U18CO018
Shubham Shekhaliya
Assignment – 5(SS)

Generate Macro Definition Table(MDT) for given macro definition:

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;
}

bool isNumberOnly(string s) {
    for(char c : s) {
        if(!isdigit(c)) {
            return false;
        }
    }
    return true;
}

int main() {

    string line;
    ifstream input("input.asm");
```

```
    vector<string> in;

    getline(input, line);
    in = simple_tokenizer(line);

    if(in[0] != "MACRO") {
        cout<<"error"<<endl;
        exit(0);
    }

    int cpntab = 0;
    int ckpdtab = 0;
    int cevntab = 0;
    int cssntab = 0;

    unordered_map<string, int> pntab;
    unordered_map<string, pair<string, int>> kpdtab;
    unordered_map<string, int> evntab;
    unordered_map<string, int> ssntab;

    getline(input, line);
    in = simple_tokenizer(line);

    string macroName = in[0];
    cout<<macroName<<endl;

    for (int i = 1; i<in.size();i++) {
        string t = in[i];

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

        int p = -1;

        for(int j = 0; j<t.size();j++) {
            char c = t[j];
            if (c == '=') {
                p = j;
                break;
            }
        }

        if(p != -1) {
```

```
            string t1 = t.substr(1,p-1);
            string t2 = t.substr(p+1);

            pntab[t1] = ++cpntab;
            kpdtab[t1] = {t2, ++ckpdtab};

        } else {
            string temp = t.substr(1);
            pntab[temp] = ++cpntab;
        }

    }
    while(getline(input, line)) {
        in = simple_tokenizer(line);

        // cout<<line;

        for(int i = 0; i<in.size();i++) {
            string p = in[i];
            if(i==0) {
                if(p.substr(0,1) == ".") {
                    ssntab[p] = ++cssntab;
                }

                if(p == "LCL") {
                    string pp = in[1].substr(1);
                    evntab[pp] = ++cevntab;
                }
            }
        }
    }

    // for(auto i : pntab) {
    //     cout<<i.first << " " << i.second;
    // }

    // cout<<"pntab " << pntab.size() ;

    cout<<endl<<"*********************************"<<endl;
    cout<<"PNTAB "<<endl;
    for(pair<string , int > p:pntab) {
        cout<<p.first<<" "<<p.second<<endl;
    }
```

```
    cout<<"*******************************";
    cout<<endl<<"KPDTAB "<<endl;
    for(pair<string , pair<string,int> > p:kpdtab) {
        cout<<p.first<<" "<<p.second.first<<" "<<p.second.second<<endl;
    }

    cout<<"*******************************";
    cout<<endl<<"EVNTAB "<<endl;
    for(pair<string,int > p:evntab) {
        cout<<p.first<<" "<<p.second<<endl;
    }
    cout<<"*******************************";
    cout<<endl<<"SSNTAB "<<endl;
    for(pair<string,int > p:ssntab) {
        cout<<p.first<<" "<<p.second<<endl;
    }

    line = "";
    ifstream input2("input.asm");

    getline(input2, line);
    getline(input2, line);
    vector<string> ans1;
    set< string > st;
    st.insert("LCL");
    st.insert("SET");
    st.insert("MOVER");
    st.insert("MOVEM");
    st.insert("SET");
    st.insert("AIF");
    st.insert("MEND");

    int c=0;

    while(getline(input2, line)) {
        in = simple_tokenizer(line);

        string str1 = "";

        if(in[0] == "LCL") {
            str1 += "( LCL )    ";
            string pp = in[1].substr(1);

            str1 += "(E, " + to_string(evntab[pp]) + ")     ";
            ans1.push_back(str1);
```

```
            continue;
        }

        for(int i = 0; i<in.size();i++) {
            if(i == 0) {
                if(st.find(in[i]) == st.end()) {

                    if (in[i].substr(0,1) == ".") {
                        str1 += "(S, " + to_string(ssntab[in[i]]) + " )     ";
                    } else {
                        str1 += "(P, " + to_string(pntab[in[i]]) + " )      ";
                    }
                } else {
                    str1 += in[i] + "       ";
                }
            } else {
                string temp = in[i];
                if(temp.substr(0,1) == "&") {
                    temp = temp.substr(1);
                    str1 += "(P, " + to_string(pntab[temp]) + " )      ";
                } else if (isNumberOnly(in[i])) {
                    str1 +=   "(    "+ in[i] +"    )";
                } else if (in[i].substr(0,1) == "=") {
                    str1 += "(L, " +  to_string(++c) + ")";
                } else if (in[i] == "EQ" || in[i] == "+" || in[i] == "*") {
                    str1 += "( "+in[i]+")";
                }
            }
        }
        ans1.push_back(str1);
    }

    cout<<endl<<"*********************************"<<endl;
    cout<<"ANS "<<endl;
    for(string p:ans1) {
        cout<<p<<endl;
    }

    return 0;
}
```

Input.asm

```
MACRO
CLEARMEM &X, &N, &REG=AREG
LCL &M
&M SET 0
MOVER &REG, ='0'
.MORE MOVEM        &REG, &X + &M
&M SET &M + 1
AIF (&M NE N) .MORE
MEND
```

Output :-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  1: bash

shubham@DESKTOP-RB7GMLA:/mnt/c/users/shubh/desktop/new folder/ss/assignment5$ g++ main.cpp
shubham@DESKTOP-RB7GMLA:/mnt/c/users/shubh/desktop/new folder/ss/assignment5$ ./a.out
CLEARMEM
&X
&N
&REG=AREG

********************************
PNTAB
N 2
REG 3
X 1
********************************
KPDTAB
REG AREG 1
********************************
EVNTAB
M 1
********************************
SSNTAB
.MORE 1

********************************
ANS
( LCL )    (E, 1)
(P, 0 )      (    0    )
MOVER       (P, 0 )      (L, 1)
(S, 1 )     (P, 0 )      (P, 1 )      ( +)(P, 0 )
(P, 0 )      (P, 0 )      ( +)(    1    )
AIF
MEND
shubham@DESKTOP-RB7GMLA:/mnt/c/users/shubh/desktop/new folder/ss/assignment5$
```