U18CO018

Shekhaliya Shubham

Sub: System Software

Assignment 3

Implement First Pass Assembler(Symbol Table,Literal Table,Pool Table and Table of Incomplete Instructions) for multiplication of two numbers.

CODE

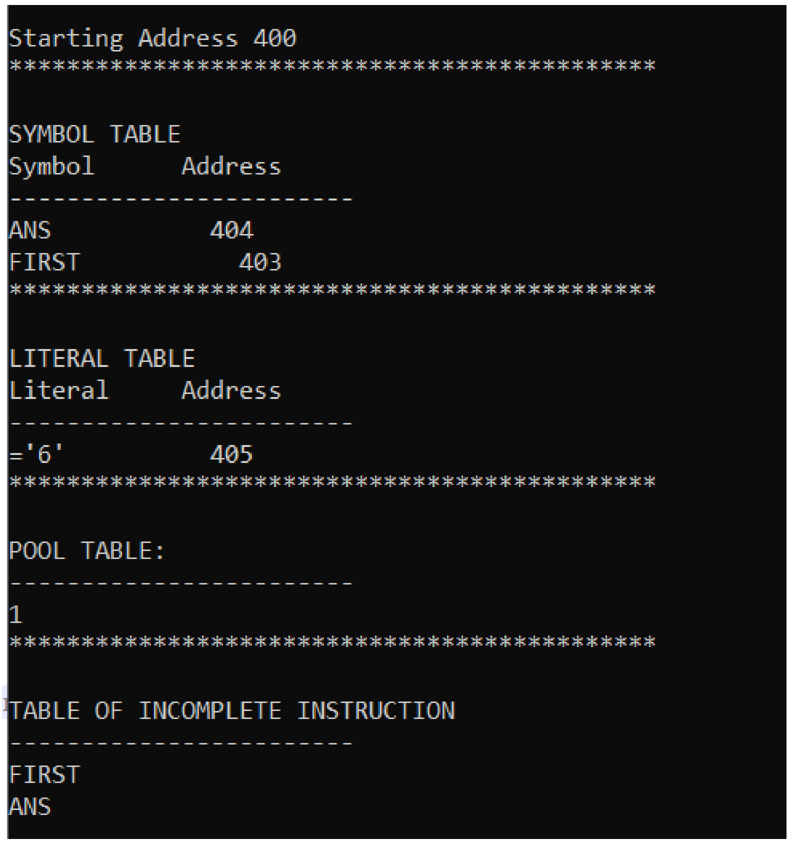
```
#include <bits/stdc++.h>

using namespace std;

vector<string> simple_tokenizer(string s)
{
    vector<string> in;
    stringstream ss(s);
    string word;
    while (ss >> word) {
        in.push_back(word);
    }
    return in;
}

bool isLetterOnly(string s) {

    for ( char c : s) {
        if(!isalpha(c)) {
            return false;
        }
    }

    return true;
}

int main() {
    map<string , int>
            mnemonics;
    mnemonics["MOVER"] = 1;
    mnemonics["MOVEM"] = 1;
    mnemonics["ADD"] = 1;
    mnemonics["SUB"] = 1;
    mnemonics["BC"] = 1;
    mnemonics["MOVER"] = 1;
    mnemonics["STOP"] = 1;
    mnemonics["MULT"] = 1;
    mnemonics["DS"] = 1;
    mnemonics["DC"] = 1;
    mnemonics["START"] = 0;
    mnemonics["LTROG"] = 0;
    mnemonics["END"] = 0;
    mnemonics["ORIGIN"] = 0;
    mnemonics["EQU"] = 0;
    mnemonics["COMP"] = 1;
    mnemonics["BC"] = 1;
    mnemonics["READ"] = 1;
    mnemonics["PRINT"] = 1;
    mnemonics["JUMP"] = 1;

    set<string> registerAndCondition;
    registerAndCondition.insert("LT");
    registerAndCondition.insert("LE");
    registerAndCondition.insert("EQ");
    registerAndCondition.insert("GT");
    registerAndCondition.insert("GE");
    registerAndCondition.insert("ANY");
    registerAndCondition.insert("AREG");
    registerAndCondition.insert("BREG");
    registerAndCondition.insert("CREG");
    registerAndCondition.insert("DREG");

    int literal = 0;



    // answere
    map<string, int> symbolTable;

    vector<int> poolTable;
    poolTable.push_back(1);

    vector<string> TII;

    vector<pair<string, int>> literalTable;


    string line;
    ifstream
input("E:\\Asem6\\ss\\practical\\Assignment3\\input1.asm");
```

```
    int add = 0;

    getline(input,line);
    vector<string> in = simple_tokenizer(line);
    add = stoi(in[1]);
    cout<<endl<<"Starting Address "<<add<<endl;

cout<<"*********************************************"<<endl<<e
ndl;

    while(getline(input,line)) {
        in = simple_tokenizer(line);
        if(in[0] == "LTROG" || in[0] == "END") {
//              add += literal;
            if(literal != 0) {
                int x = poolTable[poolTable.size() - 1] - 1;
                for(int i = 0;i<literal;i++) {
                    literalTable[x] =
make_pair(literalTable[x].first, add++);
                    x++;
                }
                poolTable.push_back(poolTable[poolTable.size()
- 1] + literal);
            }
            literal = 0;

            if(in[0] == "END")
                break;

        } else if (in[0] == "START") {
            continue;
        } else if (in[0] == "ORIGIN") {
            add = stoi(in[1]);
        } else {
            if(mnemonics.find(in[0]) == mnemonics.end()) {
                // then is the symbol at teh start of teh
instruction
                symbolTable[in[0]] = add++;

                if(in[1] == "EQU") {
                    symbolTable[in[0]] = symbolTable[in[2]];
                } else {
                    for(int i = 2; i<in.size();i++) {

                        string t = in[i];

                        if(t[t.size()-1] == ',') {
                            t = t.substr(0,t.size() - 1);
                        }
                        if(registerAndCondition.find(t) !=
registerAndCondition.end()) {
                            continue;
                        }

                        if(t.substr(0,1) == "=") {

literalTable.push_back(make_pair(t,-1));
                            literal++;

                        } else {
                            if(isLetterOnly(t) &&
symbolTable.find(t) == symbolTable.end()) {
                                TII.push_back(t);
                                symbolTable[t] = -1;
                            }
                        }
                    }
                }
            } else {
                for(int i = 1; i<in.size();i++) {

                    string t = in[i];

                    if(t[t.size()-1] == ',') {
                        t = t.substr(0,t.size() - 1);
                    }

                    if(registerAndCondition.find(t) !=
registerAndCondition.end()) {
                        continue;
                    }

                    if(t.substr(0,1) == "=") {
                        literalTable.push_back(make_pair(t,-
1));

                        literal++;

                    } else {
                        if(isLetterOnly(t) &&
symbolTable.find(t) == symbolTable.end()) {
                            TII.push_back(t);
                            symbolTable[t] = -1;
                        }
                    }
                }
                add++;
            }


        }
```

```
    }
    cout<<"SYMBOL TABLE"<<endl;
    cout<<"Symbol      Address"<<endl;
    cout<<"------------------------"<<endl;
    for (const auto& i : symbolTable) {
        cout<< i.first << "           " <<i.second <<endl;
    }

cout<<"*********************************************"<<endl<<e
ndl;

    cout<<"LITERAL TABLE"<<endl;
    cout<<"Literal     Address"<<endl;
    cout<<"------------------------"<<endl;
    for(const auto&i : literalTable) {
        cout<<i.first<<"          "<<i.second<<endl;
    }

cout<<"*********************************************"<<endl<<e
ndl;

    poolTable.pop_back();
    cout<<"POOL TABLE:"<<endl;
    cout<<"------------------------"<<endl;
    for(const auto& i : poolTable) {
        cout<<i<<endl;
    }

cout<<"*********************************************"<<endl<<e
ndl;

    cout<<"TABLE OF INCOMPLETE INSTRUCTION"<<endl;
    cout<<"------------------------"<<endl;
    for(const auto& i : TII) {
        cout<<i<<endl;
    }

    return 0;
}
```

1. INPUT 1

```
START 400

MOVER AREG, FIRST

MULT AREG, ='6'

MOVEM AREG, ANS

FIRST DC 5

ANS DS 1

END
```



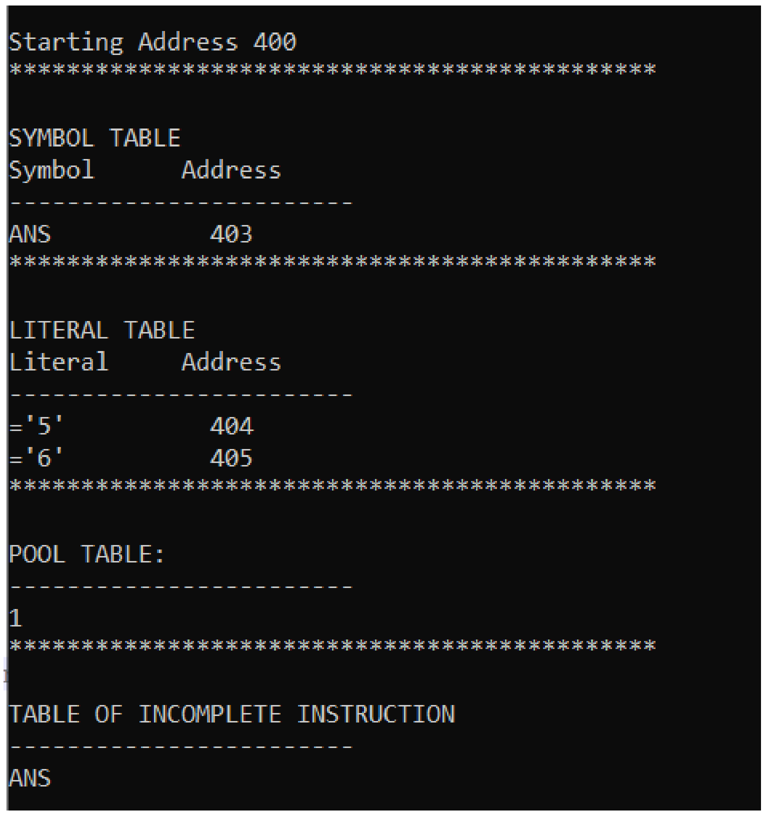
```
Starting Address 400
********************************************

SYMBOL TABLE
Symbol      Address
-----------------------
ANS           404
FIRST           403
********************************************

LITERAL TABLE
Literal     Address
-----------------------
='6'          405
********************************************

POOL TABLE:
-----------------------
1
********************************************

TABLE OF INCOMPLETE INSTRUCTION
-----------------------
FIRST
ANS
```

2. INPUT 2

```
START 400

MOVER AREG, ='5'

MULT AREG, ='6'

MOVEM AREG, ANS
```

ANS DS 1

END



```
Starting Address 400
*********************************************

SYMBOL TABLE
Symbol      Address
------------------------
ANS           403
*********************************************

LITERAL TABLE
Literal     Address
------------------------
='5'          404
='6'          405
*********************************************

POOL TABLE:
------------------------
1
*********************************************

TABLE OF INCOMPLETE INSTRUCTION
------------------------
ANS
```

3. INPUT 3

START 400

MOVER AREG, ='5'

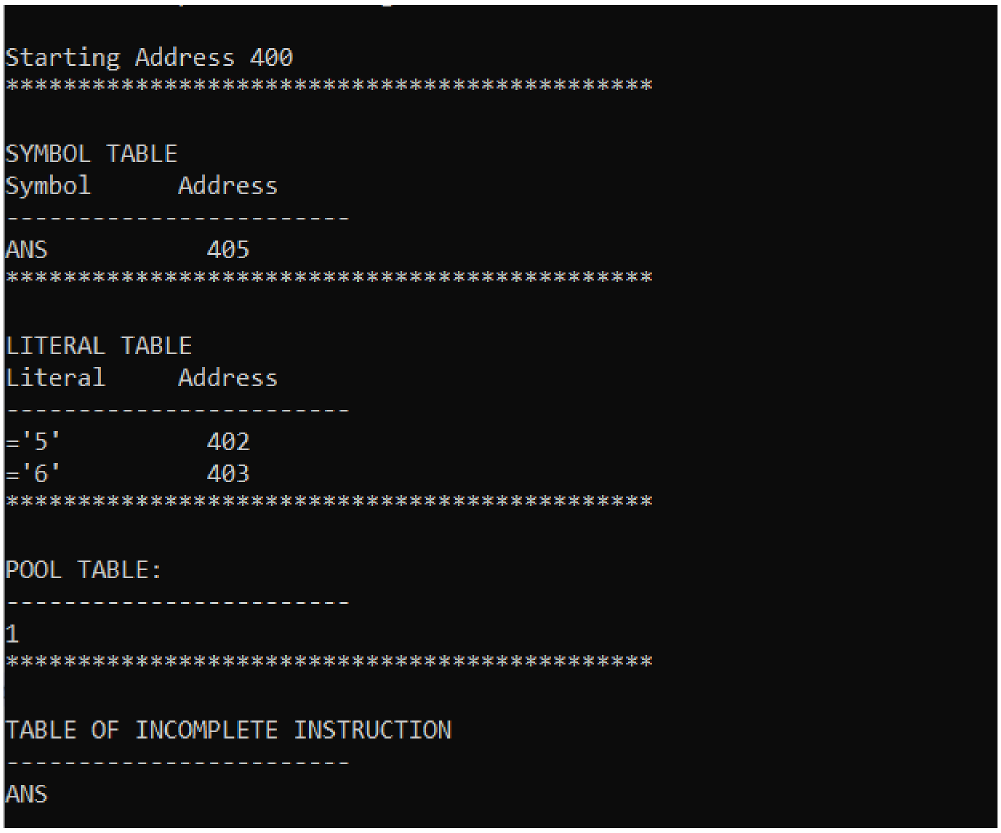
MULT AREG, ='6'

LTROG

MOVEM AREG, ANS

ANS DS 1

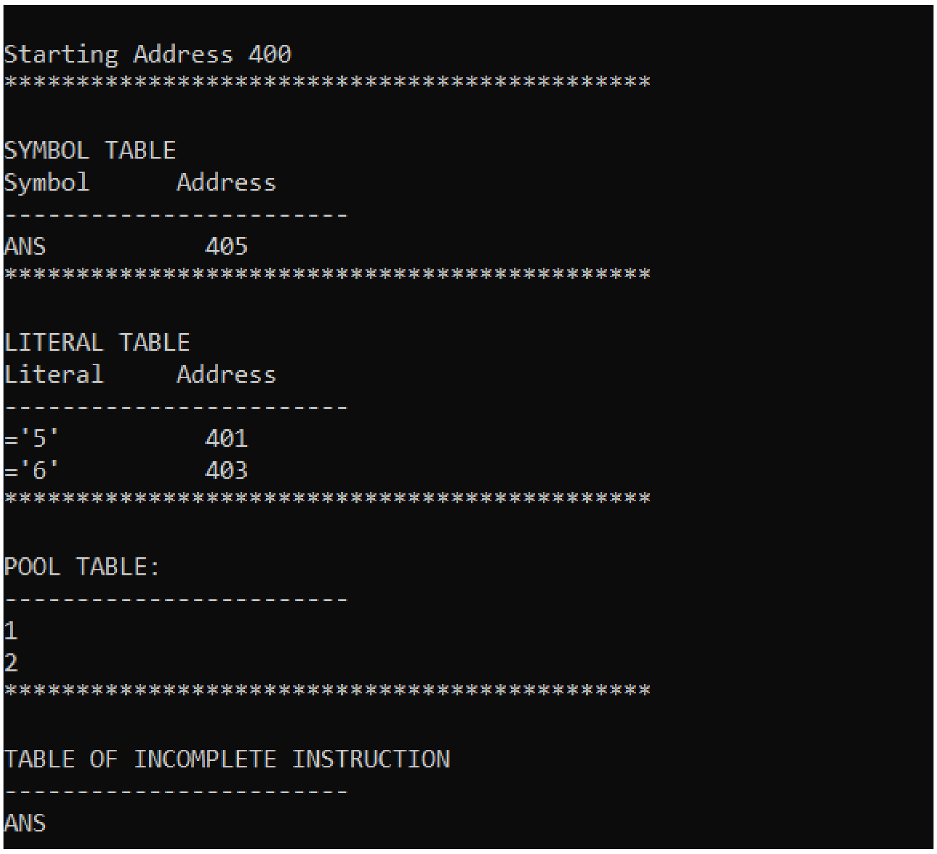
END



```
Starting Address 400
*********************************************

SYMBOL TABLE
Symbol      Address
------------------------
ANS           405
*********************************************

LITERAL TABLE
Literal     Address
------------------------
='5'          402
='6'          403
*********************************************

POOL TABLE:
------------------------
1
*********************************************

TABLE OF INCOMPLETE INSTRUCTION
------------------------
ANS
```

4. INPUT 4

```
START 400

MOVER AREG, ='5'

LTROG

MULT AREG, ='6'

LTROG

MOVEM AREG, ANS

ANS DS 1

END
```



```
Starting Address 400
*********************************************

SYMBOL TABLE
Symbol      Address
------------------------
ANS           405
*********************************************

LITERAL TABLE
Literal     Address
------------------------
='5'          401
='6'          403
*********************************************

POOL TABLE:
------------------------
1
2
*********************************************

TABLE OF INCOMPLETE INSTRUCTION
------------------------
ANS
```