#include "masfiles.h"
//-----------------------------------------------------------------------
//Externals
//-----------------------------------------------------------------------
extern Files F;
//-----------------------------------------------------------------------
//Global Variables
//-----------------------------------------------------------------------
static map<string,int> RW;
static int tokencode;
static string* TokenName;
int Line=1;
int Col=1;
//-----------------------------------------------------------------------
//Functions
//-----------------------------------------------------------------------
int TokenMgr(int t);                    //Token post processing
                                        //Print the token and attributes
void PrintToken(ostream& o,int tc,int l,int c);
//-----------------------------------------------------------------------
//Exceptions
//-----------------------------------------------------------------------
struct StringTokenException{
    StringTokenException(char* t,int l,int c)
    {   cout << endl;
        cout << "line(" << l << ") col (" << c << ")" ;
        cout << " ";
        cout << "Lexical error: ";
        cout << "Strings cannot span lines";
        cout << endl;
        cout << "|" << t << "|";
        cout << endl;
    }
};
struct BadCharacterException{
    BadCharacterException(char p,int l,int c)
    {   cout << endl;
```

```
        cout << endl;
        cout << "Lexical error: ";
        cout << "Illegal character ASCII code=" << (int)p;
        cout << endl;
    }
};
%}

%%
[ \t]+                                  {Col+=strlen(yytext);}
[\r][\n]                                {Line++;Col=1;}
[\n]                                    {Line++;Col=1;}
[_a-zA-Z][_a-zA-Z0-9]*                  return TokenMgr(IDENTIFIER);
[0-9a-fA-F]+                            return TokenMgr(HEXLIT);
[\/][^\n]*[\n]                          {Line++;Col=1;}
","                                     return TokenMgr(COMMA);
.                                       {throw BadCharacterException
                                                (*yytext
                                                ,Line
                                                ,Col
                                                );
                                        }
%%
//-----------------------------------------------------------------------
//Class Lexer implementation
//-----------------------------------------------------------------------
//Function TokenMgr processes the token after it has been recognized
//-----------------------------------------------------------------------
int TokenMgr(int t)
{   int tc=t;
    //-------------------------------------------------------------------
    //Convert all strings to lower case.
    //-------------------------------------------------------------------
    for (int a=0;a<strlen(yytext);a++) yytext[a]=tolower(yytext[a]);
    if (t==IDENTIFIER) {
        //---------------------------------------------------------------
        //yylval is the yacc variable associated with the %union
        //directive. Member token was declared to have the semantic
        //value for tokens.
        //---------------------------------------------------------------
        yylval.token=new string(yytext);
        //---------------------------------------------------------------
        //Find the identifier in the list of reserve words. If the identifier
        //is a reserve word return its unique token code. Otherwise,
        //return zero.
        //---------------------------------------------------------------
        tc=RW[string(yytext)];
        //---------------------------------------------------------------
        //If the identifier was not a reserve word then restore the
        //original token code.
        //---------------------------------------------------------------
        if (tc==0) tc=t;

    } else {
        //---------------------------------------------------------------
        //Make the strings available for all tokens.
        //---------------------------------------------------------------
        yylval.token=new string(yytext);
    }
    PrintToken(F.tfs,tc,Line,Col);
    Col+=yyleng;
    return tc;
}
//-----------------------------------------------------------------------
//Constructor Lexer is used to redirect the input file stream from the
```

<<<< USERNAME:tt FILENAME: maslex.l >>>>

```
//--------------------------------------------------------------------------
Lexer::Lexer(FILE* i)
{    yyin=i;
     const int MAXSY=26;
     static string sy[]=
     { "TOKEN_BEGIN"
     , "COMMA"
     , "RESERVE_WORDS"
     , "JNS"
     , "LOAD"
     , "STORE"
     , "ADD"
     , "SUBT"
     , "INPUT"
     , "OUTPUT"
     , "HALT"
     , "SKIPCOND"
     , "JUMP"
     , "CLEAR"
     , "ADDI"
     , "JUMPI"
     , "LOADI"
     , "STOREI"
     , "DEC"
     , "HEX"
     , "ORG"
     , "END"
     , "REGULAR_EXPRESSIONS"
     , "IDENTIFIER"
     , "HEXLIT"
     , "TOKEN_END"
     };
     TokenName=new string[MAXSY];
     for (int a=0;a<MAXSY;a++) TokenName[a]=sy[a];
     static string rw[]=
     {
      "jns"
     ,"load"
     ,"store"
     ,"add"
     ,"subt"
     ,"input"
     ,"output"
     ,"halt"
     ,"skipcond"
     ,"jump"
     ,"clear"
     ,"addi"
     ,"jumpi"
     ,"loadi"
     ,"storei"
     ,"dec"
     ,"hex"
     ,"org"
     ,"end"
     };
     static int    tc[]=
     {
       JNS
     , LOAD
     , STORE
     , ADD
     , SUBT
     , INPUT
     , OUTPUT
```

<<<< USERNAME:tt FILENAME: maslex.l >>>>

```
    , SKIPCOND
    , JUMP
    , CLEAR
    , ADDI
    , JUMPI
    , LOADI
    , STOREI
    , DEC
    , HEX
    , ORG
    , END
    };
    for (int a=0;a<19;a++) RW[rw[a]]=tc[a];
}
//--------------------------------------------------------------------
//Function Lex calls yylex
//--------------------------------------------------------------------
int Lexer::Lex(void)
{   tokencode=yylex();
    return tokencode;
}
//--------------------------------------------------------------------
//Function PrintToken prints the token code name and the spelling of the
//token.
//--------------------------------------------------------------------
void PrintToken(ostream& o,int tc,int l,int c)
{   o << endl;
    o << "Token";
    o << ":Code="     << setw( 3) << tc;
    o << " Name="     << setw(10) << TokenName[tc-TOKEN_BEGIN];
    o << " line="     << setw( 3) << l;
    o << " col="      << setw( 3) << c;
    o << " Spelling=\"" << (char*)yytext << "\"";
}

//----------------------End of Lex Definition-----------------------
```