Минобрнауки России
Федеральное государственное автономное образовательное
Учреждение высшего образования
«Санкт-Петербургский государственный электротехнический
Университет им. В.И. Ульянова (Ленина)»
(СПГЭТУ «ЛЭТИ»)
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчет по лабораторной работе №3

на тему: «Процессы и потоки»

по дисциплине «Операционные системы»

Выполнил студент группы 9308: Семенов А.И.

Принял: к.т.н., доцент Тимофеев А.В.

Санкт-Петербург

2021 г.

Содержание

Цель работы

Исследовать механизмы создания и управления процессами и потоками в ОС Windows.

Реализация многопоточного приложения с использованием функций Win32 API

Указания к выполнению

1. Создайте приложение, которое вычисляет число pi с точностью N знаков после запятой по следующей формуле:

$$\pi = \left(\frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \ldots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, \; i = \overline{0, N-1},$$

где N=100000000.

- Используйте распределение итераций блоками (размер блока = 10 * Nстудбилета) по потокам. Сначала каждый поток по очереди получает свой блок итераций, затем тот поток, который заканчивает выполнение своего блока, получает следующий свободный блок итераций. Освободившиеся потоки получают новые блоки итераций до тех пор, пока все блоки не будут исчерпаны;
- Создание потоков выполняйте с помощью функции Win32 API **CreateThread**;
- Для реализации механизма распределения блоков итераций необходимо сразу в начале программы создать необходимое количество потоков в приостановленном состоянии, для освобождения потока из приостановленного состояния используйте функцию Win32 API **ResumeThread**;
- По окончании обработки текущего блока итераций поток не должен завершаться, а должен быть, например, приостановлен с помощью

функции Win32 API SuspendThread. Затем потоку должен быть предоставлен следующий свободный блок итераций, и поток должен быть освобожден, например, с помощью функции Win32 API **ResumeThread**.

2. Произведите замеры времени выполнения приложения для разного числа потоков (1, 2, 4, 8, 12, 16). По результатам измерений постройте график и определите число потоков, при котором достигается наибольшая скорость выполнения. Запротоколируйте результаты в отчет.

Результаты выполнения программы

Программа позволяет настраивать различное количество замеров. При единичном замере получаются различные результаты, представленные на рисунке 1, в связи с чем получить однозначно время вычисления числа Пи нельзя. В связи с этим было проведено 100 замеров и вычислено среднее время вычисления. Используемые процессоры в замерах – Intel Core i5-9400F, имеющий 6 ядер и потоков, работающий на максимальной частоте в 4100 МГц (результаты замера представлены на рисунке 2); Intel Core i7-7700, имеющий 4 ядра и 8 потоков, работающий на максимальной частоте в 4200 МГц (результаты замера – на рисунке 3); Intel Core i5-3230M, имеющий 2 ядра и 4 потока, работающий на максимальной частоте в 3200 МГц (результаты замеров показаны на рисунке 4).

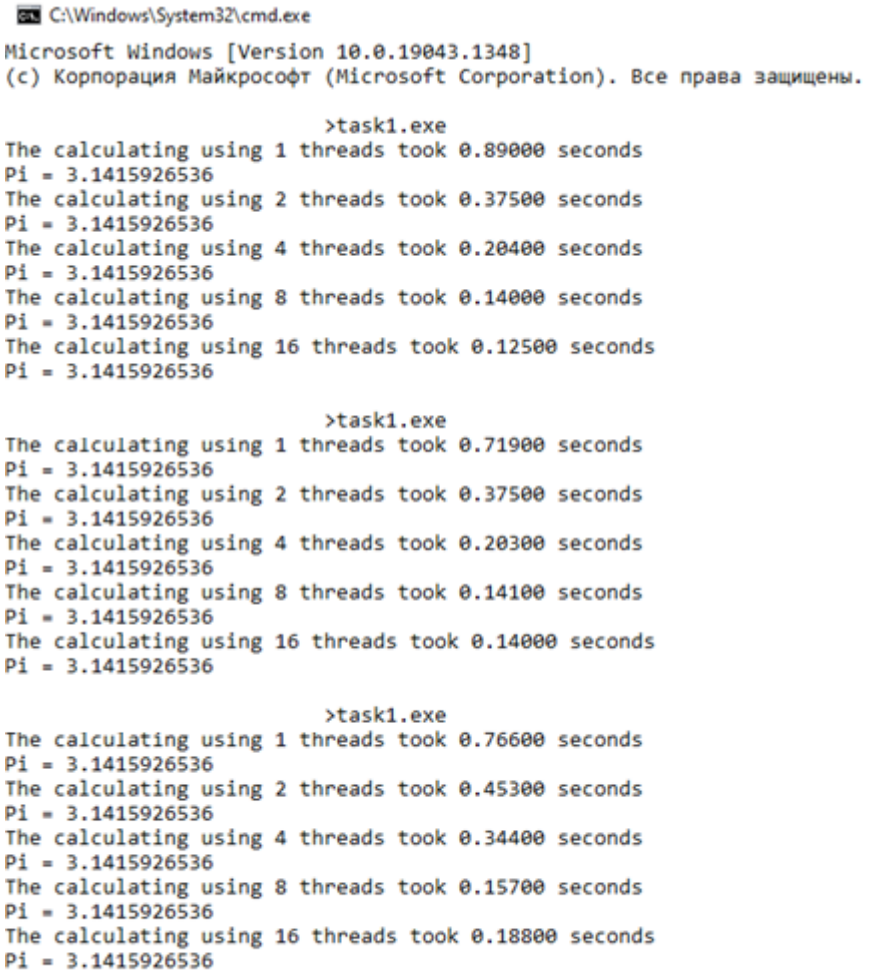

Рисунок 1 – Одиночные замеры вычислений

```
Number of threads - taken time
1 - 0.6884600000
2 - 0.3687500000
4 - 0.1941800000
8 - 0.1354900000
16 - 0.1345200000
```

Рисунок 2 – Результат замеров работы программы i5-9400F

```
Number of threads - taken time
1 - 0.6875500000
2 - 0.3719500000
4 - 0.1882500000
8 - 0.1359000000
16 - 0.1187000000
```

Рисунок 3 – Результат замеров работы программы i7-7700

```
Number of threads - taken time
1 - 1.4094500000
2 - 0.9187500000
4 - 0.7381000000
8 - 0.7385000000
16 - 0.7217500000
```

Рисунок 4 – Результат замеров работы программы i5-3230M

Ниже представлены графики данных выше.

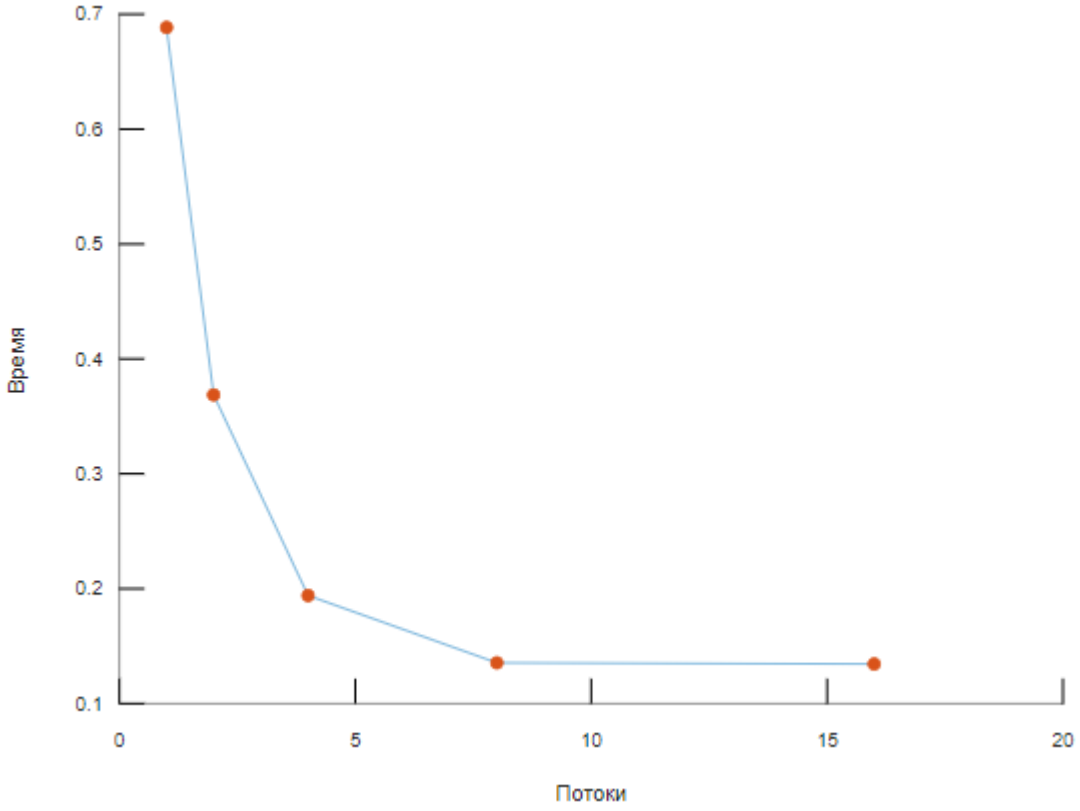

Рисунок 5 – Зависимость времени вычисления от числа потоков i5-9400F

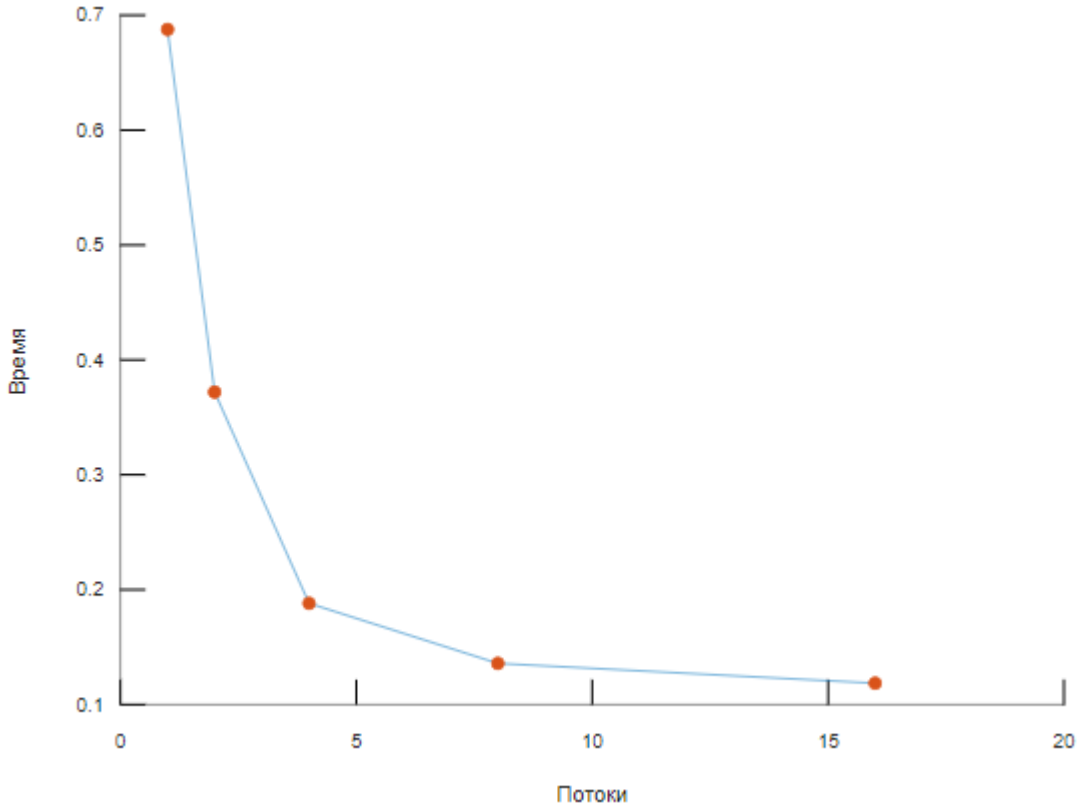

Рисунок 6 – Зависимость времени вычисления от числа потоков i7-7700

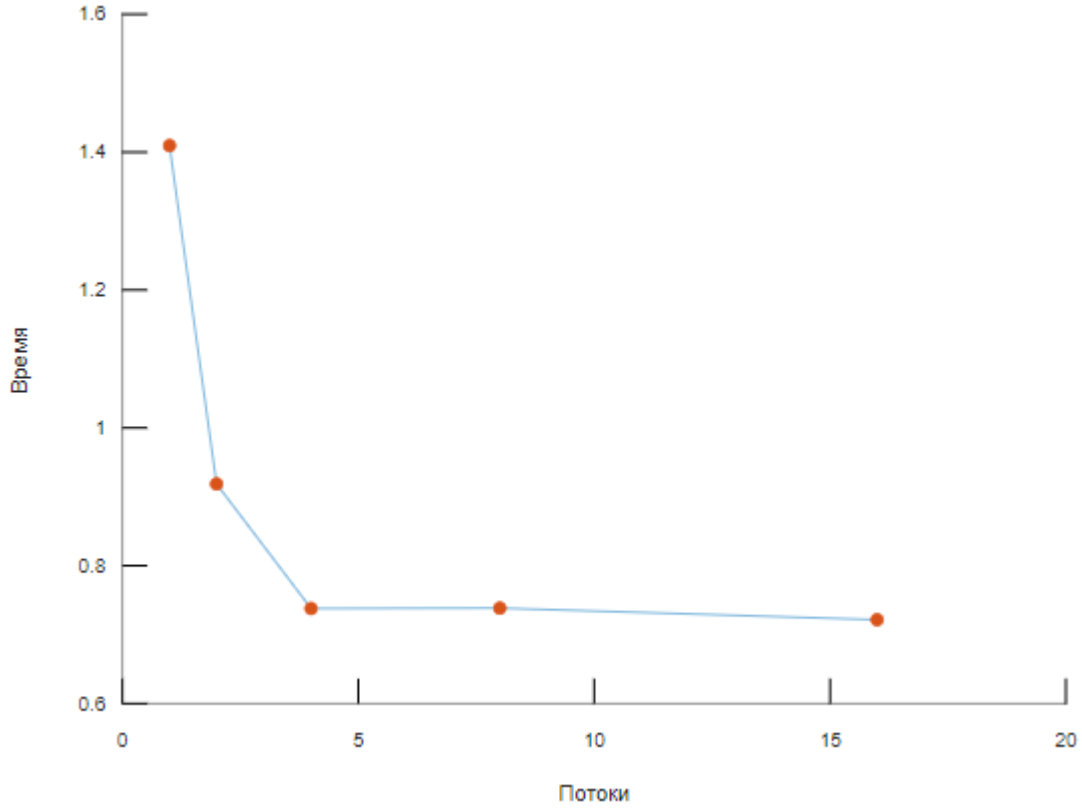

Рисунок 7 – Зависимость времени вычисления от числа потоков i5-3230M

Вывод по заданию

Были изучены механизмы создания и управления процессами и потоками в ОС Windows. По результатам замера можно заметить, что оптимальное число потоков, работающих в процессе для вычисления числа Пи, равно числу потоков на самом процессоре. При увеличении числа потоков сверх тех, что есть у процессора время вычисления практически одинаковое, т.к. во время работы программы процессор мог бы занят другими задачами.

Реализация многопоточного приложения с использованием технологии OpenMP

Указания к выполнению

1. Создайте приложение, которое вычисляет число пи с точностью N знаков после запятой по следующей формуле:

$$\pi = \left(\frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \ldots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, \; i = \overline{0, N-1},$$

где N=100000000.

- Распределите работу по потокам с помощью OpenMP-директивы **for**;
- Используйте динамическое планирование блоками итераций (размер блока = 10 * $N_{студбилета}$).

2. Произведите замеры времени выполнения приложения для разного числа потоков (1, 2, 4, 8, 12, 16). По результатам измерений постройте график и определите число потоков, при котором достигается наибольшая скорость выполнения. Запротоколируйте результаты в отчет, сравните с результатами прошлой работы.

Результаты выполнения программы

Для объективного сравнения с реализацией данной задачи с помощью WinAPI было также проведено 100 замеров вычисления числа Пи посредством технологии OpenMP. Результаты замеров для процессора i5-9400F представлены на рисунке 8.

```
Number of threads - taken time
1 - 0.6859000000
2 - 0.3669900000
4 - 0.1929000000
8 - 0.1387600000
16 - 0.1298200000
```

Рисунок 8 – Результат замеров

Вывод по заданию

Была изучена технология OpenMP, позволяющая реализовывать многотопоточные приложения. Результаты замеров (рис. 8) примерно совпадают с теми, что были получены при использовании WinAPI (рис. 2). Различия составляют не более 5%, которые могли быть вызваны различными паралелльными задачи, выполняемыми на процессоре.

Вывод

Были исследованы механизмы создания и управлениями процессами и потоками в ОС Windows посредством WinAPI и технологии OpenMP.