Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации Сибирский Государственный Университет

Телекоммуникаций и Информатики СибГУТИ

Кафедра Вычислительных систем

Лабораторная работа №2

По дисциплине “Архитектура вычислительных систем”

Выполнил:

Студент группы ИВ-921

Орлова А.А.

Работу проверил:

Ассистент кафедры ВС

Петухова Я.В.

Новосибирск 2021

Оглавление

[Постановка задачи 3](#_Toc84193554)

[Выполнение Работы 5](#_Toc84193555)

[Ход работы со \* 6](#_Toc84193556)

[Результат работы 7](#_Toc84193557)

[Листинг 9](#_Toc84193558)

# Постановка задачи

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] (Linear Algebra PACKage). Обеспечить возможность в качестве аргумента при вызове программы указать 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;InsCount;Timer;Time;LNum;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.);

**InsCount** – Instruction Count, оценка числа инструкций при выполнении типовой задачи;

**Timer** – название функции обращения к таймеру (для измерения времени);

**Time** – время выполнения отдельного испытания;

**LNum** – Launch Numer, номер испытания типовой задачи.

**AvTime** –Average Time, среднее время выполнения типовой задачи из всех испытаний[секунды];

**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.

* 1. \* Оценить среднее время испытания каждой типовой задачи с разным типом входных данных (целочисленные, с одинарной и двойной точностью).
  2. \*\* 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.

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

# Выполнение Работы

ЯиспользовалаязыкС.Вmain()задаетсяколичество видов тестов (test\_size, равное трём), количество испытаний типовых задач 10 исоздаютсямассивыдляхранениясреднеговременивыполнениятиповойзадачи,точностьоценкивремени,погрешности,общеевремявыполнениятеста.Мои типовые задачи — это умножение двух матриц. Также задаетсяфиксированныйразмерматрицы.

Далее в цикле вызывается каждый тест (тесты отличаются друг от другатипом данных, создаваемых матриц). Тесты выполняются черезфункциюbenchmark().Послеэтогооткрываетсяфайлoutput.csv,вкоторыйзаписываютсяданныевсоответствиисзаданием.

Вфункцииbenchmark()вызываетсяmatrix\_int(),matrix\_double()илиmatrix\_float() взависимости от входных данных. Последние три функции работают последующему алгоритму: создаются два массива, которые заполняются рандомнымичислами и третий массив(куда будут записываться результаты умножения двух матриц), который заполняется нулями, после чегозамеряется времяивыполняетсяумножение двух матриц. По окончанию этого снова замеряется время stop =clock().Вычисляетсявремявсекундах.

Выполнив нужную типовую задачу, измеренное время добавляется кдвум переменным (summand1 и summand2), которые нужны для измерениядисперсии. Дисперсия вычисляетсяподаннойформуле:



гдеM(X)—этоматожидание.

Далее находится среднее квадратическое отклонение путем извлечениякорня из дисперсии. Дисперсия — это точность измерения, среднееквадратическое отклонение — абсолютная погрешность. Поделив дисперсиюнасреднее время выполнениятиповойзадачиполучимотносительнуюпогрешность.

# Ход работы со \*

\*

Какбылоописановыше,типоваязадачаимееттритипа,аименно:int,double, float. Скорость выполнения зависит от заданного типа. Это особеннозаметно при использовании оптимизации. Увидеть это можно на скриншотах вразделе «Результатработы».

\*\*

Еслизапускатьпрограммубезключакомпиляциитовремявыполнениятиповых задач в зависимости от типа данных может отличаться. Если запускатьпрограмму с любым ключом оптимизации (O1, O2, O3), то хорошо заметно, чтосреднеевремявыполнениятиповойзадачиуменьшается.

При ключе O1 компилятор попытается сгенерировать быстрый, занимающийменьшеобъемакод,беззатрачиваниянаибольшеговременикомпиляции.

O2активируетнесколькодополнительныхфлаговвдобавоккфлагам,активированныхO1.СпараметромO2,компиляторпопытаетсяувеличитьпроизводительность кода без нарушения размера, и без затрачивания большогоколичества времени компиляции.O3Включаетоптимизации,являющейсядорогостоящейсточкизрениявремени компиляции и потребления памяти, что порой приводит к замедлениюсистемы из-за больших двоичных файлов и увеличения потребления памяти.

\*\*\*

Я постаралась минимизировать накладные расходы на измерение времени длятиповыхзадачтем,чтовремязамеряетсянепосредственнопередипослеумножениядвухматриц.

# Результат работы



Рисунок1:файлoutput.csvпослевыполненияпрограммы

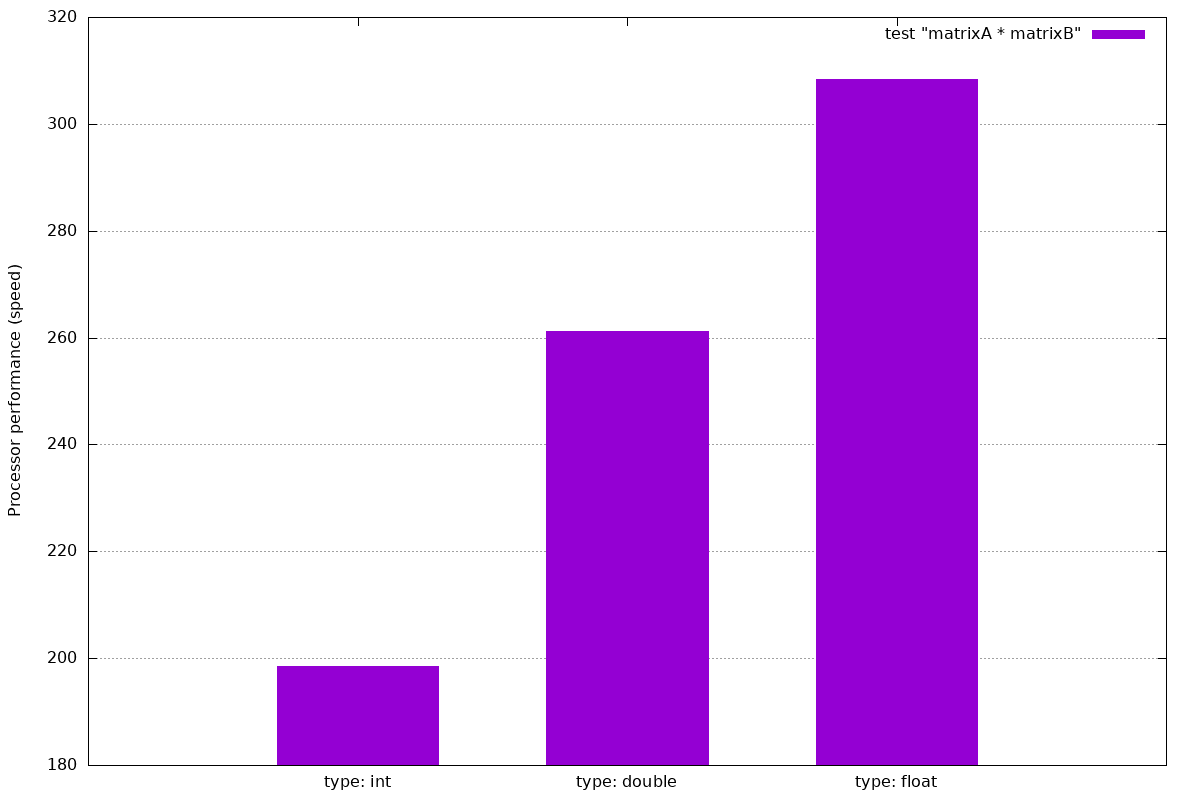


Рисунок2:диаграммапроизводительностибезключейоптимазации

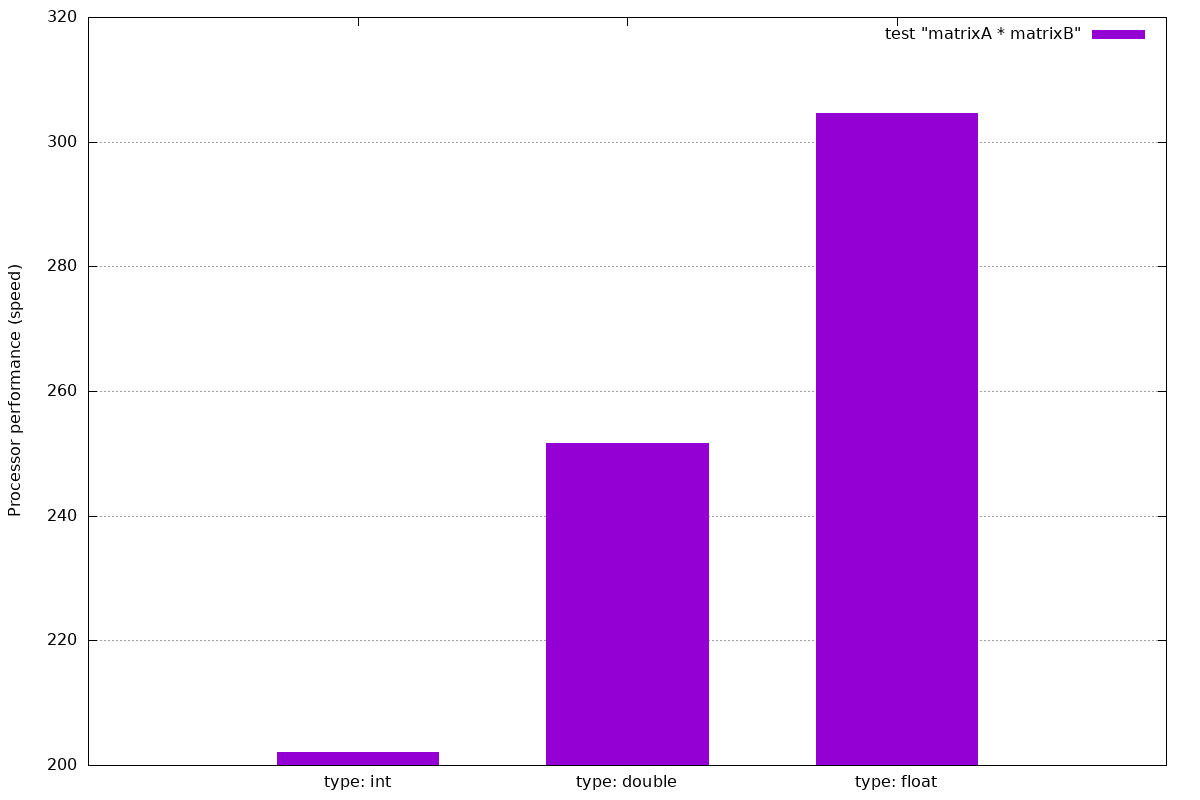


Рисунок3:диаграммапроизводительностисключом-O1

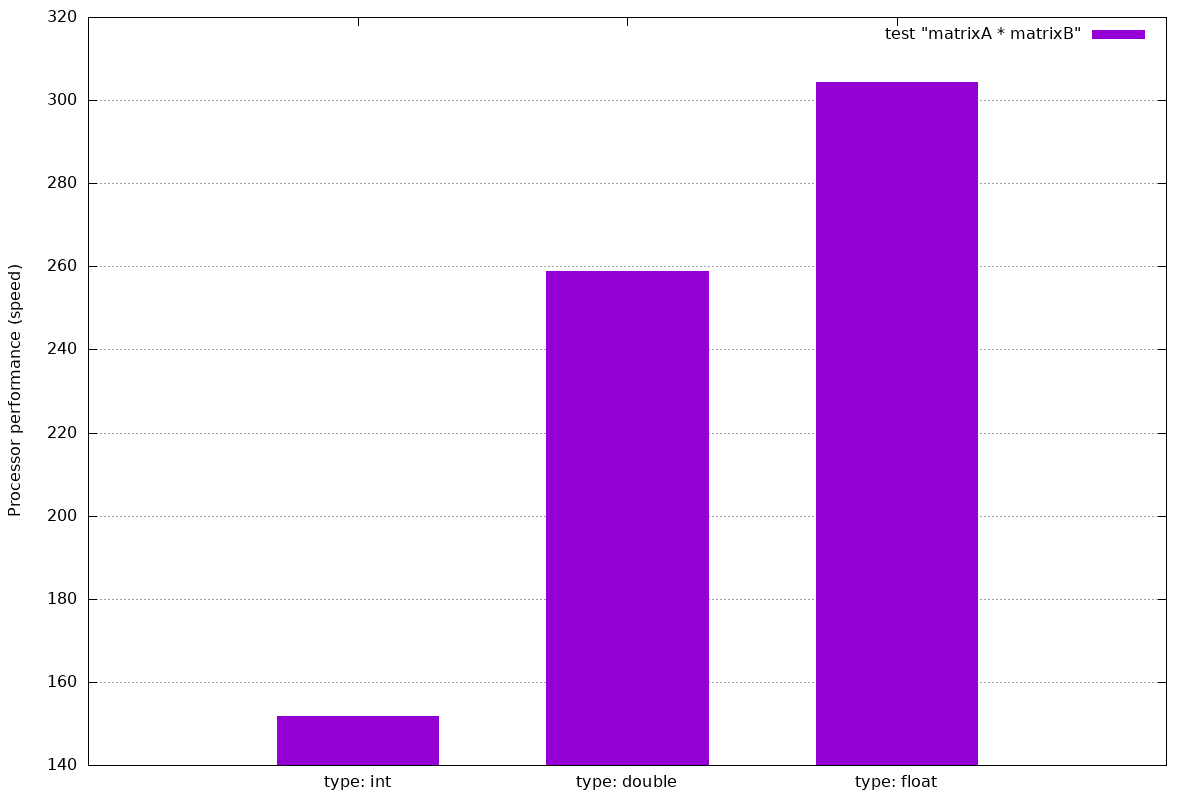


Рисунок4:диаграммапроизводительностисключом-O2

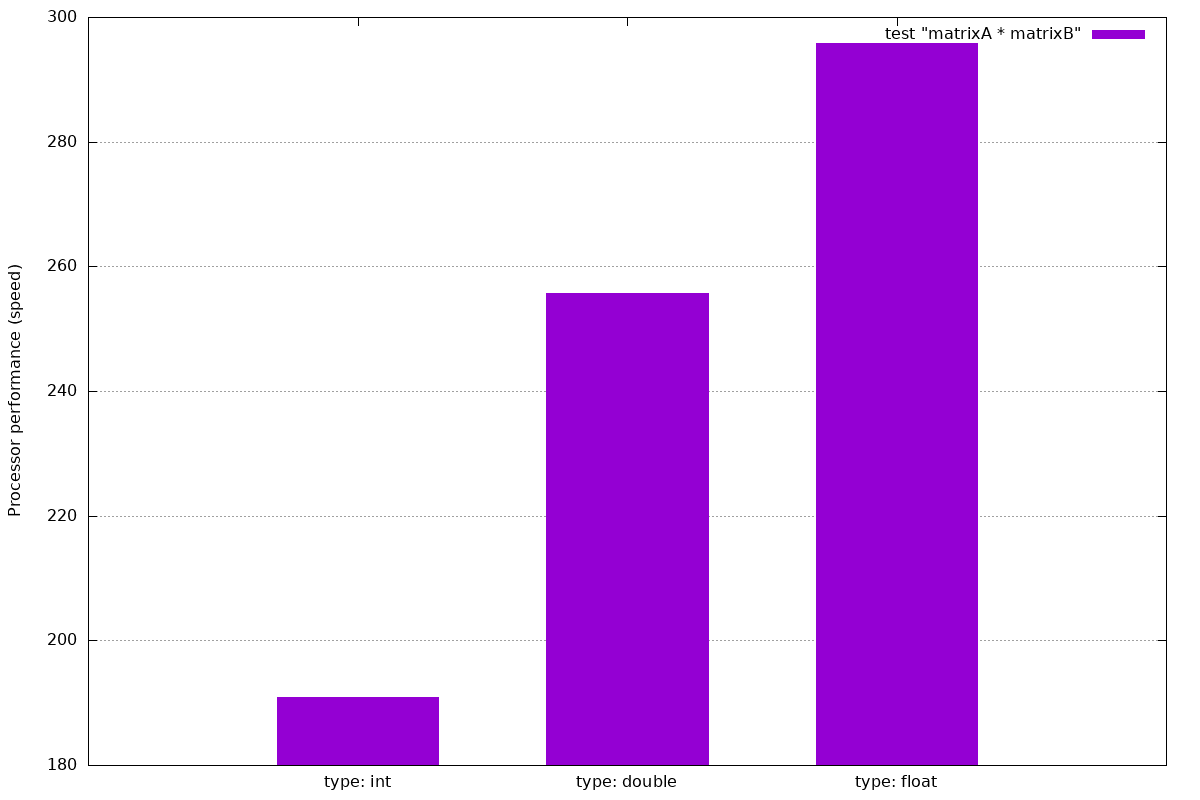


Рисунок5:диаграммапроизводительностисключом-O3

# Листинг

**Файлmain.cpp**

|  |  |
| --- | --- |
|  | #include"foo.h" |
|  | intmain() { |
|  | srand(time(0)); |
|  | int num\_test = 3; |
|  | int n = 10; |
|  | double avg\_time[num\_test], dispersion[num\_test]; |
|  | double abs\_error[num\_test], rel\_error[num\_test]; |
|  | int matrix\_size = 100; |
|  | for (int i = 0; i < num\_test; i++) { |
|  | benchmark(i, n, matrix\_size, avg\_time[i], dispersion[i], abs\_error[i], rel\_error[i]); |
|  | } |
|  | char cpuname[50] = {'\0'}; |
|  | get\_cpu\_name(cpuname); |
|  | FILE \*fout; |
|  | if ((fout = fopen("output.csv", "w")) == NULL) { |
|  | printf("Can't open output.csv \n"); |
|  | return1; |
|  | } |
|  | for (int i = 0; i < num\_test; i++) { |
|  | fprintf(fout, cpuname); |
|  | fprintf(fout, ";"); |
|  | switch (i) { |
|  | case0: |
|  | fprintf(fout, "matrixA \* matrixB;int;None;"); |
|  | break; |
|  | case1: |
|  | fprintf(fout, "matrixA \* matrixB;double;None;"); |
|  | break; |
|  | case2: |
|  | fprintf(fout, "matrixA \* matrixB;float;None;"); |
|  | break; |
|  | default: |
|  | printf("Error writing to file.\n"); |
|  | break; |
|  | } |
|  | fprintf(fout, "clock();"); |
|  | fprintf(fout, "%d;", n); |
|  | fprintf(fout, "%g;", avg\_time[i]); |
|  | fprintf(fout, "%g;", abs\_error[i]); |
|  | fprintf(fout, "%g%%;", rel\_error[i] \* 100); |
|  | fprintf(fout, "%g;", 1 / avg\_time[i]); |
|  | fprintf(fout, "\n"); |
|  | } |
|  | fclose(fout); |
|  | return0; |
|  | } |

**Файлfoo.h**

|  |  |
| --- | --- |
|  |  |
|  | #ifndef FOO  #define FOO |
|  |  |
|  | #include<math.h> |
|  | #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<string.h> |
|  | #include<time.h> |
|  |  |
|  | voidbenchmark(int num\_test, int n, int matrix\_size, double&avg\_time, |
|  | double&dispersion, double&abs\_error, double&rel\_error); |
|  | voidget\_cpu\_name(char cpuname[]); |
|  |  |
|  | #endif |

**Файлfoo.cpp**

|  |  |
| --- | --- |
|  | #include “foo.h”  voidtest1\_1(int n, double&time) { |
|  | int a[n][n], b[n][n], c[n][n]; |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | a[i][j] = rand() % 10; |
|  | b[i][j] = rand() % 10; |
|  | } |
|  | } |
|  | clock\_t start, stop; |
|  | start = clock(); |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | c[i][j] = 0; |
|  | for (int k = 0; k < n; k++) { |
|  | c[i][j] += a[i][k] \* b[k][j]; |
|  | } |
|  | } |
|  | } |
|  | stop = clock(); |
|  | time = ((double)(stop - start)) / CLOCKS\_PER\_SEC; |
|  | } |
|  | voidtest1\_2(int n, double&time) { |
|  | double a[n][n], b[n][n], c[n][n]; |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | a[i][j] = rand() % 10; |
|  | b[i][j] = rand() % 10; |
|  | } |
|  | } |
|  | clock\_t start, stop; |
|  | start = clock(); |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | c[i][j] = 0; |
|  | for (int k = 0; k < n; k++) { |
|  | c[i][j] += a[i][k] \* b[k][j]; |
|  | } |
|  | } |
|  | } |
|  | stop = clock(); |
|  | time = ((double)(stop - start)) / CLOCKS\_PER\_SEC; |
|  | } |
|  |  |
|  | voidtest1\_3(int n, double&time) { |
|  | float a[n][n], b[n][n], c[n][n]; |
|  |  |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | a[i][j] = rand() % 10; |
|  | b[i][j] = rand() % 10; |
|  | } |
|  | } |
|  | clock\_t start, stop; |
|  | start = clock(); |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | c[i][j] = 0; |
|  | for (int k = 0; k < n; k++) { |
|  | c[i][j] += a[i][k] \* b[k][j]; |
|  | } |
|  | } |
|  | } |
|  | stop = clock(); |
|  | time = ((double)(stop - start)) / CLOCKS\_PER\_SEC; |
|  | } |
|  | // Производительность процессора |
|  | voidbenchmark(int num\_test, int n, int matrix\_size, |
|  | double&avg\_time, double&dispersion, |
|  | double&abs\_error, double&rel\_error) |
|  | { |
|  | double summand1 = 0, summand2 = 0; |
|  | double x; |
|  | for (int i = 0; i < n; i++) { |
|  | switch (num\_test) { |
|  | case0: |
|  | test1\_1(matrix\_size, x); |
|  | break; |
|  | case1: |
|  | test1\_2(matrix\_size, x); |
|  | break; |
|  | case2: |
|  | test1\_3(matrix\_size, x); |
|  | break; |
|  | default: |
|  | printf("ERROR: wrong \"num\_test\" in benchmark() \n"); |
|  | break; |
|  | } |
|  | summand1 += x \* x; |
|  | summand2 += x; |
|  | } |
|  | summand1 /= n; |
|  | summand2 /= n; |
|  | avg\_time = summand2; |
|  | summand2 \*= summand2; |
|  | dispersion = summand1 - summand2; |
|  | abs\_error = sqrt(dispersion); |
|  | rel\_error = dispersion / avg\_time; |
|  | } |
|  |  |
|  | voidget\_cpu\_name(char cpuname[]) { |
|  | FILE \*fcpu; |
|  | if ((fcpu = fopen("/proc/cpuinfo", "r")) == NULL) { |
|  | printf("Can't open /proc/cpuinfo \n"); |
|  | return; |
|  | } |
|  | size\_t m = 0; |
|  | char \*line = NULL; |
|  | while (getline(&line, &m, fcpu) >0) { |
|  | if (strstr(line, "model name")) { |
|  | strcpy(cpuname, &line[13]); |
|  | break; |
|  | } |
|  | } |
|  | for (int i = 0; i <50; i++) { |
|  | if (cpuname[i] == '\n') |
|  | cpuname[i] = '\0'; |
|  | } |
|  | fclose(fcpu); |
|  | } |

**Файлdiagram.gpi**

|  |  |
| --- | --- |
|  | #! /usr/bin/gnuplot  #! /usr/bin/gnuplot -persist |
|  | set terminal png font "Verdana,12" size 1200, 800 |
|  | set output "diagram\_O3.png" |
|  | set datafile separator ';' |
|  | set ylabel "Processor performance (speed)" |
|  | set grid ytics |
|  | set xrange [-1:3] |
|  | set xtics ("type: int" 0, "type: double" 1, "type: float" 2,) |
|  | set style data boxes |
|  | set boxwidth 0.6 absolute |
|  | set style fill solid 1 |
|  | plot"output.csv"using10title"test \"matrixA \* matrixB\"" |