<<<< USERNAME:tt FILENAME: mex.cpp >>>>

```
//-----------------------------------------------------------------------
//File mex reads the executable file having a suffix of .mex
//and executes the instructions in the file.
//-----------------------------------------------------------------------
//Author: Thomas R. Turner
//E-Mail: trturner@uco.edu
//Date:   November, 2015
//-----------------------------------------------------------------------
//C++ Standard Include Files
//-----------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <cstdio>
#include <cstdlib>
#include <cstring>
using namespace std;
//-----------------------------------------------------------------------
//Application Include Files
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
//File Global Variables
//-----------------------------------------------------------------------
static short M[4096];                    //MARIE Memory
static short MAR=0;                      //MARIE Memory Address Register
static short PC=0;                       //MARIE Program Counter
static short MBR=0;                      //MARIE Memory Buffer Register
static short AC=0;                       //MARIE Accumulator
static unsigned char InReg=0;            //MARIE Input Register
static unsigned char OutReg=0;           //MARIE Output Register
static short IR=0;                       //MARIE Instruction Register
static int icount;                       //Number of instructions executed
//-----------------------------------------------------------------------
//Function ReadTrace traces the action of reading the input file
//-----------------------------------------------------------------------
void ReadTrace(ostream& o,const string t,int tv)
{
    o << endl;
    o << t << setfill(' ') << tv;
}
//-----------------------------------------------------------------------
//Function Trace prints the registers and the current instruction to the output
file
//-----------------------------------------------------------------------
void Trace(ostream& o,unsigned short opcode,unsigned short operand)
{   static string mnemonic[]=
        {"JnS"   ,"Load"    ,"Store","Add"  ,"Subt","Input","Output"
        ,"Halt"  ,"Skipcond","Jump" ,"Clear","AddI","JumpI","LoadI"
        ,"StoreI"
        };
    o << endl ;
    o << endl ;
    o << "opcode=" << setw(1) << hex << opcode;
    o << " ";
    o << "mnemonic=" << setw(8) << setfill(' ') << mnemonic[opcode];
    o << " ";
    o << "operand=" << setw(4) << setfill('0') << hex << operand;
    o << endl;
    o << "PC =" << setw(4) << setfill('0') << hex << PC;
    o << " ";
    o << "MAR=" << setw(4) << setfill('0') << hex << MAR;
    o << " ";
    o << "MBR=" << setw(4) << setfill('0') << hex << MBR;
    o << " ";
```

<<<< USERNAME:tt FILENAME: mex.cpp >>>>

```
    o << " ";
    o << "IR =" << setw(4) << setfill('0') << hex << IR;
    o << " ";
    o << "InReg =" << setw(4) << setfill('0') << hex << ((unsigned short int)InR
eg);
    o << " ";
    o << "OutReg =" << setw(4) << setfill('0') << hex << ((unsigned short int)Ou
tReg);
}
//-------------------------------------------------------------------------
//Function Fetch obtains the next instruction and decodes it into
//opcode and operand.
//-------------------------------------------------------------------------
void Fetch(unsigned short& opcode,unsigned short& operand)
{   unsigned short code=M[PC]&0xF000;
    opcode=code>>12;
    unsigned short rand=M[PC]&0x0FFF;
    operand=rand;
}
//-------------------------------------------------------------------------
//Function Interpret executes MARIE instructions.
//-------------------------------------------------------------------------
void Interpret(ostream& o)
{   enum mnemonic
        {JnS     ,Load    ,Store   ,Add     ,Subt    ,Input   ,Output
        ,Halt    ,Skipcond,Jump    ,Clear   ,AddI    ,JumpI   ,LoadI
        ,StoreI
        };
    unsigned short opcode;                          //Opcode for the current instru
ction
    unsigned short operand;                         //Operand for the current instr
uction
    Fetch(opcode,operand);
    Trace(o,opcode,operand);
    while (opcode!=Halt) {
        PC++;
        switch (opcode) {
            case JnS:
            break;
            case Load:
                AC=M[operand];
            break;
            case Store:
                M[operand]=AC;
            break;
            case Add:
                AC=AC+M[operand];
            break;
            case Subt:
                AC=AC-M[operand];
            break;
            case Input:
                cout << endl;
                cout << "Enter an integer i, 0<=i<=255. ";
                cin  >> InReg;
                AC=InReg;
            break;
            case Output:
                OutReg=AC;
                cout << "Output= " << "'" << OutReg << "'";
                cout << endl;
            break;
            case Halt:
            break;
            case Skipcond:
```

<<<< USERNAME:tt FILENAME: mex.cpp >>>>

```
                    case 0: if (AC<0)  PC=PC+1;
                    break;
                    case 1: if (AC==0) PC=PC+1;
                    break;
                    case 2: if (AC>0)  PC=PC+1;
                    break;
                    default:   //throw an exception
                    break;
                }//end switch
            break;
            case Jump:
                PC=operand;
            break;
            case Clear:
                AC=0;
            break;
            case AddI:
                AC=AC+M[M[operand]];
            break;
            case JumpI:
                PC=M[M[operand]];
            break;
            case LoadI:
                AC=M[M[operand]];
            break;
            case StoreI:
                M[M[operand]]=AC;
            break;
        }
        icount++;            //Increment the number of instructions executed
        Fetch(opcode,operand);
        Trace(o,opcode,operand);

    }
    o << endl << "icount=" << dec << icount;
    o << endl;

}
//------------------------------------------------------------------------
//FileException
//------------------------------------------------------------------------
struct FileException {
    FileException(const char* fn)
    {   cout << endl;
        cout << "File " << fn << " could not be opened.";
        cout << endl;
    }
};
//------------------------------------------------------------------------
//CommandLineException
//------------------------------------------------------------------------
struct CommandLineException {
    CommandLineException(int m,int a)
    {   cout << endl;
        cout << "Too many arguments on the command line.";
        cout << endl;
        cout << m << " argument(s) are permitted on the command line.";
        cout << endl;
        cout << a << " argument(s) appeared on the command line.";
        cout << endl;
    }
};
//------------------------------------------------------------------------
//InvalidSuffixException
//------------------------------------------------------------------------
```

<<<< USERNAME:tt FILENAME: mex.cpp >>>>

```
    InvalidSuffixException(char* fn,const char* s)
    {   cout << endl;
        cout << "File name " << "\"" << fn << "\"";
        cout << " must have the suffix " << "\"" << s << "\"";
        cout << endl;
    }
};
//---------------------------------------------------------------------------
//Function ValidSuffix determines if string fn has string s as a suffix.
//---------------------------------------------------------------------------
bool ValidSuffix(char* fn,const char* s)
{   char* p=strstr(fn,s);
    if (!p) return false;
    int sp=strlen(fn)-strlen(s);
    int cp=p-fn;
    if (cp!=sp) return false;
    return true;
}
//---------------------------------------------------------------------------
//Function Prefix removes the
//---------------------------------------------------------------------------
void Prefix(char* p,char* s)
{   int l=strlen(s)-4;
    strncpy(p,s,l);
    p[l]=0;
}
//---------------------------------------------------------------------------
//Function main directs the process of interpreting a MARIE program
//---------------------------------------------------------------------------
int main(int argc, char* argv[])
{
    try {
        //-------------------------------------------------------------------
        //Process command line arguments
        //-------------------------------------------------------------------
        char ifn[255];
        switch (argc) {
            case 1:                 //Prompt for the input file name
                cout << "Enter the input file name. ";
                cin >> ifn;
                break;
            case 2:                 //Read the input file name
                strcpy(ifn,argv[1]);
                break;
            default:
                throw CommandLineException(1,argc-1);
                break;
        }
        //-------------------------------------------------------------------
        //Validate the input file name and create output file having the same
        //prefix as the input file name but with a ".lst" suffix.
        //-------------------------------------------------------------------
        if (!ValidSuffix(ifn,".mex")) throw InvalidSuffixException(ifn,".mex");
        char ofn[255];              //Allocate storage for the output filename
        Prefix(ofn,ifn);            //Create the output filename prefix
        strcat(ofn,".lst");         //Append the suffix the output filename prefix

        ofstream o(ofn); if (!o) throw FileException(ofn);
        FILE* i=fopen(ifn,"rb"); if (!i) throw FileException(ifn);

        //-------------------------------------------------------------------
        //Read the binary file named on the command line into memory M.
        //Assign the initial value of the program counter, PC, also stored
        //in the binary file.
        //-------------------------------------------------------------------
```

<<<< USERNAME:tt FILENAME: mex.cpp >>>>

```
        unsigned short int length=0;

        int rc=fread(&origin,sizeof(short),1,i);
        ReadTrace(o,"rc",rc);
        ReadTrace(o,"origin",origin);
        o << endl;

        rc=fread(&length,sizeof(short),1,i);
        ReadTrace(o,"rc",rc);
        ReadTrace(o,"length",length);
        o << endl;

        rc=fread(&M[origin],sizeof(short),length,i);
        ReadTrace(o,"rc",rc);

        for (int address=origin;address<origin+length;address++) {
            o << endl;
            o << hex << setw(4) << setfill('0') << address;
            o << " ";
            o << hex << setw(4) << setfill('0') << M[address];
        }
        o << setfill(' ');
        o << endl;

        PC=origin;                  //Assign the address of the first instruction

        Interpret(o);               //Begin interpreting the program

    } catch(...) {
        cout << endl;
        cout << "Program terminated!";
        cout << endl;
        cout << "I won't be back!";
        cout << endl;
        exit(EXIT_FAILURE);
    }
    return 0;
}
```