Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**Управление потоками в ОС. Обеспечение синхронизации между потоками.**

Студент: Гаптулхаков Руслан Рамилевич

Группа: М80 – 208Б-19

Вариант: 6

Преподаватель: Миронов Евгений Сергеевич

Дата:20.04.21

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска программы.

Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы.

Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.

Произвести перемножение 2-ух матриц, содержащих комплексные числа.

1. **Общие сведения о программе**

Программа написана на языке Си в UNIX-подобной операционной системе (Ubuntu). Для компиляции программы требуется указать ключ –pthread. Для запуска программы в качестве аргумента командной строки необходимо указать количество потоков, которые могут быть использованы программой, при этом программа имеет ограничение по максимальному числу потоков.

Программа имеет глобальные переменные, которые хранят размеры матриц, счётчик для потоков. Структура Data хранит 3 сслыку на двумерные массивы.

Программа включает в себя потоковую функцию void\* Produce(void\* ptr), в которой выполняемся перемножение. Каждый поток перемножает одну строку первой матрицы на все столбцы второй. Потоковая функция принимает ссылку на структуру Data.

В программе предусмотрена проверка на системные ошибки – ошибки выделения памяти, ошибки запуска.

1. **Общий метод и алгоритм решения**

Перемножать матрицы можно только такого вида: (mxt)\*(txn) .

Из аргументов командной строки берётся количество потоков, которое может использовать программа. Программа динамически выделятет память под структуру Data и под три матрицы. Заполняются матрицы вручную пользователем в виде: 1 1, где первая цифра действительная часть, а вторая мнимаю часть.

Потоковая функция получает на вход ссылку структуру Data, через которую происходит обращение к матрицам.

Прогрмма выводит перемножаемые матрицы и результат в консоль.

1. **Основные файлы программы**

/\*

Лабораторная работа №3 по курсу "Операционные системы"

Задание: Перемножение двух матриц с использованием потоков, потоки синхронезировать при помощи mutex.

\*/

#define \_POSIX\_C\_SOURCE 199309L

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <pthread.h>

#include <memory.h>

// Global variables

#define NUM\_OF\_THREADS 100

pthread\_mutex\_t mut;

// size of matrix

static int A, B, C, D;

// Determinate the element result's matrix

static int string = -1;

static int column = 0;

// counter for pthread

static int count = 0;

// Struct for complex number representation

typedef struct Complex{

    int real;

    int imaginary;

} Complex;

// Struct with for pointers on three matriсes

typedef struct Data{

    Complex\*\* matrix1;

    Complex\*\* matrix2;

    Complex\*\* result;

} Data;

// The function, which will be transmitted in the threads

void\* Produce(void\* ptr){

    pthread\_mutex\_lock(&mut);

    int str = string;

    int col = column;

    column++;

    pthread\_mutex\_unlock(&mut);

    Data\* myData = (Data\*) ptr;

    for (int k = 0; k < B; ++k){

        myData->result[str][col].real += myData->matrix1[str][k].real\*myData->matrix2[k][col].real - myData->matrix1[str][k].imaginary\*myData->matrix2[k][col].imaginary;

        myData->result[str][col].imaginary += myData->matrix1[str][k].real\*myData->matrix2[k][col].imaginary + myData->matrix1[str][k].imaginary\*myData->matrix2[k][col].real;

    }

    return NULL;

}

int main(int argc, char\* argv[]){

    struct timespec start, finish;

    if (argc != 2) {

        printf("Syntax: ./main Number\_of\_threads\n");

        return 1;

    }

    // restrictions from above for threads

    int num = atoi(argv[1]);

    if(\*argv[1] > NUM\_OF\_THREADS){

        num = NUM\_OF\_THREADS;

    }

    // Create array variables for threads

    pthread\_t\* threads = (pthread\_t\*)calloc(num, sizeof(pthread\_t));

    if (threads == NULL) {

        printf("Can't allocate space for threads\n");

        return 2;

    }

    // Sizes of two matrices

    printf("Enter first matrix size:\n");

    scanf("%d %d", &A, &B);

    printf("Enter second matrix size:\n");

    scanf("%d %d", &C, &D);

    if(B != C){

        printf("It is impossible to multiply matrices!\n");

        return -1;

    }

    // Stracut for transmitted data in thread

    Data\* myData = (Data\*) malloc(sizeof(Data));

    // Initialization

    myData[0].matrix1 = (Complex\*\*) malloc(sizeof(Complex\*)\*A);

    for(int i = 0; i < A; ++i){

        myData[0].matrix1[i] = (Complex\*) malloc(sizeof(Complex)\*B);

    }

    myData[0].matrix2 = (Complex\*\*) malloc(sizeof(Complex\*)\*C);

    for(int i = 0; i < C; ++i){

        myData[0].matrix2[i] = (Complex\*) malloc(sizeof(Complex)\*D);

    }

    myData[0].result = (Complex\*\*) malloc(sizeof(Complex\*)\*A);

    for(int i = 0; i < A; ++i){

        myData[0].result[i] = (Complex\*) malloc(sizeof(Complex)\*D);

    }

    // filling the arrays

    printf("Enter the first matrix:\n");

    for(int i = 0; i < A; ++i){

        for(int j = 0; j < B; ++j){

            scanf("%d", &myData->matrix1[i][j].real);

            scanf("%d",&myData->matrix1[i][j].imaginary);

        }

    }

    printf("Enter the second matrix:\n");

    for(int i = 0; i < C; ++i){

        for(int j = 0; j < D; ++j){

            scanf("%d", &myData->matrix2[i][j].real);

            scanf("%d",&myData->matrix2[i][j].imaginary);

        }

    }

    for(int i = 0; i < A; ++i){

        for(int j = 0; j < D; ++j){

            myData->result[i][j].real = (int) 0;

            myData->result[i][j].imaginary = (int) 0;

        }

    }

    printf("First matrix:\n");

    for(int i = 0; i < A; ++i){

        for(int j = 0; j < B; ++j){

            if(myData->matrix1[i][j].imaginary >= 0)

                printf("%d+%d\*i ", myData->matrix1[i][j].real, myData->matrix1[i][j].imaginary);

            if(myData->result[i][j].imaginary < 0)

                printf("%d-%d\*i ", myData->matrix1[i][j].real, myData->matrix1[i][j].imaginary);

        }

        printf("\n");

    }

    printf("Second matrix:\n");

    for(int i = 0; i < C; ++i){

        for(int j = 0; j < D; ++j){

            if(myData->matrix2[i][j].imaginary >= 0)

                printf("%d+%d\*i ", myData->matrix2[i][j].real, myData->matrix2[i][j].imaginary);

            else

                printf("%d-%d\*i ", myData->matrix2[i][j].real, myData->matrix2[i][j].imaginary);

        }

        printf("\n");

    }

    clock\_gettime(CLOCK\_MONOTONIC, &start);

    pthread\_mutex\_init(&mut, NULL);

    // create threads

    for (int i = 0; i < A; ++i) {

            string++;

            column = 0;

        for(int j = 0; j < D; ++j){

            pthread\_create(&threads[count % num], NULL, &Produce, (void\*) myData);

            count++;

        }

    }

    for (int i = 0; i < num; i++)

        pthread\_join(threads[i], NULL);

    printf("Result matrix:\n");

    for(int i = 0; i < A; ++i){

        for(int j = 0; j < D; ++j){

            if(myData->result[i][j].imaginary >= 0)

                printf("%d+%d\*i ", myData->result[i][j].real, myData->result[i][j].imaginary);

            else

                printf("%d%d\*i ", myData->result[i][j].real, myData->result[i][j].imaginary);

        }

        printf("\n");

    }

    clock\_gettime(CLOCK\_MONOTONIC, &finish);

    double elapsed = (finish.tv\_sec - start.tv\_sec);

    elapsed += (finish.tv\_nsec - start.tv\_nsec) / 1000000000.0;

    pthread\_mutex\_destroy(&mut);

    printf("Time: %.4f seconds\n", elapsed);

    free(myData->matrix1);

    free(myData->matrix2);

    free(myData->result);

    free(myData);

    return 0;

}

1. **Демонстрация работы программы**

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ gcc -Wall -pthread help.c -o help

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 1 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0227 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 2 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0116 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 3 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0084 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 4 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0065 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 5 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0054 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 6 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0067 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 7 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0063 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 8 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0057 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 9 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0053 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 10 < test7.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 0.0050 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 1 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 25.0673 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 2 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 12.5087 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 3 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 8.3784 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 4 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 7.3852 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 5 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.5799 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 6 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 6.0503 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 7 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.4421 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 8 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.6350 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 9 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.7819 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ ./help 10 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.5771 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ strace -f -e trace="%process,write" -o log\_file.txt ./help 10 < test10x10.txt

Enter first matrix size:

Enter second matrix size:

Enter the first matrix:

Enter the second matrix:

Time: 5.4807 seconds

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$ cat log\_file.txt

422 execve("./help", ["./help", "10"], 0x7fffc35ba800 /\* 26 vars \*/) = 0

422 arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffd26af990) = -1 EINVAL (Invalid argument)

422 arch\_prctl(ARCH\_SET\_FS, 0x7f08c0a40740) = 0

422 write(1, "Enter first matrix size:\n", 25) = 25

422 write(1, "Enter second matrix size:\n", 26) = 26

422 write(1, "Enter the first matrix:\n", 24) = 24

422 write(1, "Enter the second matrix:\n", 25) = 25

422 clone(child\_stack=0x7f08c0a2ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[423], tls=0x7f08c0a30700, child\_tidptr=0x7f08c0a309d0) = 423

422 clone(child\_stack=0x7f08c021ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[424], tls=0x7f08c0220700, child\_tidptr=0x7f08c02209d0) = 424

422 clone(child\_stack=0x7f08bfa0ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[425], tls=0x7f08bfa10700, child\_tidptr=0x7f08bfa109d0) = 425

422 clone(child\_stack=0x7f08bf1fffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[426], tls=0x7f08bf200700, child\_tidptr=0x7f08bf2009d0) = 426

422 clone(child\_stack=0x7f08be9effb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[427], tls=0x7f08be9f0700, child\_tidptr=0x7f08be9f09d0) = 427

422 clone(child\_stack=0x7f08be1dffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[428], tls=0x7f08be1e0700, child\_tidptr=0x7f08be1e09d0) = 428

422 clone(child\_stack=0x7f08bd9cffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[429], tls=0x7f08bd9d0700, child\_tidptr=0x7f08bd9d09d0) = 429

422 clone(child\_stack=0x7f08bd1bffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[430], tls=0x7f08bd1c0700, child\_tidptr=0x7f08bd1c09d0) = 430

422 clone(child\_stack=0x7f08bc9affb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[431], tls=0x7f08bc9b0700, child\_tidptr=0x7f08bc9b09d0) = 431

422 clone(child\_stack=0x7f08bc19ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[432], tls=0x7f08bc1a0700, child\_tidptr=0x7f08bc1a09d0) = 432

426 exit(0) = ?

426 +++ exited with 0 +++

430 exit(0) = ?

430 +++ exited with 0 +++

424 exit(0) = ?

424 +++ exited with 0 +++

425 exit(0) = ?

425 +++ exited with 0 +++

427 exit(0) = ?

427 +++ exited with 0 +++

423 exit(0) = ?

423 +++ exited with 0 +++

429 exit(0) = ?

429 +++ exited with 0 +++

432 exit(0) = ?

432 +++ exited with 0 +++

431 exit(0) = ?

431 +++ exited with 0 +++

428 exit(0) = ?

428 +++ exited with 0 +++

422 write(1, "Time: 5.4807 seconds\n", 21) = 21

422 exit\_group(0) = ?

422 +++ exited with 0 +++

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/OS\_Lab\_3\_2020$

1. **Исследование ускорения и эффективности**

Для исследования ускорения и эффективности параллельного перемножения матриц замерим время работы программы для следующих входных данных:

1. 100x100, 100x100
2. 1000x1000, 1000x1000

Тесты будут проводиться на 6 ядерном процессоре intel core i7. На ОС windows/ wsl(ubuntu).

Время работы программы будет замеряться стандартной утилитой time. Нужно учитывать, что время работы может варьироваться в небольших пределах из-за постоянной работы фоновых процессов. Результаты будут занесены в таблицы.

Таблица 1. Тест первый

|  |  |  |  |
| --- | --- | --- | --- |
| Количество потоков (n) | Время работы программы (Tn), сек | Ускорение  (Sn = T1 / Tn) | Эффективность  (Xn = Sn / n) |
| 1 | 0.0133 | - | - |
| 2 | 0.0069 | 1.927 | 0,96 |
| 3 | 0.0046 | 2.891 | 0,96 |
| 4 | 0.0035 | 3.8 | 0,95 |
| 5 | 0.0030 | 4.4 | 0,88 |
| 6 | 0.0039 | 3.4 | 0,56 |
| 7 | 0.0035 | 3.8 | 0,54 |
| 8 | 0.0034 | 3.91 | 0,48 |
| 9 | 0.0031 | 4.29 | 0,47 |
| 10 | 0.0030 | 4.43 | 0,44 |

Таблица 2. Тест 2

|  |  |  |  |
| --- | --- | --- | --- |
| Количество потоков (n) | Время работы программы (Tn), сек | Ускорение  (Sn = T1 / Tn) | Эффективность  (Xn = Sn / n) |
| 1 | 14.9366 | - | - |
| 2 | 6.9849 | 2.1 | 1.05 |
| 3 | 4.7217 | 3.1 | 1.03 |
| 4 | 3.5521 | 4.2 | 1.05 |
| 5 | 3.0825 | 4.9 | 0.98 |
| 6 | 2.8755 | 5.3 | 0.88 |
| 7 | 3.1968 | 4.8 | 0.68 |
| 8 | 3.0099 | 4.9 | 0.61 |
| 9 | 3.2182 | 4.6 | 0.51 |
| 10 | 3.2473 | 4.6 | 0.46 |

Промежуточный вывод по первой версии:

Из тестов видно, что выйгрыша в производительности нет, и смысла писать сложную программу тоже не было. Было сделано несколько вариантов программ. Одна давала потоку просчитать один элемент результирующей матрицы, вторая давала потоку строку результирующей матрицы. Тесты 2 программы были лучше, но параллельность не помогла. Возможно из-за того, что слишком мало вычислений даётся одному потоку.

Промежуточный вывод по второй версии:

По времени мы получаем ускорение при увеличении потоков. Эффективность при для первых пяти потоков 0.9 - 1. Это замечательный результат. На обоих тестах алгоритм показывает почти одинаковую эффективность.

1. **Выводы**

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого на Unix-подобных системах требуется подключить библиотеку pthread.h.

Создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому многопоточность – один из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать парарллельно.

Средствами языка Си можно совершать системные запросы на создание потока, ожидания завершения потока, а также использовать различные примитивы синхронизации.

В данной лабораторной работе был реализован и исследован алгоритм перемножения двух матриц с комплексными числами. Установили, что при использовании 2-5 потоков можно получить выигрыш по времени примерно в полтора - два раза, что значительно ускоряет обработку большого количества чисел. Но при использовании большего количества потоков ускорение не будет большим, так как операционной системе приходится тратить больше времени на выделение памяти под потоки и на их регулирование.