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. ИВ-823 Шиндель Э.Д.

Проверил: ст. преп. Кафедры ВС Токмашева Е.И.

Новосибирск 2020

## Содержание

[Постановка задачи 3](#_Toc16002)

[Выполнение работы 5](#_Toc16003)

[Результат работы 5](#_Toc16004)

[Приложение 7](#_Toc16005)

# 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;Ta skPerf], 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.).

* 1. \*\*\* 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.
  2. 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: Bash, терминал, текстовый редактор, редактор исходного кода Visual Studio Code.

В качестве тестовой программы я выбрал программу перемножения двух матриц, трудоемкостью 𝑁2 .

Основные шаги выполнения работы:.

1. Реализация программы на языке Си.
2. Проведение тестов
3. Построение диаграммы производительности.

Вывод: по итогам тестов мы видим, что наилучшая оптимизация достигает на ключе O3. Чем выше уровень оптимизации, тем более радикальные изменения компилятор вносит в программу, что положительно сказывается на быстродействии.

# Результат работы

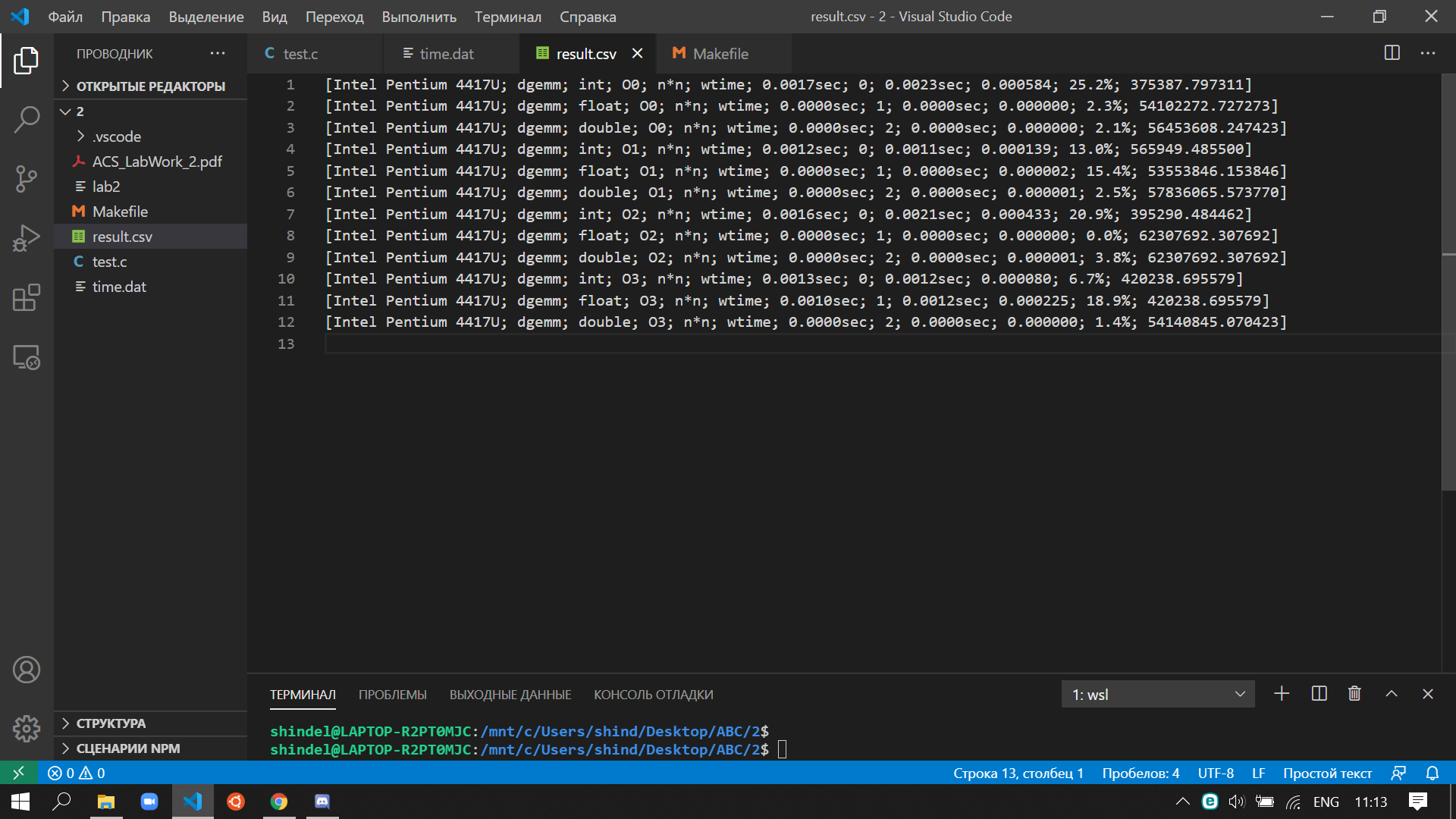


Рисунок 1. Вывод результатов испытаний в csv файл.

# Приложение

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <time.h>

#include <sys/time.h>

double wtime()

{

    struct timeval t;

    gettimeofday(&t, NULL);

    return (double)t.tv\_sec + (double)t.tv\_usec \* 1E-6;

}

void save\_time(double time, int size)

{

    FILE \*f = fopen("time.dat", "a");

    fprintf(f, "%f %d\n", time, size);

    fclose(f);

}

void print\_result(int num, char \*typ)

{

    double sum = 0.0, tt[num], W[num], AE[num], RE[num], PT = 0;

    FILE \*f = fopen("time.dat", "r");

    for (int i = 0; i < num; i++) {

        fscanf(f, "%lf %lf", &tt[i], &W[i]);

        sum += tt[i];

        W[i] \*= W[i];

        W[i] /= tt[i];

        PT += (1 / W[i]);

    }

    fclose(f);

    sum /= num;

    PT = 1 / PT;

    for (int i = 0; i < num; i++) {

        AE[i] = fabs(sum - tt[i]);

        RE[i] = AE[i] / sum \* 100;

    }

    FILE \*r = fopen("result.csv", "a");

    for (int i = 0; i < num; i++) {

        fprintf(r, "[Intel Pentium 4417U; dgemm; %s; O0; n\*n; wtime; %.4fsec; %d; %.4fsec; %f; %.1f%%; %f]\n",

                                                                    typ, tt[i], i, sum, AE[i], RE[i], PT);

    }

    fclose(r);

}

void dgemm\_int(int \*\*A, int \*\*B, int \*\*C, int size)

{

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            for (int k = 0; k < size; k++) {

                C[i][j] += A[k][j] \* B[i][k];

            }

        }

    }

}

void dgemm\_double(double \*\*A, double \*\*B, double \*\*C, int size)

{

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            for (int k; k < size; k++) {

                C[i][j] += A[k][j] \* B[i][k];

            }

        }

    }

}

void dgemm\_float(float \*\*A, float \*\*B, float \*\*C, int size)

{

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            for (int k; k < size; k++) {

                C[i][j] += A[k][j] \* B[i][k];

            }

        }

    }

}

void run\_test(int num, int typ)

{

    int size = rand() % 51 + 50;

    for (int i = 0; i < num; i++) {

        double time = 0.0;

        if (typ == 1) {

            int \*\*iA = (int \*\*) malloc(sizeof(int \*) \* size);

            int \*\*iB = (int \*\*) malloc(sizeof(int \*) \* size);

            int \*\*iC = (int \*\*) malloc(sizeof(int \*) \* size);

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

                iA[i] = (int \*) malloc(sizeof(int) \* size);

                iB[i] = (int \*) malloc(sizeof(int) \* size);

                iC[i] = (int \*) malloc(sizeof(int) \* size);

                for (int j = 0; j < size; j++) {

                    iA[i][j] = rand() % 201 - 100;

                    iB[i][j] = rand() % 201 - 100;

                    iC[i][j] = 0;

                }

            }

            time = wtime();

            dgemm\_int(iA, iB, iC, size);

            time = wtime() - time;

            save\_time(time, size);

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

                free(iA[i]);

                free(iB[i]);

                free(iC[i]);

            }

            free(iA);

            free(iB);

            free(iC);

        } else if (typ == 2) {

            double \*\*dA = (double \*\*) malloc(sizeof(double \*) \* size);

            double \*\*dB = (double \*\*) malloc(sizeof(double \*) \* size);

            double \*\*dC = (double \*\*) malloc(sizeof(double \*) \* size);

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

                dA[i] = (double \*) malloc(sizeof(double) \* size);

                dB[i] = (double \*) malloc(sizeof(double) \* size);

                dC[i] = (double \*) malloc(sizeof(double) \* size);

                for (int j = 0; j < size; j++) {

                    dA[i][j] = (double) (rand() % 201 - 100);

                    dB[i][j] = (double) (rand() % 201 - 100);

                    dC[i][j] = 0.0;

                }

            }

            time = wtime();

            dgemm\_double(dA, dB, dC, size);

            time = wtime() - time;

            save\_time(time, size);

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

                free(dA[i]);

                free(dB[i]);

                free(dC[i]);

            }

            free(dA);

            free(dB);

            free(dC);

        } else {

            float \*\*fA = (float \*\*) malloc(sizeof(float \*) \* size);

            float \*\*fB = (float \*\*) malloc(sizeof(float \*) \* size);

            float \*\*fC = (float \*\*) malloc(sizeof(float \*) \* size);

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

                fA[i] = (float \*) malloc(sizeof(float) \* size);

                fB[i] = (float \*) malloc(sizeof(float) \* size);

                fC[i] = (float \*) malloc(sizeof(float) \* size);

                for (int j = 0; j < size; j++) {

                    fA[i][j] = (float) (rand() % 201 - 100);

                    fB[i][j] = (float) (rand() % 201 - 100);

                    fC[i][j] = 0.0;

                }

            }

            time = wtime();

            dgemm\_float(fA, fB, fC, size);

            time = wtime() - time;

            save\_time(time, size);

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

                free(fA[i]);

                free(fB[i]);

                free(fC[i]);

            }

            free(fA);

            free(fB);

            free(fC);

        }

    }

}

int main(int argc, char \*argv[])

{

    if (argc != 3) {

        printf("ERROR! No arguments\n");

        return 1;

    }

    int num = atoi(argv[1]);

    if (num < 1) {

        printf("ERROR! Argument must be >= 1\n");

        return 1;

    }

    int typ;

    if (!strcmp(argv[2], "int")) typ = 1;

    else if (!strcmp(argv[2], "double")) typ = 2;

    else if (!strcmp(argv[2], "float")) typ = 3;

    else {

        printf("ERROR! Wrong argument\n");

        return 1;

    }

    srand(time(NULL));

    run\_test(num, typ);

    print\_result(num, argv[2]);

    return 0;

}