#include <bits/stdc++.h>

using namespace std;

class passOne

{

    map<string, pair<string, string>> optab;

    map<string, int> reg;

    map<string, int> cond;

    vector<pair<string, int>> symtab;

    vector<pair<string, int>> littab;

    vector<string> pooltab;

    vector<string> split(char del, string line)

    {S

        vector<string> res;

        int i = 0;

        string word = "";

        while (line[i] != '\0')

        {

            if (line[i] == del)

            {

                res.push\_back(word);

                word = "";

            }

            else

            {

                word += line[i];

            }

            i++;

        }

        if (word != "")

        {

            res.push\_back(word);

        }

        return res;

    }

    void makeOptab()

    {

        ifstream obj("optab.txt");

        if (!obj.is\_open())

        {

            cerr << "\nError in opening file!";

            return;

        }

        string line;

        while (getline(obj, line))

        {

            vector<string> ans = split(' ', line);

            optab[ans[0]] = {ans[1], ans[2]};

        }

        obj.close();

        //      for(auto i:optab){

        //          cout<<i.first<<" "<<i.second.first<<" "<<i.second.second<<endl;

        //      }

    }

    void makeReg()

    {

        reg["AREG"] = 1;

        reg["BREG"] = 2;

        reg["CREG"] = 3;

        reg["DREG"] = 4;

    }

    void makeCond()

    {

        cond["LT"] = 1;

        cond["LE"] = 2;

        cond["EQ"] = 3;

        cond["GT"] = 4;

        cond["GE"] = 5;

        cond["ANY"] = 6;

    }

    int find\_symtab(string x)

    {

        for (int i = 0; i < symtab.size(); i++)

        {

            if (symtab[i].first == x)

            {

                return i;

            }

        }

        return -1;

    }

public:

    passOne()

    {

        makeOptab();

        makeReg();

        makeCond();

    }

    void generate\_ic(string file\_name)

    {

        ifstream obj(file\_name);

        if (!obj.is\_open())

        {

            cerr << "\nError in opening file!";

            return;

        }

        ofstream output("output.txt");

        if (!output.is\_open())

        {

            cerr << "\nError in opening file!";

            return;

        }

        string line;

        int lc = 0;

        bool org = 0;

        while (getline(obj, line))

        {

            vector<string> ans = split(' ', line);

            //              cout<<line<<endl;

            // START

            if (ans[0] == "START")

            {

                lc = stoi(ans[1]);

                output << "\t(AD,01) (C," << lc << ")" << endl;

            }

            // LTORG,STOP

            else if (ans[0] == "LTORG" || ans[0] == "END")

            {

                bool flag = 0;

                if (ans[0] == "END")

                {

                    output << "(AD,02)" << endl;

                }

                for (int i = 0; i < littab.size(); i++)

                {

                    if (littab[i].second == -1)

                    {

                        if (flag == 0)

                        {

                            string a = to\_string(i + 1);

                            pooltab.push\_back("#" + a);

                            flag = 1;

                        }

                        littab[i].second = lc;

                        string val = "";

                        for (int j = 2; j < littab[i].first.length() - 1; j++)

                        {

                            val += littab[i].first[j];

                        }

                        output << lc << " (DL,01) (C," << val << ")" << endl;

                        lc++;

                    }

                }

            }

            else if (ans[0] == "ORIGIN")

            {

                output << "(AD,03) ";

                string lab = "";

                int i = 0;

                while (i < ans[1].size() && ans[1][i] != '+' && ans[1][i] != '-')

                {

                    lab += ans[1][i];

                    i++;

                }

                char sign = ans[1][i];

                i++;

                string pos = "";

                while (i < ans[1].size())

                {

                    pos += ans[1][i];

                    i++;

                }

                i = find\_symtab(lab);

                //                  cout<<"sign "<<lab<<endl;

                //                  cout<<"ps "<<pos<<endl;

                output << "(S," << i + 1 << ")" << sign << pos << endl;

                if (sign == '-')

                {

                    lc = symtab[i].second - stoi(pos);

                }

                else

                {

                    lc = symtab[i].second + stoi(pos);

                }

                if (org == 1)

                {

                    org = 0;

                }

                else

                {

                    org = 1;

                }

            }

            else

            { // other lines

                int ind = 0;

                output << lc << " ";

                map<string, pair<string, string>>::iterator it;

                map<string, int>::iterator it2;

                it = optab.find(ans[0]);

                if (it == optab.end())

                { // label

                    if (ans[1] == "EQU")

                    {

                        int i = 0;

                        string lab = "";

                        while (i < ans[2].length() && ans[2][i] != '+' && ans[2][i] != '-')

                        {

                            lab += ans[2][i];

                            i++;

                        }

                        int pos = 0;

                        char sign;

                        if (i != ans[2].length())

                        {

                            sign = ans[2][i];

                            i++;

                            string x = "";

                            while (i < ans[2].length())

                            {

                                x += ans[2][i];

                                i++;

                            }

                            pos = stoi(x);

                        }

                        i = find\_symtab(lab);

                        int j = find\_symtab(ans[0]);

                        if (sign == '+')

                        {

                            pos = pos + symtab[i].second;

                        }

                        else

                        {

                            pos = symtab[i].second - pos;

                        }

                        if (j != -1)

                        {

                            symtab[j].second = pos;

                        }

                        else

                        {

                            symtab.push\_back({ans[0], pos});

                        }

                        ind += 2;

                        lc--;

                    }

                    else

                    {

                        int pos = find\_symtab(ans[0]);

                        if (pos == -1)

                        {

                            symtab.push\_back({ans[0], lc});

                        }

                        else

                        {

                            symtab[pos].second = lc;

                        }

                        ind++;

                        it = optab.find(ans[1]); // opcode

                    }

                }

                if (it != optab.end())

                { // no label

                    output << "(" << (it->second).first << "," << (it->second).second << ") ";

                }

                ind++;

                if (ind != ans.size())

                {

                    // symbol,reg,condtional,constant

                    it2 = reg.find(ans[ind]);

                    if (it2 != reg.end())

                    { // register

                        output << it2->second << " ";

                    }

                    else

                    {

                        it2 = cond.find(ans[ind]);

                        if (it2 != cond.end())

                        { // conditional

                            output << it2->second << " ";

                        }

                        else

                        {

                            if (ans[ind - 1] == "DC")

                            {

                                string c = "";

                                string a = ans[ind];

                                cout << ans[ind] << endl;

                                for (int i = 1; i < a.length() - 1; i++)

                                {

                                    cout << a[i] << endl;

                                    c += a[i];

                                }

                                output << "(C," << c << ") ";

                            }

                            else if (ans[ind - 1] == "DS")

                            { // constant

                                output << "(C," << ans[ind] << ") ";

                            }

                            else

                            { // symbol

                                int pos;

                                pos = find\_symtab(ans[ind]);

                                if (pos == -1)

                                { // forward reference

                                    symtab.push\_back({ans[ind], -1});

                                    output << "(S," << symtab.size() << ") ";

                                }

                                else

                                {

                                    output << "(S," << pos + 1 << ") ";

                                }

                            }

                        }

                    }

                    ind++;

                }

                if (ind != ans.size())

                {

                    // symbol,literal,register

                    it2 = reg.find(ans[ind]);

                    if (it2 != reg.end())

                    { // register

                        output << it2->second << " ";

                    }

                    else

                    {

                        if (ans[ind][0] == '=')

                        { // literal

                            littab.push\_back({ans[ind], -1});

                            output << "(L," << littab.size() << ") ";

                        }

                        else

                        { // symbol

                            int pos;

                            pos = find\_symtab(ans[ind]);

                            if (pos == -1)

                            { // forward reference

                                symtab.push\_back({ans[ind], -1});

                                output << "(S," << symtab.size() << ") ";

                            }

                            else

                            {

                                output << "(S," << pos + 1 << ") ";

                            }

                        }

                    }

                }

                output << endl;

                lc++;

            }

        }

        output.close();

        obj.close();

    }

    void make\_files()

    {

        ofstream file;

        file.open("symtab.txt");

        for (auto x : symtab)

        {

            file << x.first << " " << x.second << endl;

        }

        file.close();

        file.open("littab.txt");

        for (auto x : littab)

        {

            file << x.first << " " << x.second << endl;

        }

        file.close();

        file.open("pooltab.txt");

        for (string x : pooltab)

        {

            file << x << endl;

        }

        file.close();

    }

};

int main()

{

    passOne obj;

    obj.generate\_ic("example3.txt");

    obj.make\_files();

    return 0;

}

/\*

example3.txt -

START 100

A DS 1

L1 MOVER AREG B

ADD AREG C

MOVEM AREG D

D EQU A+1

L2 PRINT D

ORIGIN A-1

C DC '5'

ORIGIN L2+1

STOP

B DC '19'

END L1

optab.txt -

STOP IS 00

ADD IS 01

SUB IS 02

MULT IS 03

MOVER IS 04

MOVEM IS 05

COMP IS 06

BC IS 07

DIV IS 08

READ IS 09

PRINT IS 10

START AD 01

END AD 02

ORIGIN AD 03

EQU AD 04

LTORG AD 05

DC DL 01

DS DL 02

\*/