ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КАФЕДРА ВС

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2  
«Оценка производительности процессора»  
по дисциплине «Архитектура вычислительных систем»

Выполнил: студент гр. ИП-814  
Краснов И. В.

Проверил: ст. преп. Кафедры ВС   
Ткачева Т. А.

Новосибирск 2020

Содержание

[Постановка задачи 3](#_gjdgxs)

[Выполнение работы 5](#_1fob9te)

[Результат работы 7](#_2et92p0)

[Приложение 8](#_tyjcwt)

# Постановка задачи

Задание. Реализовать программу для оценки производительности процессора (benchmark).

1. Написать программу(ы) (benchmark) на языке С/С++/C# для оценки производительности процессора. В качестве набора типовых задач использовать либо минимум 3 функции выполняющих математические вычисления, либо одну функцию по работе с матрицами и векторами данных с несколькими типами данных. Можно использовать готовые функции из математической библиотеки (math.h) [3], библиотеки BLAS [4] (англ. Basic Linear Algebra Курс «Архитектура вычислительных систем». СибГУТИ. 2020 г. Subprograms — базовые подпрограммы линейной алгебры) и/или библиотеки LAPACK [5] (LinearAlgebra PACKage). Обеспечить возможность подать на вход программы общее число испытаний для каждой типовой задачи (минимум 10). Входные данные для типовой задачи сгенерировать случайным образом.

2. С помощью системного таймера (библиотека time.h, функции clock() илиgettimeofday()) или с помощью процессорного регистра счетчика TSC реализовать оценку в секундах среднего времени испытания каждой типовой задачи. Оценить точность и погрешность (абсолютную и относительную) измерения времени (рассчитать дисперсию и среднеквадратическое отклонение).

3. Результаты испытаний в самой программе (или с помощью скрипта) сохранить в файл в формате CSV со следующей структурой: [PModel;Task;OpType;Opt;LNum;InsCount;Timer;AvTime;AbsErr;RelErr;TaskPerf], где PModel – Processor Model, модель процессора, на котором проводятся испытания; Task – название выбранной типовой задачи (например, sin, log, saxpy, dgemv, sgemm и др.); OpType – Operand Type, тип операндов используемых при вычислениях типовой задачи; Opt – Optimisations, используемы ключи оптимизации (None, O1, O2 и др.); LNum – Launch Numer, число испытаний типовой задачи. InsCount – Instruction Count, оценка числа инструкций при выполнении типовой задачи; AvTime – Average Time, среднее время выполнения типовой задачи в секундах; AbsError – Absolute Error, абсолютная погрешность измерения времени в секундах; RelError – Relative Error, относительная погрешность измерения времени в %; TaskPerf – Task Performance, производительность (быстродействие) процессора при выполнении типовой задачи.

3. \* Оценить среднее время испытания каждой типовой задачи с разным типом входных данных (целочисленные, с одинарной и двойной точностью).

3. \*\* Оценить среднее время испытания каждой типовой задачи с оптимизирующими преобразования исходного кода компилятором (ключи –O1, O2, O3 и др.).

3. \*\*\* Оценить и постараться минимизировать накладные расходы(время на вызов функций, влияние загрузки системы и т.п.) при испытании, то есть добиться максимальной точности измерений. 4. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее быстродействие (производительность) для равновероятного использования типовых задач.

4. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее быстродействие (производительность) для равновероятного использования типовых задач.

# Выполнение работы

Инструменты, используемые в ходе работы: cmd, редактор исходного кода Visual Studio Code, MS Word.

*Центральный процессор* — [электронный блок](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) либо [интегральная схема](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0), исполняющая [машинные инструкции](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (код программ), главная часть [аппаратного обеспечения](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) или [программируемого логического контроллера](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80). Иногда называют [микропроцессором](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) или просто процессором.

Одна и та же модель ЦП в разных компьютерах может работать с разной частотой и обеспечивать разную производительность. Чтобы узнать реальную производительность и сравнить ее с другими ЦП этой же модели или других — необходимо использовать специализированные тесты.

В качестве эталонной тестовой программы я выбрал работу с матрицей, а точнее её сортировку.

Основные шаги выполнения работы:

1. Изучение основных показателей эффективности ЭВМ.
2. Выбор типовых задач.
3. Определение необходимого количества операций для выполнения типовых задач. Определение количества инструкций процессора, необходимого для вычисления типовой задачи, производилось путем дизассемблирования исходного кода с использованием Compiler Explorer.
4. Реализация бенчмарки на языке С++.
5. Тестирование написанной программы.
6. Построение сводной диаграммы производительности в зависимости от задач и выбранных исходных параметров испытаний.
7. Оценка среднего быстродействия (производительности) для равновероятного использования типовых задач.
8. Определение среднего времени работы процессора при использовании разных типов данных при вычислениях (int, float, double).
9. Определение среднего времени выполнения решения типовой задачи с использованием разного уровня оптимизации компилятора:

-О1 (базовый уровень оптимизации)

-О2 (использование всех доступных надежных оптимизаций)

# Результат работы

Результат работы представлен на рисунке 1.

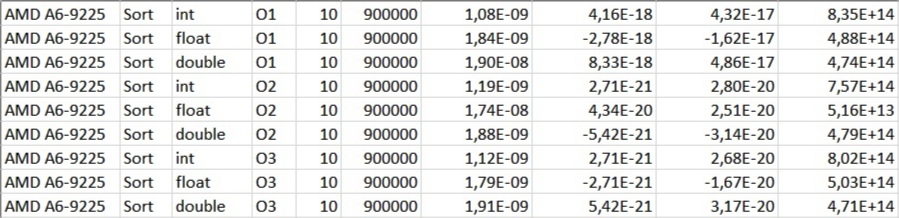


Рисунок 1. Вывод результатов испытаний в CSV файл.

**Приложение**

main.cpp:

#include <cmath>

#include <string>

#include <iomanip>

#include <ctime>

#include <cstring>

#include <iostream>

#include <windows.h>

#include <fstream>

#define PROCESSOR\_TYPE "AMD A6-9225"

using namespace std;

const double eps = 0.000001;

double timeStart = 0;

double\* timeCount;

double AbsErr;

double RelErr;

double dispersion = 0, deviation = 0;

struct **Info** {

string PModel;

string Task;

string OpType;

string Opt;

int LNum;

int InsCount;

double AvTime;

double AbsError;

double RelError;

double TaskPerf;

};

void **bTimer**(char task[], int number\_try)

{

if (strcmp(task, "START") == 0){

timeStart = clock();

} else if(strcmp(task, "STOP") == 0){

timeCount[number\_try] = ((double)(clock() - timeStart)) / CLOCKS\_PER\_SEC;

}

}

void **bAccuracy**(int number\_of\_tests)

{

double temp = 0;

double average = 0;

for (int i = 0; i < number\_of\_tests; i++){

average += timeCount[i];

}

average /= number\_of\_tests;

for (int i = 0; i < number\_of\_tests; i++){

AbsErr += (timeCount[i] - average);

}

AbsErr /= number\_of\_tests;

RelErr = AbsErr / average;

dispersion = 0;

deviation = 0;

for (int i = 0; i < number\_of\_tests; i++){

dispersion += sqrt(timeCount[i] - average);

}

dispersion /= number\_of\_tests;

deviation = sqrt(dispersion);

}

template < typename testT >

float **testSort**(int &number\_try)

{//Сортировка

bool swaped;

int n = 300;

testT \*\*massiv = new testT\* [n];

for (int i = 0; i < n; i++) massiv[i] = new testT [n];

for(int i = 0; i < n; i++)

for(int j = 0; j < n; j++)

massiv[i][j] = rand();

bTimer("START", number\_try);

for(int str = 0; str < n; str++)

{//сортируем строки

for(int i = 1, swap, temp\_n = n; temp\_n > 1; temp\_n--)

{

swaped = false;

for(int j = i; j < temp\_n; j++)

{

if(massiv[str][j - 1] > massiv[str][j])

{

swaped = true;

swap = massiv[str][j];

massiv[str][j] = massiv[str][j - 1];

massiv[str][j - 1] = swap;

}

}

if(swaped == false) break;

}

}

for(int slbc = 0; slbc < n; slbc++)

{//сортируем столбцы

for(int i = 1, swap, temp\_n = n; temp\_n > 1; temp\_n--)

{

swaped = false;

for(int j = i; j < temp\_n; j++)

{

if(massiv[j - 1][slbc] > massiv[j][slbc])

{

swaped = true;

swap = massiv[j][slbc];

massiv[j][slbc] = massiv[j - 1][slbc];

massiv[j - 1][slbc] = swap;

}

}

if(swaped == false) break;

}

}

bTimer("STOP", number\_try);

return (double)(\*timeCount) / CLOCKS\_PER\_SEC;

}

int **main**()

{

SYSTEM\_INFO siSysInfo;

GetSystemInfo(&siSysInfo);

int number\_of\_tests;

string type;

srand(time(0));

cout << endl << "Enter the number of tests: ";

cin >> number\_of\_tests;

cout << endl << "Enter type data: ";

cin >> type;

timeCount = new double[number\_of\_tests];

float avTimeTest = 0;

if(type == "int") for(int i = 0; i < number\_of\_tests; i++) avTimeTest += testSort<int>(*i*);

else if(type == "float") for(int i = 0; i < number\_of\_tests; i++) avTimeTest += testSort<float>(*i*);

else if(type == "double")for(int i = 0; i < number\_of\_tests; i++) avTimeTest += testSort<double>(*i*);

else

{

cout << endl << "Can't found type name";

return 1;

}

avTimeTest /= number\_of\_tests \* 300 \* 300;

double opers = number\_of\_tests \* 300 \* 300;

double taskPerfomance = opers / avTimeTest;

bAccuracy(number\_of\_tests);

Info \*info = new Info;

info->PModel = PROCESSOR\_TYPE;

info->Task = "Sort";

info->OpType = type;

info->Opt = "O1";

info->LNum = number\_of\_tests;

info->InsCount = opers;

info->AvTime = avTimeTest;

info->AbsError = AbsErr;

info->RelError = RelErr;

info->TaskPerf = taskPerfomance;

ofstream file("text.csv", ios::app);

if(!file.is\_open())

{

cout << endl << "Can't open/find file template\_vmf\_end.txt" << endl;

return 1;

}

else

{

file << info->PModel << ";"

<< info->Task << ";"

<< info->OpType << ";"

<< info->Opt << ";"

<< info->LNum << ";"

<< info->InsCount << ";"

<< info->AvTime << ";"

<< info->AbsError << ";"

<< info->RelError << ";"

<< info->TaskPerf << endl;

}

file.close();

cout << endl << "Complete";

}