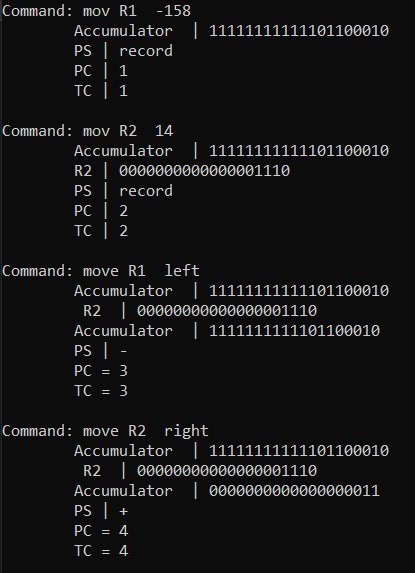
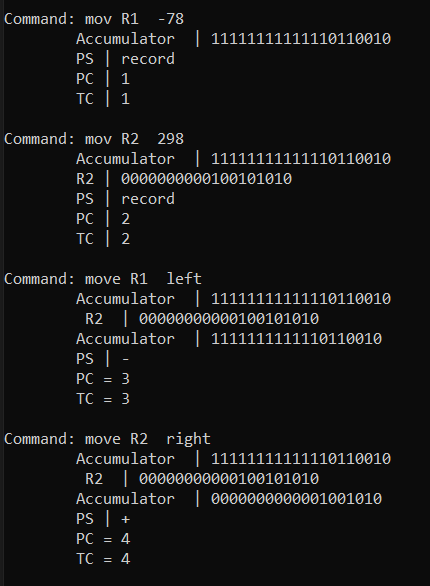
***Звіт***

**«Імітаційна модель процесора*»***

**Мета**

Суть даної лабораторної роботи є продемонструвати як працює процесор та розібратися як працюють регістри. При чому, кожний з регістрів має 16 бітів (в кожного варіанту своя бітність), а також необхідно написати операцію, яка виконує певну дію над регістрами (залежить від варіанту). Також, можливо, що один з регістрів буде акумулятором і необхідно буде зберігати результат саме в ньому. (повний варіант 1.5.5 одноадресність, 1-й операнд завжди в акумуляторі, результат команди заноситься в акумулятор,  бітність регістрів/стеку (слово процесора) та операндів команд: 20, кількість одиничних/нульових бітів в залежності від значення *1* чи *0* операнда команди)

**Лістинг програми**

**** 

**Висновок**

Бітність процесора не впливає на алгоритми виконання команд, а впливає лише на вмістимість числових регістрів. Регістри є однією з найважливіших частин процесора. У регістрах зберігається вся інформація, яку обробляє процесор.

Код програми:

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <bitset>

using namespace std;

// 1.5.5

int tact = 0;

void mov(string R, string R1, string R2, string PS, string way, int PC, int TC)

{

int matr[20], i, k, j;

for (i = 0; i < 20; i++)

{

matr[i] = bitset<20>(stoi(R))[19 - i];

}

if (way == "left")

{

k = matr[0];

for (i = 1; i < 20; i++)

{

matr[i - 1] = matr[i];

}

matr[19] = k;

}

else if (way == "right")

{

j = matr[19];

for (i = 19; i > 0; i--)

{

matr[i] = matr[i - 1];

}

matr[0] = j;

}

cout << "\t" << "Accumulator " << " | " << bitset<20>(stoi(R1)) << endl;

cout << "\t" << " R2 " << " | " << bitset<20>(stoi(R2)) << endl;

cout << "\t" << "Accumulator " << " | ";

for (i = 0; i < 19; i++)

{

cout << matr[i];

}

cout << endl;

cout << "\tPS" << " | " << PS << endl;

cout << "\tPC" << " = " << PC << endl;

cout << "\tTC" << " = " << TC << endl;

}

vector <string> operation(string& command)

{

tact++;

vector <string >result = { "","","","" };

int i = 0;

while (command[i] != ' ')

{

result[0] += command[i];

i++;

}

i++;

while (command[i] != ',') {

result[1] += command[i];

i++;

}

while (command[i] != ',') {

result[2] += command[i];

i++;

}

for (int j = i + 1; unsigned(j) < command.length(); ++j) {

result[3] += command[j];

}

return result;

}

int main()

{

ifstream instructions;

instructions.open("D://Labaoc2//lab2.txt");

string command, R1, R2, accumulator, PS;

vector < string > result;

int commandCounter = 0;

while (getline(instructions, command))

{

vector < string > result;

result = operation(command);

if (result[0] == "mov")

{

if (result[1] == "R1")

{

commandCounter++;

R1 = result[3];

cout << "\n" << "Command: ";

int k = 0;

while (k < 4)

{

cout << result[k] << " ";

k++;

}

cout << endl;

cout << "\t" << "Accumulator " << " | " << bitset<20>(stoi(R1)) << endl;

cout << "\t" << "PS" << " | " << "record" << endl;

cout << "\t" << "PC" << " | " << commandCounter << endl;

cout << "\t" << "TC" << " | " << tact << endl;

}

else if (result[1] == "R2")

{

commandCounter++;

R2 = result[3];

cout << "\n" << "Command: ";

int k = 0;

while (k < 4)

{

cout << result[k] << " ";

k++;

}

cout << endl;

cout << "\t" << "Accumulator " << " | " << bitset<20>(stoi(R1)) << endl;

cout << "\t" << "R2" << " | " << bitset<19>(stoi(R2)) << endl;

cout << "\t" << "PS" << " | " << "record" << endl;

cout << "\t" << "PC" << " | " << commandCounter << endl;

cout << "\t" << "TC" << " | " << tact << endl;

}

}

else if (result[0] == "move")

{

if (result[1] == "R1")

{

if (stoi(R1) < 0)

{

PS = "-";

}

else if (stoi(R1) == 0)

{

PS = "0";

}

else

{

PS = "+";

}

string way;

way = result[3];

commandCounter++;

cout << "\n" << "Command: ";

int k = 0;

while (k < 4)

{

cout << result[k] << " ";

k++;

}

cout << endl;

mov(R1, R1, R2, PS, way, commandCounter, tact);

}

if (result[1] == "R2")

{

if (stoi(R2) < 0)

{

PS = "-";

}

else if (stoi(R2) == 0)

{

PS = "0";

}

else

{

PS = "+";

}

string way;

way = result[3];

commandCounter++;

cout << "\n" << "Command: ";

int k = 0;

while (k < 4)

{

cout << result[k] << " ";

k++;

}

cout << endl;

mov(R2, R1, R2, PS, way, commandCounter, tact);

}

}

}

}