LP-I Assignment No. 1

Name : Ravindra Dayaram Bagul
Roll No. : C31105
Batch : T-1

Problem Statement : Design suitable Data structures and implement Pass-I of a two-pass assembler for pseudo machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

PROGRAM

```
#include <iostream>
#include <conio.h>
#include <string.h>

using namespace std;

void lite(string lit[2][3])
{
   for (int l = 0; l < 2; l++)
   {
      cout << "(DL, 02)  (C, " << lit[l][1]<<")" << endl;
   }
}
void chcklit(string lit[2][3], string l)
{
   for (int i = 0; i < 2; i++)
   {
      if (l == lit[i][1])
      {
         cout << "(L, " << i << ")";
      }
   }
}

void chcksym(string sym[2][3], string s)
{
   int cons=0;
   for (int i = 0; i < 2; i++)
   {
      if (s == sym[i][1])
      {
         cout << "(S, " << i << ")";
         cons++;
      }

   }
```

```
    if(cons==0)
    {
        cout << "(C, " << s << ")";
    }

}

void check(string MOT[8][3], string instr[8][5], string sym[2][3], string lit[2][3])
{

    int i, j;
    i = 0;

    while (i < 8)
    {
        j = 0;
        if (instr[i][j] == "origin")
        {
            cout << "(AD, 03)" << endl;
            i++;
        }
        else if (instr[i][j] == "ltorg")
        {
            lite(lit);
            i++;
            cout << endl;
        }
        else
        {
            while (j < 5 && instr[i][j]!="origin" && instr[i][j]!="ltorg")
            {
                /* code */
                int m=0;
                while(m<8)
                {
                    if (instr[i][j] == MOT[m][0])
                    {
                        cout << "(" << MOT[m][1] << ", " << MOT[m][2] << ") ";
                        j++;
                        m=0;
                    }
                    else
                    {
                        m++;
                    }
                }
```

```
            if (instr[i][j] == "2" || instr[i][j] == "1")
            {
               string litval = instr[i][j];
               chcklit(lit, litval);
               j++;
            }
            else if (instr[i][j] ==" " || instr[i][j] == "+" || instr[i][j] == "," || instr[i][j]
== ",=")
            {
               j++;
               continue;
            }

            else
            {
               string symb = instr[i][j];
               chcksym(sym, symb);
               j++;
            }
         }
         i++;
         cout << endl;
      }
   }
}

void MOT(string instr[8][5], string sym[2][3], string lit[2][3])
{
   string machinetable[8][3] = {"start", "AD", "01",
                        "mover", "IS", "04",
                        "breg", "RG", "02",
                        "areg", "RG", "01",
                        "add", "IS", "01",
                        "origin", "AD", "04",
                        "ltorg", "AD", "03",
                        "dc","DL","01"};

   check(machinetable, instr, sym, lit);
}

void symbol(string s[2][3])
{
   cout << "\nEnter Entries in symbol table \n";
   for (int i = 0; i < 2; i++)
   {
      cout << "Enter symbol index : ";
      cin >> s[i][0];
      cout << "Enter symbol : ";
```

```
        cin >> s[i][1];
        cout << "Enter Location Counter : ";
        cin >> s[i][2];
        cout<<endl;

    }
}

void litetable(string lit[2][3])
{
    cout << "\nEnter Entries in literal table \n";
    for (int i = 0; i < 2; i++)
    {
        cout << "Enter Literal index : ";
        cin >> lit[i][0];
        cout << "Enter Literal : ";
        cin >> lit[i][1];
        cout << "Enter Location Counter : ";
        cin >> lit[i][2];
        cout<<endl;
    }
}

void display(string s[2][3], string lit[2][3])
{
    cout<<"-----------------------------------"<<endl;
    cout << "Index  "
         << "  Symbol"
         << "  Location Counter" << endl;
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {

            cout << " " << s[i][j] << "       ";
        }
        cout << endl;
    }

    cout<<"-----------------------------------"<<endl;

    cout << "Index  "
         << "   literal"
         << "    Location Counter" << endl;
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {
```

```
            cout << " " << lit[i][j] << "        ";
        }
        cout << endl;
    }
    cout<<"-------------------------------------"<<endl;
}

int main()
{
    string sym[2][3];
    string lit[2][3];
    // symbol(s);
    // litetable(li);
    // display(s);
    string let[8][5] = {"start", "100"," ", " ", " ",
                        "mover", "breg", ",=","2", " ",
                        "loop", "mover", "areg", ",", "n",
                        "add","breg", ",=", "1", " ",
                        "origin", "loop", "+", "5", " ",
                        "ltorg", " ", " ", " ", " ",
                        "n", "dc", "5", " ", " ",
                        "end", " ", " ", " ", " "};

    for(int i=0; i<8;i++)
    {
        for(int j=0; j<5; j++)
        {
            cout<<let[i][j]<<" ";
        }
        cout<<endl;
    }

    symbol(sym);
    litetable(lit);
    display(sym, lit);

   // MOT(let);
    MOT(let, sym, lit);

    return 0;
}
```

Output

```
start 100
mover breg ,= 2
loop mover areg , n
add breg ,= 1
origin loop + 5
ltorg
n dc 5
end

Enter Entries in symbol table
        Enter symbol index : 0
        Enter symbol : loop
        Enter Location Counter : 101

        Enter symbol index : 1
        Enter symbol : n
        Enter Location Counter : 106


Enter Entries in literal table
        Enter Literal index : 0
        Enter Literal : 2
        Enter Location Counter : 106

        Enter Literal index : 1
        Enter Literal : 1
        Enter Location Counter : 107

---------------------------------------------
Index   Symbol   Location Counter
 0        loop          101
 1        n             106
---------------------------------------------
Index    literal    Location Counter
 0          2            106
 1          1            107
---------------------------------------------

(AD, 01) (C, 100)
(IS, 04) (RG, 02) (L, 0)
(S, 0)(IS, 04) (RG, 01) (S, 1)
(IS, 01) (RG, 02) (L, 1)
(AD, 03)
(DL, 02)  (C, 2)
(DL, 02)  (C, 1)
(S, 1)(DL, 01) (C, 5)
(C, end)
```