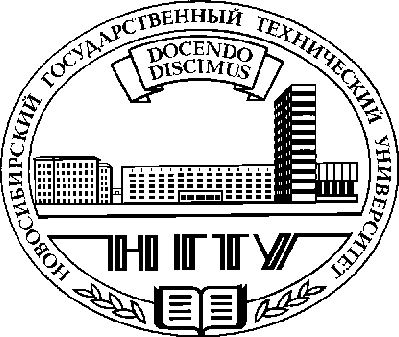
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



**ОТЧЁТ**

**по лабораторной работе №2**

**по дисциплине: «***Вычислительные системы***»**

Выполнили: Преподаватель:

Студенты гр. АММ2-21, АВТФ Перышкова Е.Н.

Антонов С.С.

Арнольд Э.В.

Новосибирск

**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) , библиотеки BLAS и/или библиотеки LAPACK (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]

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.

**Код:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <chrono>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#include <stdio.h>

//#pragma GCC optimize ("O3")

#define m 4000

#define n 4000

double lNum = 10.0;

//

//void Vector(float\* matrix, float\* vector, float\* vectorIt)

//{

// for (int i = 0; i < m; i++)

// {

// vectorIt[i] = 0.0;

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

// vectorIt[i] += matrix[i \* n + j] \* vector[j];

// }

//}

//void Vector(double\* matrix, double\* vector, double\* vectorIt)

//{

// for (int i = 0; i < m; i++)

// {

// vectorIt[i] = 0.0;

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

// vectorIt[i] += matrix[i \* n + j] \* vector[j];

// }

//}

void Vector(int \*matrix, int \*vector, int \*vectorIt)

{

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

{

vectorIt[i] = 0.0;

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

vectorIt[i] += matrix[i \* n + j] \* vector[j];

}

}

double wtime()

{

using std::chrono::system\_clock;

using std::chrono::duration\_cast;

using std::chrono::milliseconds;

auto t = duration\_cast<milliseconds>(system\_clock::now().time\_since\_epoch()).count();

return t;

}

double test()

{

srand(time(0));

int\* matrix, \* vector, \* vectorIt;

matrix = (int\*)malloc(sizeof(\*matrix) \* m \* n);

vector = (int\*)malloc(sizeof(\*vector) \* n);

vectorIt = (int\*)malloc(sizeof(\*vectorIt) \* n);

double t;

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

{

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

matrix[i \* n + j] = rand() % 1000;

}

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

vector[j] = rand() % 1000;

t = wtime();

Vector(matrix, vector, vectorIt);

t = (wtime() - t);

//printf("Elapsed time (serial): %.6f sec.\n", t);

free(matrix);

free(vector);

free(vectorIt);

return t/1000.0;

}

void Math()

{

double dispersion = 0, average = 0, deviation = 0, taskPerf = 0, absError = 0, relError = 0;

int insCount = n \* (m + (n - 1)) + m;

double testAr[10];

double t = test();

for (int i = 0; i < 10; i++)

{

testAr[i] = test();

average += testAr[i];

taskPerf += (insCount / testAr[i]);

dispersion += pow(testAr[i] - average, 2);

}

taskPerf = pow((1 / taskPerf), -1);

deviation = sqrt(dispersion / lNum);

average /= lNum ;//среднеквадратичное отклонение

absError = abs(average - t);

relError = (absError / average) \* 100;

FILE\* file;

file = fopen("itog.cvs", "a");

fprintf(file, "PModel: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz \n");

fprintf(file, "Task: multiplying a matrix by a vector OpType: None \nOpt float\nInsCount %d \n", insCount);

fprintf(file, "Timer: wtime()\nTime: %.6f \nLNum: 10 \nAvTime: %.6f \nAbsErr: %.6f \n", t, average, absError);

fprintf(file, "RelErr: %.6f %%\nTaskPer: %.2f % \n\n", relError, taskPerf);

fclose(file);

}

int main()

{

Math();

return 0;

}

**Результаты работы :**

* PModel: Task: OpType: Opt: InsCount: Timer: Time: LNum: AvTime: AbsErr: RelErr: TaskPer:  
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None int 32000000 wtime() 0.034000 10 0.033200 0.000800 2.409639% 9639928698.75  
    
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None double 32000000 wtime() 0.045000 10 0.045200 0.000200 0.442478% 7085594146.83  
    
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None float 32000000 wtime() 0.045000 10 0.044300 0.000700 1.580135% 7226448509.36  
    
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None floatO1 32000000 wtime() 0.021000 10 0.021200 0.000200 0.943396% 15099567099.57  
    
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None floatO2 32000000 wtime() 0.023000 10 0.020800 0.002200 10.576923% 15397402597.40  
    
  Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz multiplying a matrix by a vector None floatO3 32000000 wtime() 0.021100 10 0.020100 0.001000 4.9751243% 16152300449.75

