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-813  
Бурдуковский Илья Александрович

Проверил: ст. преп. Кафедры ВС   
Токмашева Елизавета Ивановна

Новосибирск 2020

Содержание

[Постановка задачи 3](#__RefHeading___Toc865_982383794)

[Выполнение работы 4](#__RefHeading___Toc867_982383794)

[Результат работы 6](#__RefHeading___Toc869_982383794)

[Приложение 8](#__RefHeading___Toc871_982383794)

# 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.
2. С помощью системного таймера 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.
3. Результаты испытаний в самой программе (или с помощью скрипта) сохранить в файл в формате CSV.
4. **\*** Оценить среднее время испытания каждой типовой задачи с разным типом входных данных (целочисленные, с одинарной и двойной точностью).
5. **\*\*** Оценить среднее время испытания каждой типовой задачи с оптимизирующими преобразования исходного кода компилятором (ключи –O1, O2, O3 и др.).
6. **\*\*\*** Оценить и постараться минимизировать накладные расходы(время на вызов функций, влияние загрузки системы и т.п.) при испытании, то есть добиться максимальной точности измерений.
7. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее быстродействие (производительность) для равновероятного использования типовых задач.

# Выполнение работы

В качестве целевого языка для написания тестов производительности был выбран язык C++. В роли типовых задач были выбраны 3 функции нахождения exp, cos и ln с определённой точностью используя ряд Тэйлора.

За само тестирование отвечали 3 функции void Clock\_exp(), void Clock\_cos(), void Clock\_ln(). Каждая соответствовала своей типовой задаче. Последовательность действий функций:

1. Получение определённой константы на вход
2. Начало тестирования
3. Засечь время старта испытаний с помощью системного таймера.
4. Непосредственное выполнение испытаний.
5. Засечь время окончания испытаний.
6. Объединение времени выполнений всех испытаний для получения времени выполнения теста
7. Сохранить результат для данного запуска.
8. Расчёт абсолютной и относительной погрешностей.

После выполнения тестов для получения модели процессора использовались команды bash, изученные в предыдущей лабораторной работе, а также функция popen(), позволяющая получить результат работы bash команд не используя bash-скрипт.

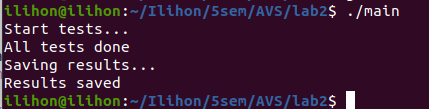
Затем все необходимые данные сохраняются в csv файл используя базовую библиотеку для работы с файлами:

1. создаётся объект класса ofstream
2. объект связывается с файлом “bench result.csv”
3. устанавливается режим для открытия и записи в конец файла
4. все данные записываются в определённом стандарте scv файлов

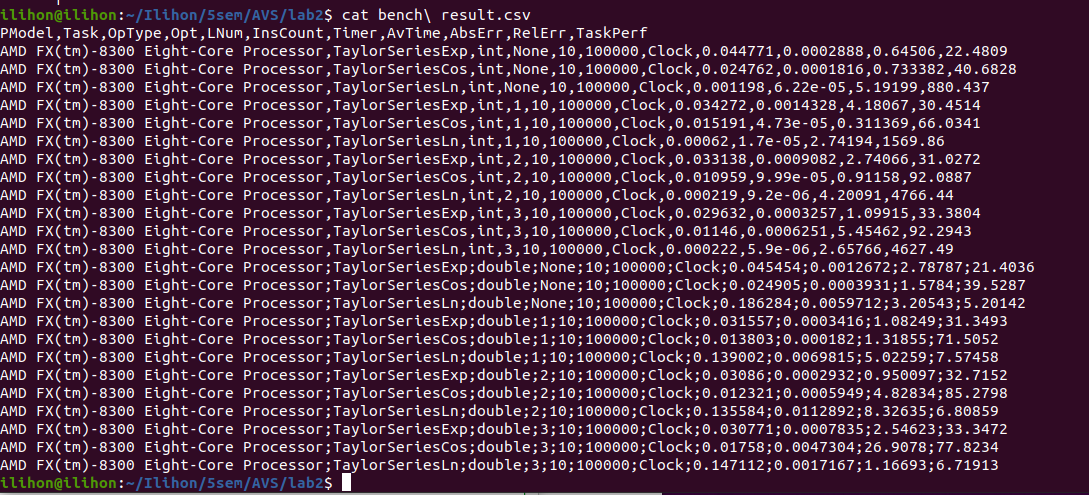
Чтобы оценить среднее время испытания каждой типовой задачи с разным типом входных данных и ключами оптимизации программа выполнялась несколько раз.

# Результат работы

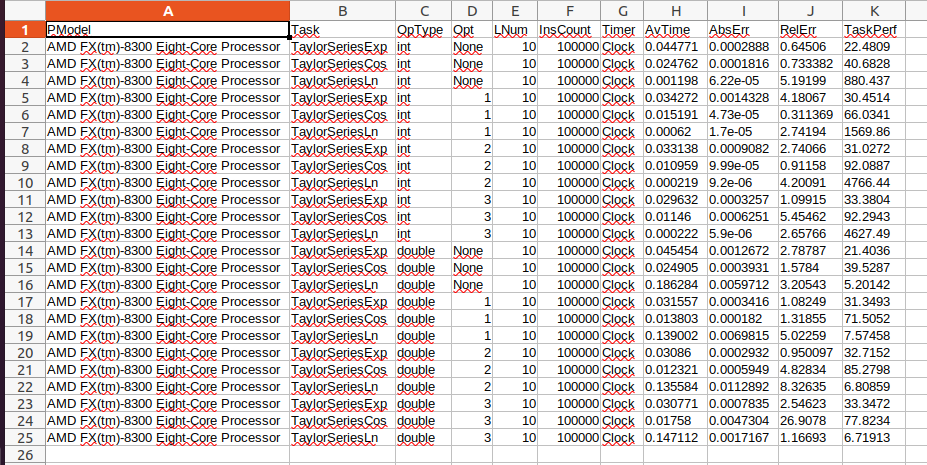
**Запуск программы**

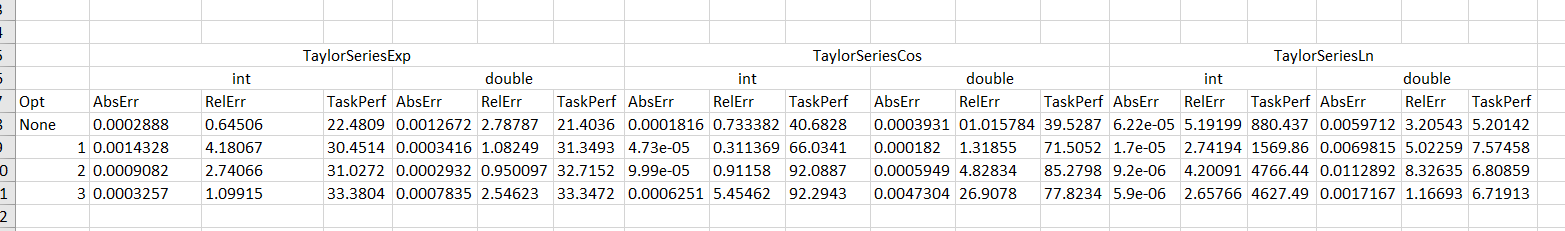


**Просмотр scv файла в консоли**

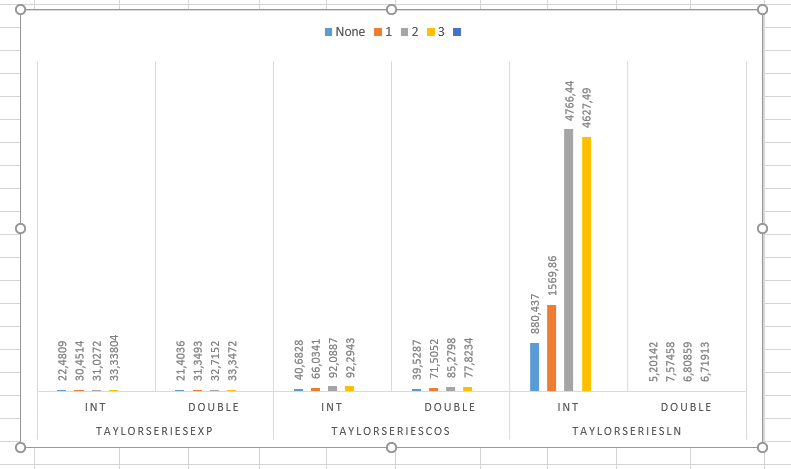


**Импорт scv файла в таблицу libre office**





Сводная таблица с результатами бенчмарки для четырёх уровней оптимизации.



Гистограмма производительности.

Можно отметить следующие интересные особенности:

* Во всех случаях типовых задач, использование ключей оптимизации увеличивает производительность в 1,5 и более раз.
* В TaylorSeriesLn тестах заметная разница в производительности с входными данными типа int и типа double . Результаты прогонов с разными флагами оптимизации сильно отличаются в случае int и слабо в случае double.
* Во всех остальных случаях разница в производительности между разными ключами оптимизации не значительна.

# Приложение

#include <iostream>

#include <time.h>

#include <fstream>

#include <math.h>

#define eps 0.00000000000001

#define N 100000

#define M 10

using namespace std;

double exp\_(const int x)

{

double s = 1;

double n = 1;

double a = 1;

while (1)

{

a = a \* x / n;

if (fabs(a) <= eps) break;

s = s + a;

n++;

}

return s;

}

double cos\_(const int x)

{

double s = 0;

double n = 1;

double a = 1;

while (1)

{

s = s + a;

a = a \* (-1)\*x\*x/((2 \* n - 1) \* (2 \* n));

if (fabs(a) <= eps) break;

n++;

}

return s;

}

double ln\_(const int x)

{

double s = 0;

double n = 1;

double a = (x-1)/x;

while (1)

{

s = s + a;

a = a \* n \* (x - 1)/((n + 1)\*x);

if (fabs(a) <= eps) break;

n++;

}

return s;

}

void Clock\_exp(const int x, double &absolute, double &relative, double &result);

void Clock\_cos(const int x, double &absolute, double &relative, double &result);

void Clock\_ln(const int x, double &absolute, double &relative, double &result);

int main()

{

const int x =5;

cout.precision(15);

double absolute\_exp = 0;

double relative\_exp = 0;

double result\_exp = 0;

double absolute\_cos = 0;

double relative\_cos = 0;

double result\_cos = 0;

double absolute\_ln = 0;

double relative\_ln = 0;

double result\_ln = 0;

cout << "Start tests..." << endl;

Clock\_exp(x, absolute\_exp, relative\_exp, result\_exp);

Clock\_cos(x, absolute\_cos, relative\_cos, result\_cos);

Clock\_ln(x, absolute\_ln, relative\_ln, result\_ln);

cout << "All tests done" << endl;

FILE \*process;

char CPU\_name[1024];

process = popen(" lscpu | grep 'Имя модели' | awk '{$1=$2=NULL; print $0}' | tr -d '\n' | sed -e 's/^[[:space:]]\*//' ", "r");

if (process != NULL) {

while (!feof(process)) {

fgets(CPU\_name, sizeof(CPU\_name), process);

}

pclose(process);

}

cout << "Saving results..." << endl;

ofstream benchmark\_output;

benchmark\_output.open("bench result.csv", ios\_base::app);

benchmark\_output << "PModel;Task;OpType;Opt;LNum;InsCount;Timer;AvTime;AbsErr;RelErr;TaskPerf" << endl;

benchmark\_output << CPU\_name << ";TaylorSeriesExp;int;None;" << M << ";" << N << ";Clock;" <<

result\_exp << ";" << absolute\_exp << ";" << relative\_exp << ";" << 1/ result\_exp << endl;

benchmark\_output << CPU\_name << ";TaylorSeriesCos;int;None;" << M << ";" << N << ";Clock;" <<

result\_cos << ";" << absolute\_cos << ";" << relative\_cos << ";" << 1/ result\_cos << endl;

benchmark\_output << CPU\_name << ";TaylorSeriesLn;int;None;" << M << ";" << N << ";Clock;" <<

result\_ln << ";" << absolute\_ln << ";" << relative\_ln << ";" << 1/ result\_ln << endl;

benchmark\_output.close();

cout << "Results saved" << endl;

return 0;

}

void Clock\_exp(const int x, double &absolute, double &relative, double &result) {

unsigned int start, stop;

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

start = (double)clock();

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

exp\_(x);

stop = (double)clock();

result += stop - start;

}

absolute = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC);

relative = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC) / ((double)(stop - start) / CLOCKS\_PER\_SEC) \* 100;

result = result / M / CLOCKS\_PER\_SEC;

}

void Clock\_cos(const int x, double &absolute, double &relative, double &result) {

unsigned int start, stop;

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

start = (double)clock();

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

cos\_(x);

stop = (double)clock();

result += stop - start;

}

absolute = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC);

relative = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC) / ((double)(stop - start) / CLOCKS\_PER\_SEC) \* 100;

result = result / M / CLOCKS\_PER\_SEC;

}

void Clock\_ln(const int x, double &absolute, double &relative, double &result) {

unsigned int start, stop;

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

start = (double)clock();

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

ln\_(x);

stop = (double)clock();

result += stop - start;

}

absolute = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC);

relative = fabs((result / M - (double)(stop - start)) / CLOCKS\_PER\_SEC) / ((double)(stop - start) / CLOCKS\_PER\_SEC) \* 100;

result = result / M / CLOCKS\_PER\_SEC;

}