Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

Вариант 2

Отчёт к Лабораторной работе №1

по дисциплине «Параллельные вычисления и системы»

Выполнил:

Студент группы 422–M1

\_\_\_\_Белоус Г.В.

Принял:

К.т.н., Доцент кафедры АОИ

\_\_\_\_Аксёнов С.В.

Оглавление

[1 Введение 3](#_Toc116226139)

[1.1 Цель лабораторной работы 3](#_Toc116226140)

[1.2 Задание на лабораторную работу 3](#_Toc116226141)

[2 Ход выполнения работы 4](#_Toc116226142)

[3 Вывод 8](#_Toc116226143)

# Введение

## 1.1 Цель лабораторной работы

Изучить инструменты многопоточности на CPU и написание программы для построения гауссовой пирамиды.

## **1.2 Задание на лабораторную работу**

Написать программу на языке С с использованием OpenMP (или другого инструмента, позволяющего получить многопоточность на CPU). Получить среднее значение работы процедуры обработки каждого изображения при троекратном перезапуске программы при количестве обрабатывающих потоков от 2 до 16. Сделать выводы об эффективности распараллеливания вычисления с помощью многопоточных вычислений на CPU. Оформить отчет.

Содержание отчета.

1. Титульный лист
2. Задача по полученному варианту (выбор варианта осуществляется по последнему номеру в зачетной книжке, если он равен 0 или 5 – то берем вариант 1, если 2 или 6 – вариант 2, если 3 или 7 то вариант 3, если 4 или 8 - вариант 4, если 5 – вариант 5, если 9 – вариант 6)
3. Листинг программы
4. Примеры входных и выходных файлов
5. Описание результатов тестирования
6. Вывод

Вход: матрица M с размерами 2D×2S, где D-целое ≥10, S – целое ≥10

Выход: гауссова пирамида N уровней, для которой матрица M является начальным уровнем.

Для тестов взять следующий значения:

a) D=12, S=12, N=6

b) D =13, S=12, N=7

c) D = 14, S=13, N =7

# Ход выполнения работы

Программа для обработки изображения была написана на C# с использованием интерфейса программирования приложений Windows Forms.

При запуске программы изображение загружается в форму по указанному пути с использования класса Bitmap, используемого для работы с изображениями, определенными данными пикселями.

Для удобности работы с изображением был написан класс MatrixCompute, который хранит в себе матрицу пикселей и все данные связанные с изображением. В классе MatrixCompute был создан метод MatrixCompilation(), который создает расширенную матрицу N раз, и метод ComputeReducedLevel() вызываемый в методе MatrixCompilation(), рассчитывающий средние ARGB значения для каждого пикселя уменьшенной матрицы на основе изначальной.

Листинг программы можно увидеть в репозитории по ссылке https://github.com/peremichka256/ParallelComputationLabs.git.

Для проверки работы разработанной программы необходимо было взять три изображения с разрешениями a)4096x4096, b)8192x4096 и c)16384x8192. Все изображения были составлены из одной первоначальной картинки и при увеличении разрешения картинка растягивалась с помощью Paint. Первоначальное изображение показано на рисунке 2.1:



Рисунок 2.1 – Изображение загружаемое в приложение

Однако с изображением c)16384x8192 возникла ошибка в выделении памяти для матрицы подобного размера, продемонстрированная на рисунке 2.2:

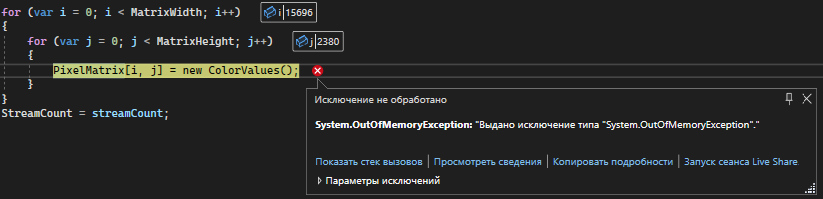


Рисунок 2.2 – Ошибка при выделении памяти 16384x8192 изображения

Полученная картинка после уменьшения изображения a)4096x4096, показана на рисунке 2.3:

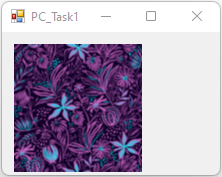


Рисунок 2.3 – Уменьшенное изображение 4096x4096 в 64 раза

Так же полученное изображение после расчетов b)8192x4096, продемонстрировано ниже на рисунке 2.4:

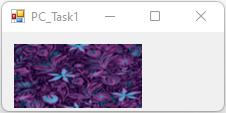


Рисунок 2.4 – Уменьшенное изображение b)8192x4096 в 128 раз

Метод ComputeReducedLevel() был написан таким образом, чтобы в него возможно была передача изначальной матрицы и расчёта определенного куска начиная с заданного индекса, за счёт этого и было выполнено распараллеливание на 2, 4 и 16 потоков, результаты расчетов показаны на рисунках 2.5 и 2.6:

Рисунок 2.5 – График зависимости времени расчётов (в миллисекундах) от количества потоков при трех измерениях изображения a)

Рисунок 2.6 – График зависимости времени расчётов (в миллисекундах) от количества потоков при трех измерениях изображения b)

# Вывод

В ходе выполнения лабораторной работы изучены инструменты многопоточности на CPU и написание программы для построения гауссовой пирамиды с данными из изображений различного расширения.

Исходя из результатов, продемонстрированных на рисунках 2.5 и 2.6 можно сделать вывод о том, что увеличение количества потоков для большого количества расчетов снижает время, затрачиваемое на арифметические операции.

Можно обратить внимание, что разница между расчетами в четыре и шестнадцать потоков не показывает такого большого снижения времени расчётов, как при одном и двух потоках. Скорее всего это связано с тем, что компьютер на котором разрабатывалась данная программа и выполнялись тесты имеет восемь логических процессоров, не позволяющих создание шестнадцати потоков, характеристики процессора показаны на рисунке ниже:

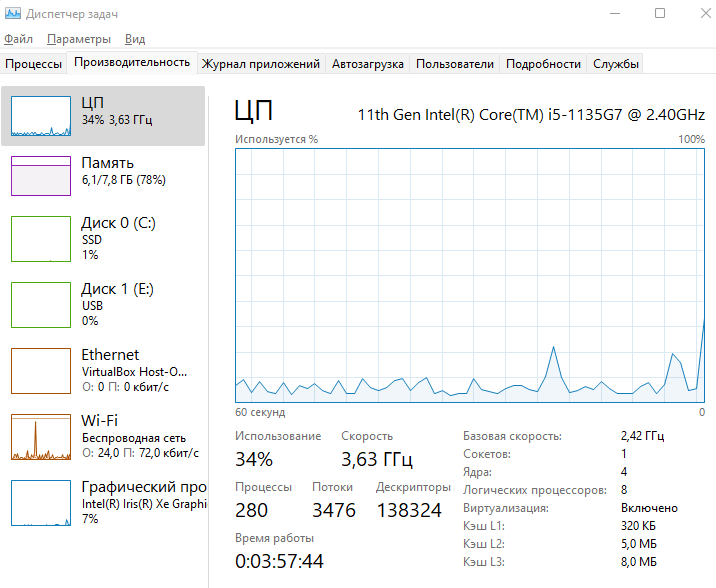


Рисунок 3.1 – Характеристики процессора на котором, проводились расчёты