// Two-Pass Assembler: Pass-2 Code

**Input:**

--- (AD,01) (C,200) NAN

200 (IS,04) (1) (L,01)

201 (IS,05) (1) (S,01)

202 (IS,04) (1) (S,01)

203 (IS,04) (3) (S,03)

204 (IS,01) (3) (L,02)

205 (IS,04) (1) (S,01)

206 (IS,04) (3) (S,03)

207 (IS,04) (1) (S,01)

208 (IS,04) (3) (S,03)

209 (IS,04) (1) (S,01)

210 (IS,07) (6) (S,04)

211 (DL,01) (C,5) NAN

212 (DL,01) (C,1) NAN

213 (IS,04) (1) (S,01)

214 (IS,02) (1) (L,02)

215 (IS,07) (1) (S,05)

216 (IS,00) NAN NAN

--- (AD,03) (S,02)+2 NAN

204 (IS,03) (3) (S,03)

--- (AD,03) (S,06)+1 NAN

217 (DL,02) (C,1) NAN

--- (AD,04) NAN NAN

218 (DL,02) (C,1) NAN

--- (AD,02) NAN NAN

219 (DL,01) (C,1) NAN

**Code:**

#include <bits/stdc++.h>

using namespace std;

// Function to fetch symbol/literal address from symbol\_table or literal\_table

string table(ifstream &fin, string n)

{

    string no, name, addr;

    while(fin >> no >> name >> addr)

    {

        if(no == n)

        {

            fin.seekg(0, ios::beg);

            return addr;

        }

    }

    fin.seekg(0, ios::beg);

    return "NAN";

}

int main()

{

    ifstream ic, st, lt;

    // pass1 output files as input to pass2

    ic.open("ic.txt"); st.open("symtable.txt"); lt.open("littable.txt");

    // generate file output of machine code

    ofstream mc;

    mc.open("machine\_code.txt");

    string lc, ic1, ic2, ic3;

    cout << "\n -- ASSEMBLER PASS-2 OUTPUT --" << endl;

    cout << "\n LC\t <INTERMEDIATE CODE>\t\t\tLC\t <MACHINE CODE>" << endl;

    while(ic >> lc >> ic1 >> ic2 >> ic3) // reading input file line by line

    {

        string MC; // machine code

        // no machine code for AD and DL,02 i.e. DS opcodes

        if(ic1.substr(1, 2) == "AD" || (ic1.substr(1, 2) == "DL" && ic1.substr(4, 2) == "02"))

            MC = " -No Machine Code-";

        // if opcode is DL i.e. DL,01 then display constant value at the place of memory operand

        else if(ic1.substr(1, 2) == "DL" && ic1.substr(4, 2) == "01")

        {

            MC = "00\t0\t00" + ic2.substr(3, 1);

        }

        else // IS opcode

        {

            if(ic1 == "(IS,00)") // specifically for STOP

                MC = ic1.substr(4, 2) + "\t0\t000";

            else if(ic2.substr(1, 1) == "S") // if opcode in pass1 was ORIGIN

                MC = ic1.substr(4, 2) + "\t0\t" + table(st, ic2.substr(4, 1));

            else

            {

                if(ic3.substr(1, 1) == "S") // for symbols

                    MC = ic1.substr(4, 2) + "\t" + ic2.substr(1, 1) + "\t" + table(st, ic3.substr(4, 1));

                else // for literals

                    MC = ic1.substr(4, 2) + "\t" + ic2.substr(1, 1) + "\t" + table(lt, ic3.substr(4, 1));

            }

        }

        if(ic1 == "(AD,03)") // just for console output display format

        {

            cout << " " << lc << "\t" << ic1 << "\t" << ic2 << " " << ic3 << "\t\t\t" << lc << "\t" << MC << endl;

            mc << lc << "\t" << MC << endl;

            continue;

        }

        // console output

        cout << " " << lc << "\t" << ic1 << "\t" << ic2 << "\t " << ic3 << "\t\t\t" << lc << "\t" << MC << endl;

        mc << lc << "\t" << MC << endl;

    }

    return 0;

}