

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт  информационных технологий** | **Кафедра информационных систем** |

|  |  |  |
| --- | --- | --- |
| **Основная образовательная программа 09.03.02 «Информационные системы и технологии»**  **Отчет по дисциплине «Архитектура ЭВМ и ВС»** | | |
| **по лабораторной работе №3-5**  **Вариант №3** | | |
| **Тема: «Параллельные вычисления»** | | |
|  | | |
| **Проверил**  **преподаватель кафедры ИС** |  | **Шевляков К. А.** |
|  | подпись |  |
| **Выполнил**  **студент группы ИДБ-22-06** |  | **Мустафаева П.М.** |
| Москва, 2023 | подпись |  |

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc151487146)

[ГЛАВА 1. ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ 4](#_Toc151487147)

[1.1. БЛОК-СХЕМА 4](#_Toc151487148)

[1.2. КОНТРОЛЬНЫЙ РАСЧЁТ 5](#_Toc151487149)

[ГЛАВА 2. РАЗРАБОТКА ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА 6](#_Toc151487150)

[2.1. РАСЧЁТ И РЕАЛИЗАЦИЯ 6](#_Toc151487151)

[2.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 6](#_Toc151487152)

[2.3. ЗАГРУЗКА ПРОЦЕССОРА 7](#_Toc151487153)

[ГЛАВА 3. МНОГОПОТОЧНЫЙ ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ 8](#_Toc151487154)

[3.1. РАСЧЁТ И РЕАЛИЗАЦИЯ 8](#_Toc151487155)

[3.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 9](#_Toc151487156)

[3.3. ЗАГРУЗКА ПРОЦЕССОРА 9](#_Toc151487157)

[ГЛАВА 4. ПАРАЛЛЕЛЬНОЕ ВЫЧИСЛЕНИЕ АРИФМЕТИЧЕСКОГО ВЫРАЖЕНИЯ 10](#_Toc151487158)

[4.1. РАСЧЁТ И РЕАЛИЗАЦИЯ 10](#_Toc151487159)

[4.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 11](#_Toc151487160)

[4.3. ЗАГРУЗКА ПРОЦЕССОРА 11](#_Toc151487161)

[ГЛАВА 5. ПОСТРОЕНИЕ ГРАФИКА 12](#_Toc151487162)

[5.1. ВРЕМЯ ВЫПОЛНЕНИЯ ОДНОПОТОЧНОГО, МНОГОПОТОЧНОГО И ПАРАЛЛЕЛЬНОГО ПРИЛОЖЕНИЙ 12](#_Toc151487163)

[ВЫВОД 13](#_Toc151487164)

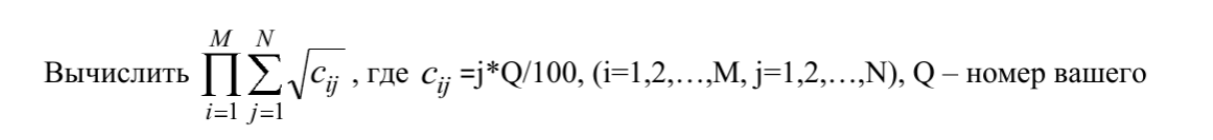
# **ВВЕДЕНИЕ**

Целью работы является получение навыков написания программ с элементами распараллеливания в рамках MIMD архитектуры ВС. Использование различных техник.

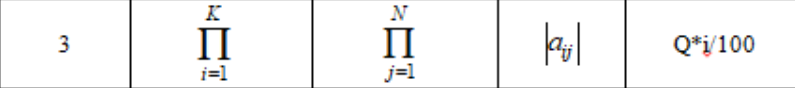
# **ГЛАВА 1. ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ**

**1.1. БЛОК-СХЕМА**

Задача:



Для своего варианта разработать последовательный алгоритм решения задачи и реализовать алгоритм программно, добавив функцию определения времени.

****

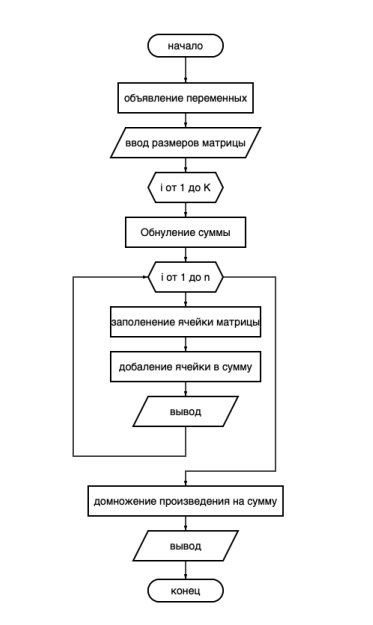


Рис. 1. Блок-схема.

**1.2. КОНТРОЛЬНЫЙ РАСЧЁТ**

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Контрольный расчет для матрицы 3х3 | | | |
| 0.3 | 0.6 | 0.9 | Сумма строки = 1.8 |
| 0.3 | 0.6 | 0.9 | Сумма строки = 1.8 |
| 0.3 | 0.6 | 0.9 | Сумма строки = 1.8 |
| Итоговое произведение выданное программой = 5.832 | | Итоговое произведение посчитанное вручную = 5.832 | |

# **ГЛАВА 2. РАЗРАБОТКА ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА**

**2.1. РАСЧЁТ И РЕАЛИЗАЦИЯ**

Ниже представлен код программы для последовательного алгоритма:

#include <iostream>

#include <cmath>

#include <ctime>

using namespace std;

int main()

{

unsigned long start\_time = clock();

setlocale(LC\_ALL, "Russian");

int N, K;

double Q = 3, multiplication = 1, summ = 0;

cout << "Введите количество строк матрицы: " << endl;

cin >> K;

cout << "Введите количество столбцов матрицы: " << endl;

cin >> N;

double\*\* arr = new double\* [K + 1];

for (int i = 1; i <= K; i++)

{

arr[i] = new double[N + 1];

summ = 0;

for (int j = 1; j <= N; j++)

{

arr[i][j] = Q \* i / 100;

summ += abs(arr[i][j]);

cout << round(arr[i][j] \* 10) / 10 << " ";

}

multiplication \*= summ;

cout << endl;

}

cout << "Итоговое прозведение:" << multiplication << endl;

cout << "Время работы програмы:" << clock() - start\_time << endl;

}

## **2.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

Результат измерений времени выполнения программы при изменении размерности матриц представлен в таблице 2:

Таблица 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5x5 | 10x10 | 50x50 | 100x100 | 150x150 |
| Время выполнения в с | 1500 | 2543 | 3423 | 5522 | 7505 |

**2.3. ЗАГРУЗКА ПРОЦЕССОРА**

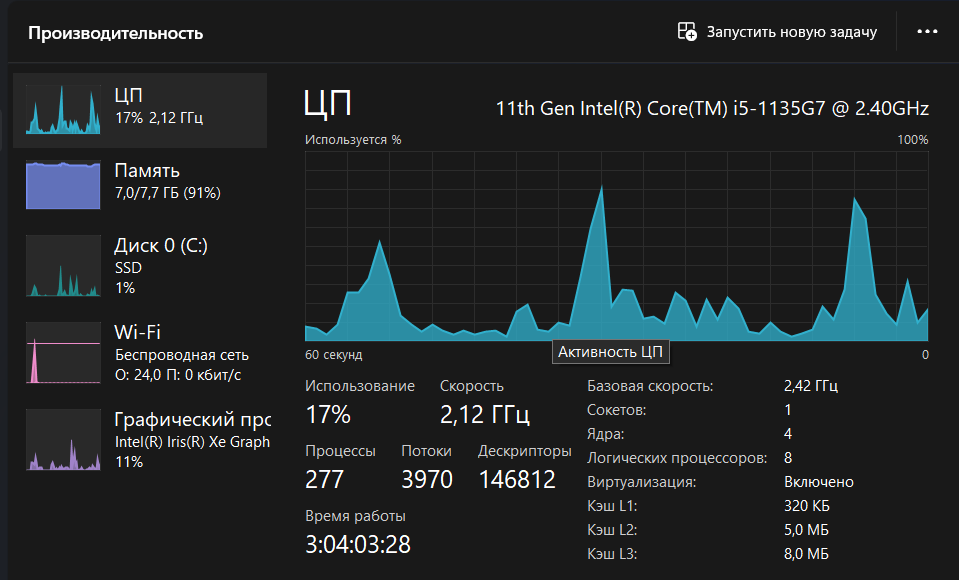
****

Рис. 2. Принтскрин загрузки процессора.

# **ГЛАВА 3. МНОГОПОТОЧНЫЙ ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ**

## **3.1. РАСЧЁТ И РЕАЛИЗАЦИЯ**

Проанализируйте созданный последовательный алгоритм и предложите его многопоточную реализацию.

Ниже представлен код программы:

#include <mutex>

#include <vector>

#include <thread>

#include <iostream>

#include <cmath>

using namespace std;

using namespace this\_thread;

int main()

{

setlocale(LC\_ALL, "Russian");

int N = 0, K = 0;

float Q = 3, multiplication = 1,\* multiplicationPointer = &multiplication;

cout << "Введите количество cтрок матрицы: " << endl;

cin >> K;

cout << "Введите количество столбцов матрицы: " << endl;

cin >> N;

mutex threadMtx;

float\*\* arr = new float\* [K + 1];

unsigned long start\_time = clock();

thread tr([&threadMtx ,N ,K ,Q ,arr ,multiplicationPointer](){

for (int i = 1; i <= K; ++i)

{

arr[i] = new float[N + 1];

float summ = 0;

for (int j = 1; j <= N; ++j){

lock\_guard<mutex> lockGuard(threadMtx);

arr[i][j] = Q \* i / 100;

summ += abs(arr[i][j]);

cout << round(arr[i][j] \* 10) / 10 << " ";

}

\*multiplicationPointer \*= summ;

cout << endl;

}

});

tr.join();

cout << "Итоговое прозведение:" << multiplication << endl;

cout << "Время работы програмы:" << clock() - start\_time << "ms" << endl;

return 0;

}

## **3.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

Результат измерений времени выполнения программы при изменении размерности матриц представлен в таблице 3:

Таблица 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5x5 | 10x10 | 50x50 | 100x100 | 150x150 |
| Время выполнения в с | 16 | 23 | 391 | 1641 | 3462 |

**3.3. ЗАГРУЗКА ПРОЦЕССОРА**

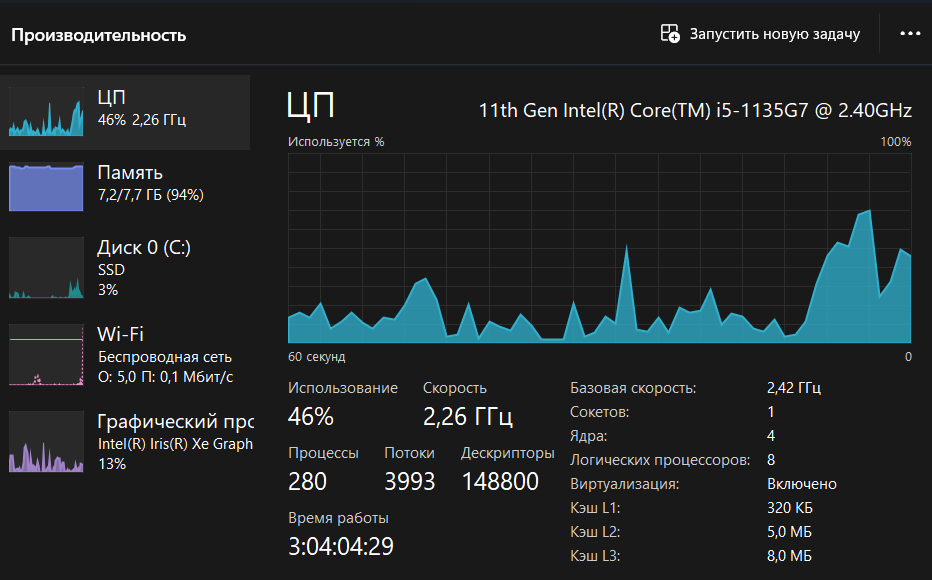
****

Рис. 3. Принтскрин загрузки процессора.

# **ГЛАВА 4. ПАРАЛЛЕЛЬНОЕ ВЫЧИСЛЕНИЕ АРИФМЕТИЧЕСКОГО ВЫРАЖЕНИЯ**

## **4.1. РАСЧЁТ И РЕАЛИЗАЦИЯ**

Аналогично реализуйте параллельное вычисление арифметического выражения.

Ниже представлен код программы:

#include <iostream>

#include <ctime>

#include <omp.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int N = 0, K = 0;

float Q = 3, multiplication = 1, summ = 0;

cout << "Введите количество cтрок матрицы: " << endl;

cin >> K;

cout << "Введите количество столбцов матрицы: " << endl;

cin >> N;

float\*\* arr = new float\* [K + 1];

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

arr[i] = new float[N + 1];

unsigned long start\_time = clock();

#pragma omp parallel

{

#pragma omp for schedule(static)

for (int i = 1; i <= K; i++) {

summ = 0;

for (int j = 1; j <= N; j++) {

arr[i][j] = Q \*i / 100;

cout << round(arr[i][j] \* 10) / 10 << " ";

summ += abs(arr[i][j]);

}

cout << endl;

multiplication \*= summ;

}

}

for (int i = 1; i < K; i++) {

for (int j = 1; j < N; j++) {

cout << round(arr[i][j] \* 10) / 10 << " ";

}

cout << endl;

}

cout << "Итоговое прозведение:" << multiplication << endl;

cout << "Время работы програмы:" << clock() - start\_time << "ms" << endl;

return 0;

}

## **4.2. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

Результат измерений матриц разных размерностей представлен в таблице 4:

Таблица 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5x5 | 10x10 | 50x50 | 100x100 | 150x150 |
| Время выполнения в с | 12 | 32 | 794 | 3354 | 7915 |

## **4.3. ЗАГРУЗКА ПРОЦЕССОРА**

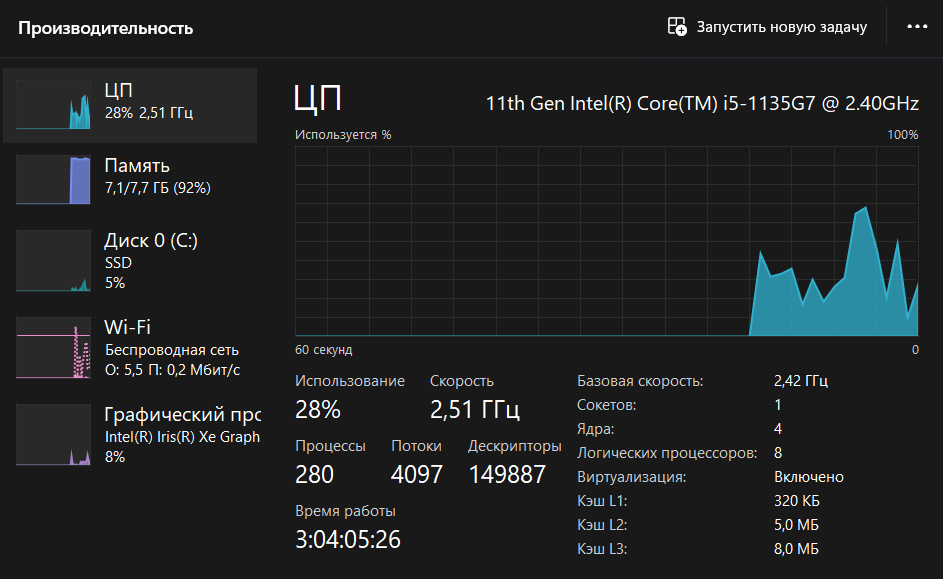
****

Рис. 4. Принтскрин загрузки процессора.

# **ГЛАВА 5. ПОСТРОЕНИЕ ГРАФИКА**

## **5.1. ВРЕМЯ ВЫПОЛНЕНИЯ ОДНОПОТОЧНОГО, МНОГОПОТОЧНОГО И ПАРАЛЛЕЛЬНОГО ПРИЛОЖЕНИЙ**

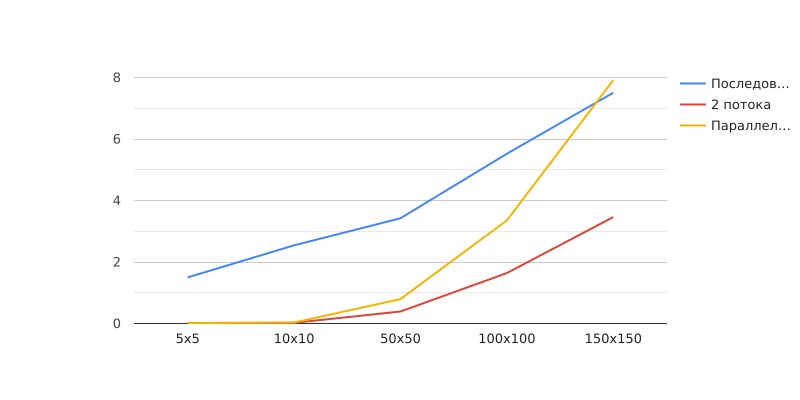
Построить график времени выполнения однопоточного, многопоточного и параллельного приложений в зависимости от изменения размерности матриц. График измерения представлен на рисунке 2:  


Рис.5. График времени выполнения алгоритмов.

# **ВЫВОД**

В ходе лабораторной работы были изучены алгоритмы распараллеливания простых арифметических выражений в рамках MIMD архитектуры ВС, анализа скорости выполнения однопоточного и многопоточного алгоритмов и распараллеливания. Выяснили что распараллеливание ускоряет работу программы при условии, что в ней происходят большие вычисления.