Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра системного аналізу та теорії прийняття рішень

Звіт

з лабораторної роботи № 1

«Визначення швидкодії обчислювальної системи»

Виконав

студент(ка) групи К-23 Старостюк Д.О.

Київ – 2023

**Постановка задачі**

Лабораторна робота передбачає розробку спрощеної системи тестів, без побудови генеральної виборки, опитування експертів, проведення статистичних вимірювань.

Необхідно розробити програму, яка вимірює кількість виконуваних базових операцій (команд) за секунду конкретною ОбСист (комп’ютер + ОС +Система програмування). Вимірювання "чистої" команди процесора непотрібне (як і є у реальних програмних комплексах, що типово розробляються на мовах високого рівня, часто навіть на платформенно незалежних) і фактично не має сенсу.

В програмі рееструється значення часу до початку циклу та після і від останнього віднімаємо попереднє.

Програма має демонструвати стабільність вимірювань для серії запусків. Потрібно враховувати, що при роботі на платформі MS Windows під час вимірювання за тестом може початися процес свопінгу системи, тому в такому випадку сусідні вимірювання у серії можуть відрізнятися на порядок. Такі запуски потрібно виключати з розгляду.

**Результати** мають бути представлені у табличній формі з відображенням для кожного тесту:

1) назви команди/операції,

2) типу/формату даних,

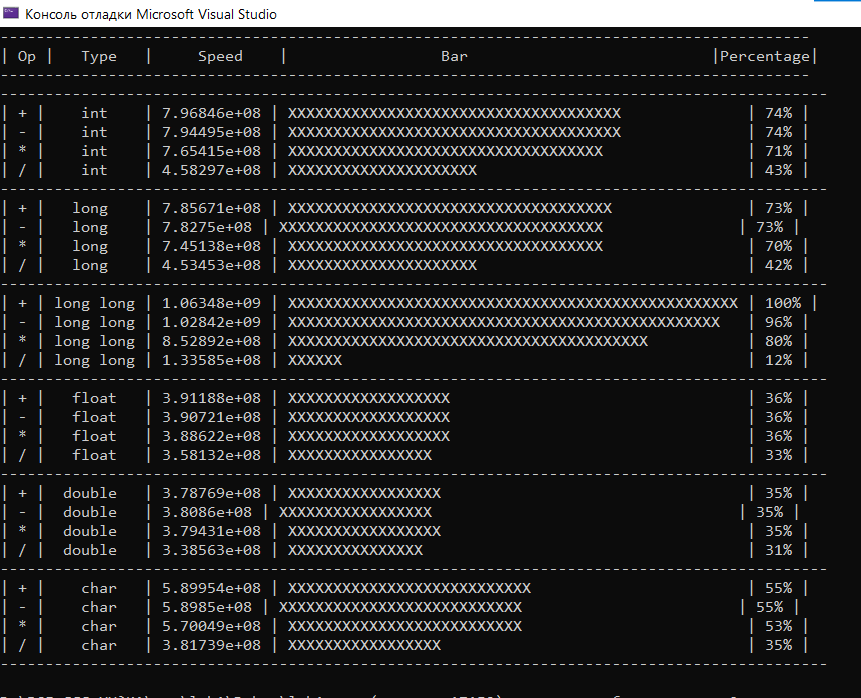
3) кількості операцій за секунду (зайві знаки у мантисі для заданої точності не відображати),

4) лінійної діаграми значення швидкості у відсотках відносно самої швидкої команди/операції, яка

береться за 100%,

5) значення у відсотках (можна округляти до цілого).

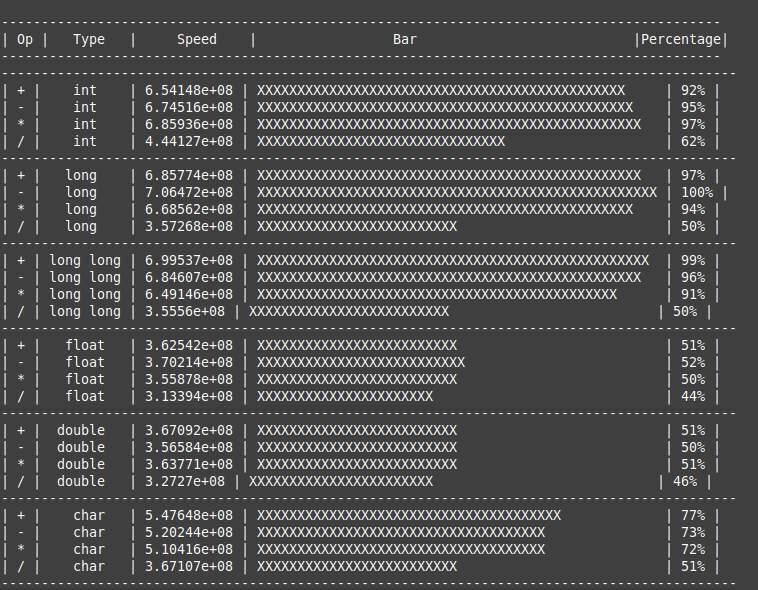
**Результати виконання**



*Процессор :Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz*

*Операційна система: Windows 10 Pro 64-bit*

*Система програмування: Microsoft Visual Studio (Mingw x64)*



***Віртуальна машина***

*Процессор: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz*

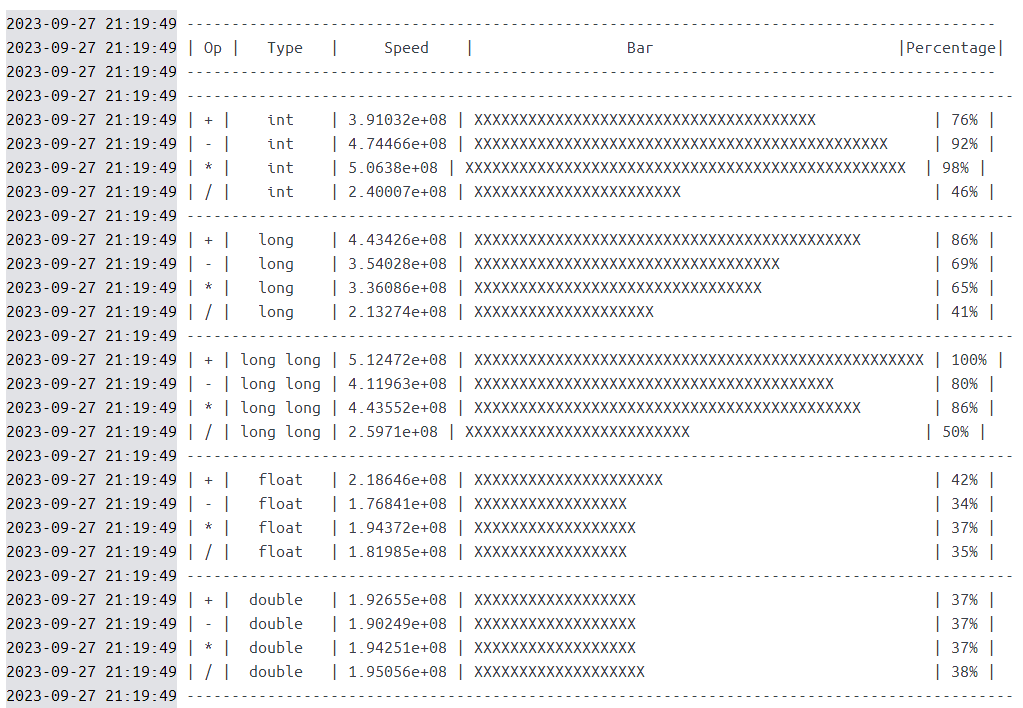
*Операційна система: Windows 10 Home 64-bit*

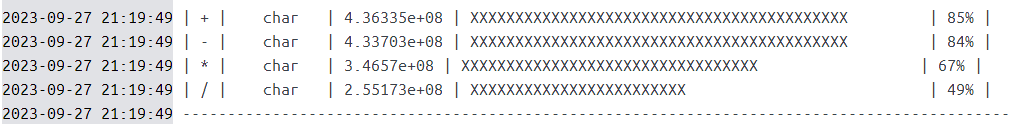
*Віртуальна машина : VirtualBox 6.1.34*

*Операційна система віртуальної машини: Linux Mint 20.2*

**Коментар:**

За умови емуляції іншої ОС на віртуальній машині очевидно зменшення щвидкості виконання операцій





*Процессор: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz*

*Операційна система: Windows 10 Pro 64-bit*

*Система контейнерезації: Docker Desktop 4.23.0,*

**Коментар:**

Можна спостерігати, що при запусках в звичайній ОбСист і в Linux-контейнері швидкість операцій майже однакова, а іноді навіть кращий результат ( похибка+-2%)

**Висновки:**

Використання Dockera є більш зручним та вігідним ніж використання віртуальної машини.

В деяких випадках результати досить дивні, операція множення швидша за очікуване додавання,

**Код**

#include <iostream>

#include <chrono>

#include <string>

using namespace std;

using namespace chrono;

const long long op\_am\_ = 10000000;

template <typename T>

T empty() {

T a1, a2 = static\_cast<T>(2), a3 = static\_cast<T>(3), a4 = static\_cast<T>(4), a5 = static\_cast<T>(5), a6 = static\_cast<T>(6), a7 = static\_cast<T>(7), a8 = static\_cast<T>(8), a9 = static\_cast<T>(9), a10 = static\_cast<T>(10);

auto begin = high\_resolution\_clock::now();

for (long long i = 0; i < op\_am\_; i++) {

a1 = a2, a2 = a3, a3 = a4, a4 = a5, a5 = a6, a6 = a7, a7 = a8, a8 = a9, a9 = a10, a10 = a1;

}

auto end = high\_resolution\_clock::now();

return (end - begin).count();

}

template <typename T>

T perform\_operation(const T& param1, const T& param2, long long t, char operation) {

T a1 = param1, a2 = param2, a3, a4, a5, a6, a7, a8, a9, a10;

auto begin = high\_resolution\_clock::now();

for (long long i = 0; i < op\_am\_; i++) {

switch (operation) {

case '+':

a3 = a1 + a2;

break;

case '-':

a3 = a1 - a2;

break;

case '\*':

a3 = a1 \* a2;

break;

case '/':

a3 = a1 / a2;

break;

}

a4 = a2 + a3;

a5 = a3 + a4;

a6 = a4 + a5;

a7 = a5 + a6;

a8 = a6 + a7;

a9 = a7 + a8;

a10 = a8 + a9;

a1 = a9 + a10;

a2 = a10 + a1;

}

auto end = high\_resolution\_clock::now();

return (double)(op\_am\_ \* 1e10 / ((end - begin).count() - t));

}

int main() {

long long time\_int = empty<int>();

long long time\_long = empty<long>();

long long time\_long\_long = empty<long long>();

long long time\_float = empty<float>();

long long time\_double = empty<double>();

long long time\_char = empty<char>();

double data[24];

data[1] = perform\_operation<int>(13859, 2, time\_int, '+');

data[0] = perform\_operation<int>(3, 2, time\_int, '-');

data[2] = perform\_operation<int>(3, 2, time\_int, '\*');

data[3] = perform\_operation<int>(123456789, 4, time\_int, '/');

data[4] = perform\_operation<long>(5589, 3442, time\_long, '+');

data[5] = perform\_operation<long>(8569731, 1473, time\_long, '-');

data[6] = perform\_operation<long>(32, 11, time\_long, '\*');

data[7] = perform\_operation<long>(8500000, 3, time\_long, '/');

data[8] = perform\_operation<long long>(456852, 12659, time\_long\_long, '+');

data[9] = perform\_operation<long long>(55873695, 1149755, time\_long\_long, '-');

data[10] = perform\_operation<long long>(6597, 8, time\_long\_long, '\*');

data[11] = perform\_operation<long long>(3900001000, 4, time\_long\_long, '/');

data[12] = perform\_operation<float>(1.489, 2.202, time\_float, '+');

data[13] = perform\_operation<float>(1.034, 2.98345, time\_float, '-');

data[14] = perform\_operation<float>(1.5864, 2.123, time\_float, '\*');

data[15] = perform\_operation<float>(9876543.267, 21.645556, time\_float, '/');

data[16] = perform\_operation<double>(1.98765432134, 2.511156555, time\_double, '+');

data[17] = perform\_operation<double>(1.87654345678, 2.656655, time\_double, '-');

data[18] = perform\_operation<double>(1.123457, 2.121103121, time\_double, '\*');

data[19] = perform\_operation<double>(3684843.1231, 18.4738473847, time\_double, '/');

data[20] = perform\_operation<int>(10, 5, time\_char, '+');

data[21] = perform\_operation<int>(20, 4, time\_char, '-');

data[22] = perform\_operation<int>(30, 2, time\_char, '\*');

data[23] = perform\_operation<int>(40, 3, time\_char, '/');

double max = 0;

for (int i = 0; i < 24; i++) {

if (data[i] > max) {

max = data[i];

}

}

char opers[4] = { '+', '-', '\*', '/' };

string types[6] = { " int ", " long ", "long long", " float ", " double ", " char " };

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

cout << "| Op | Type | Speed | Bar |Percentage|" << endl;

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

for (int i = 0; i < 24; i++) {

if (i % 4 == 0) {

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

}

cout << "| " << opers[i % 4] << " | " << types[i / 4] << " | " << data[i] << " | ";

int bar\_length = int(50 \* (data[i] / max));

for (int j = 0; j < bar\_length; j++) {

cout << 'X';

}

for (int j = 0; j < 50 - bar\_length; j++) {

cout << ' ';

}

cout << " | " << int(100 \* (data[i] / max)) << "% |" << endl;

}

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

return 0;

}