**Отчёт о проделанной работе**

**Лейсле Александра Геннадьевича, студента группы 21932**

Изучение работы программ на графическом процессоре выполнялось с использованием двух файлов: “double\_Mthread.c” и “float\_Mthread.c”:

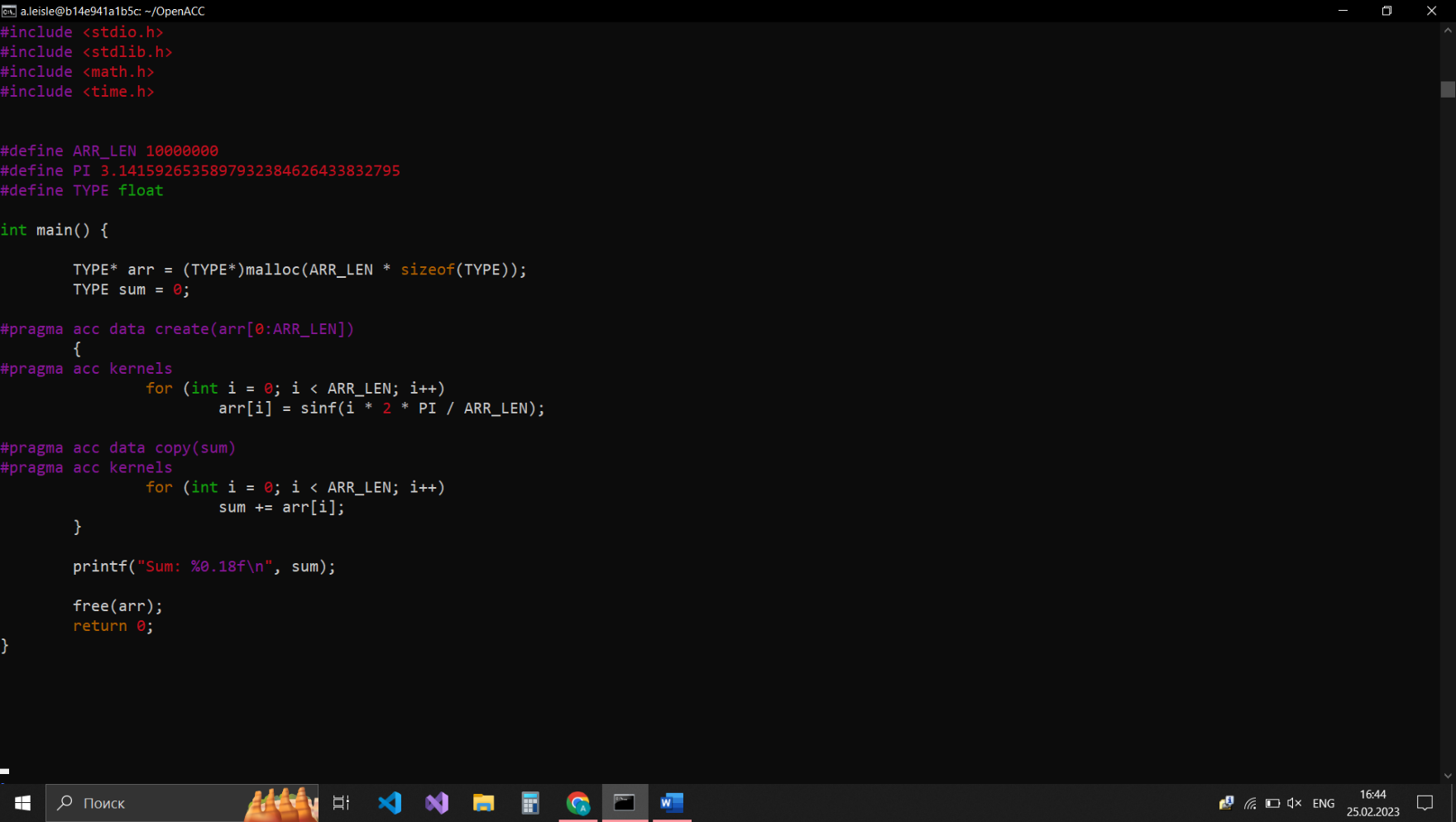


Рис.1 – Программа “float\_Mthread.c”

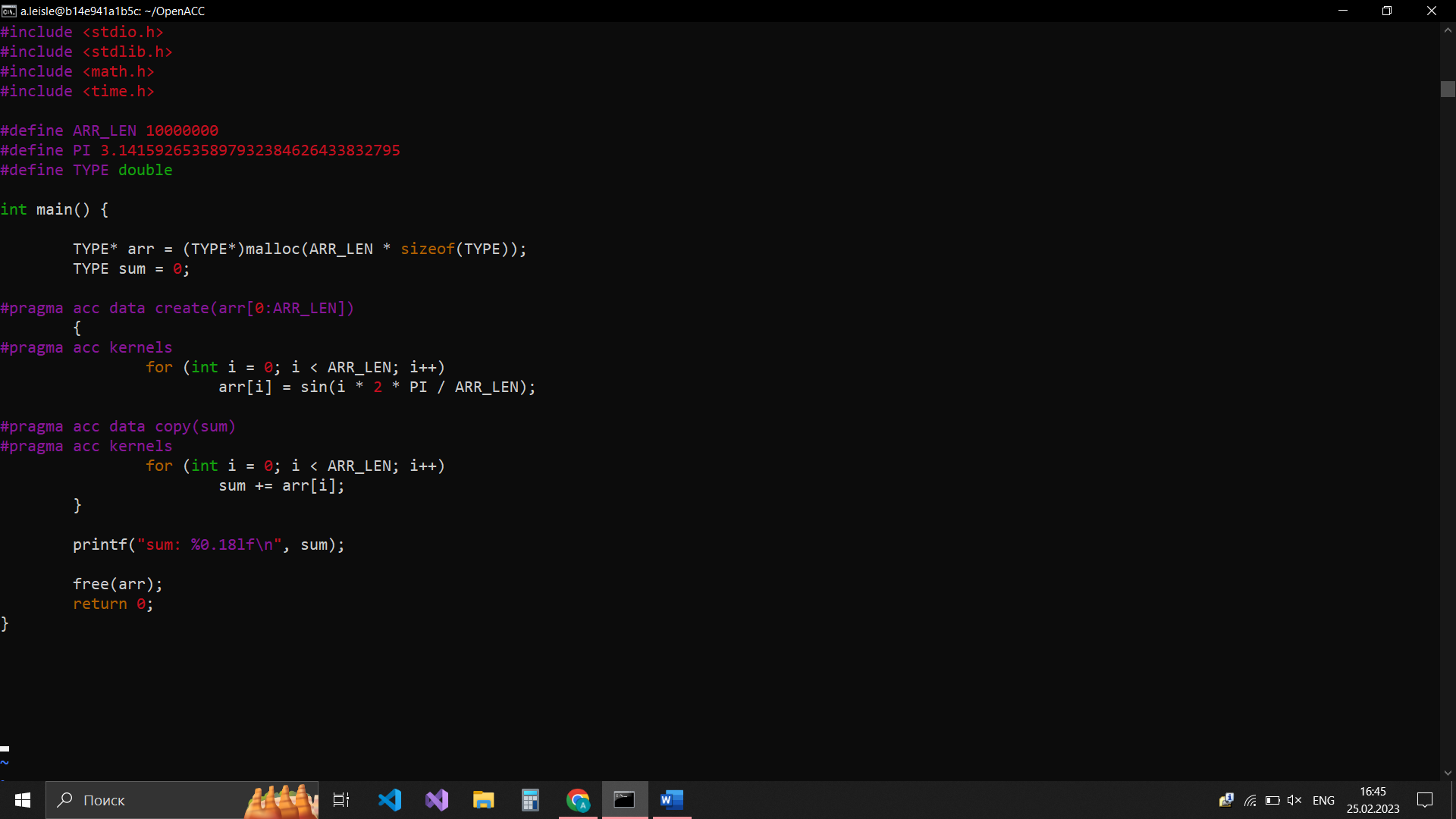


Рис.2 – Программа “double\_Mthread.c”

В каждой программе происходит заполнение массива длинной 107 элементов значениями синуса для одного периода. Затем все значения массива суммируются в переменную sum. Ответ и время выполнения программы выводятся в стандартный поток вывода. Помимо этого, в программе происходит перенаправление выполнения двух циклов на ядра графического процессора.

Компиляция программ осуществлялась на компиляторе pgcc с ключами “-acc -Minfo=accel”. Затем было выполнено профилирование программ.

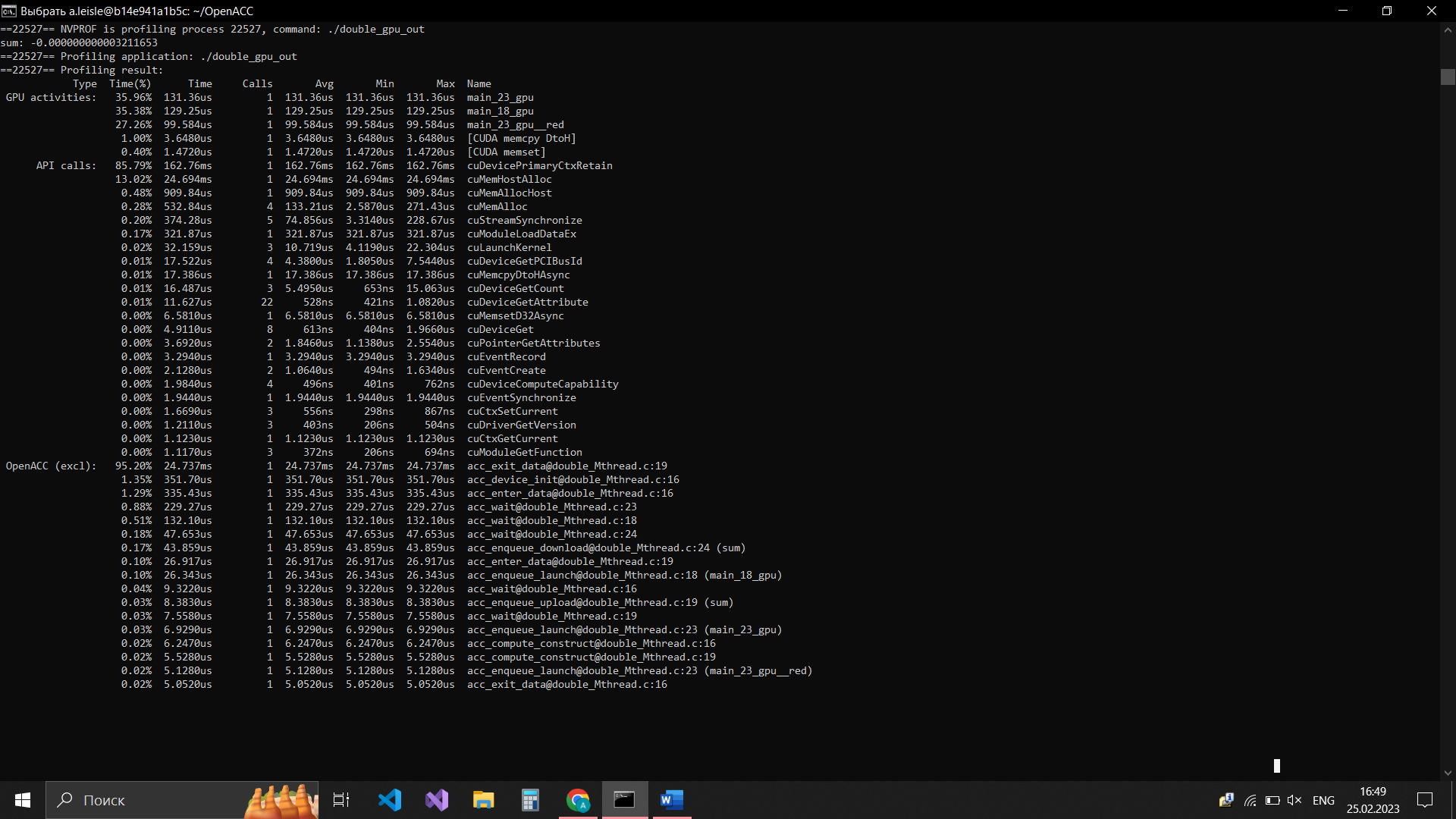
` 

Рис.3 – Профилирование программы “double\_Mthread.c”

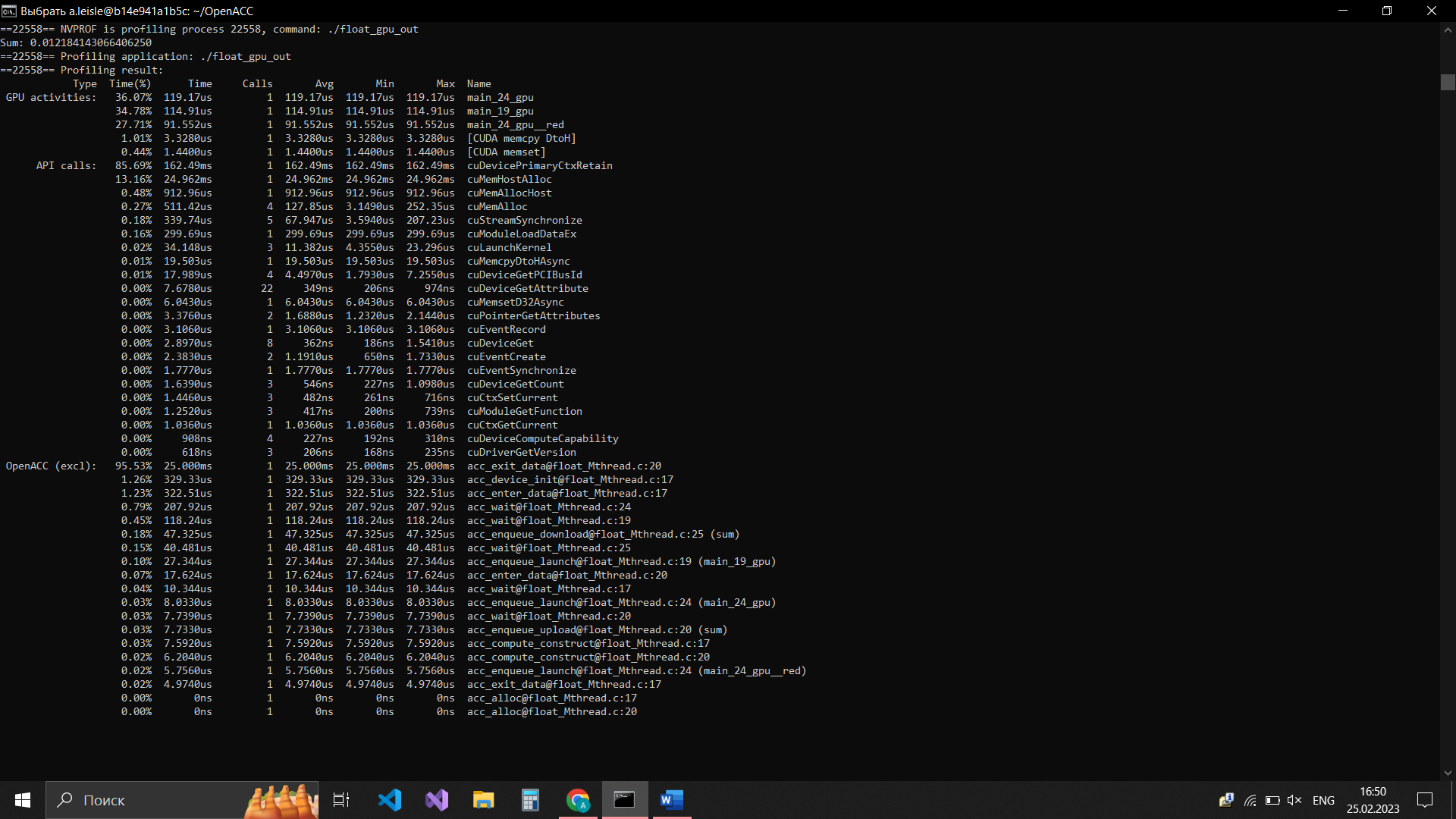


Рис.4 – Профилирование программы “float\_Mthread.c”

Как видно из профилирования программ, наибольшая точность расчёта достигается при использовании элементов массива типа double, т.к. сумма их близка к нулю (-0.000000000003211653).

При использовании типа float точность вычисления понижается и уже составляет 0.012184143066406250.

Помимо этого, в профилировании можно увидеть, что для выполнения кода используется 3 ядра: 2 ядра выделено под 2 цикла и 1 ядро выделено под редукцию, то есть для суммирования. Время выполнения циклов на этих ядрах распределяется равномерно и суммарное время их выполнения составляет 360.194мкс для “double\_Mthread.c” и 325.632мкс для “float\_Mthread.c”. Общее время выполнения программ соответственно составляет 0.2149с и 0.2149с.

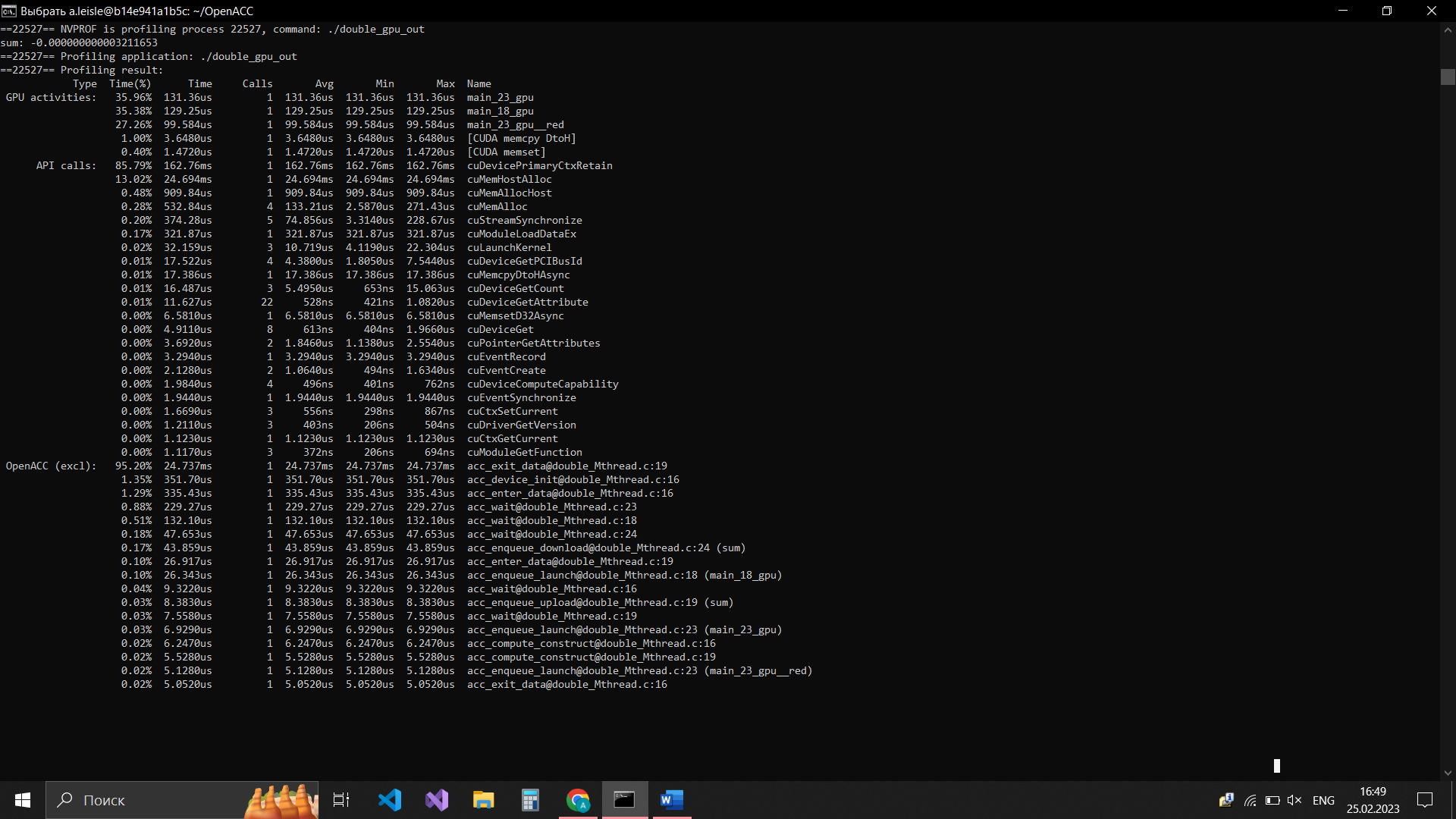


Рис.5 – Время выполнения на ядрах GPU для программы «double\_Mthread.c»

Диаграмма 1 – Время выполнения “double\_Mthread.c” на нескольких ядрах GPU

Диаграмма 2 – Время выполнения “float\_Mthread.c” на нескольких ядрах GPU

Сравним результаты выполнения данных программ с результатами, полученных при выполнении программ на CPU.

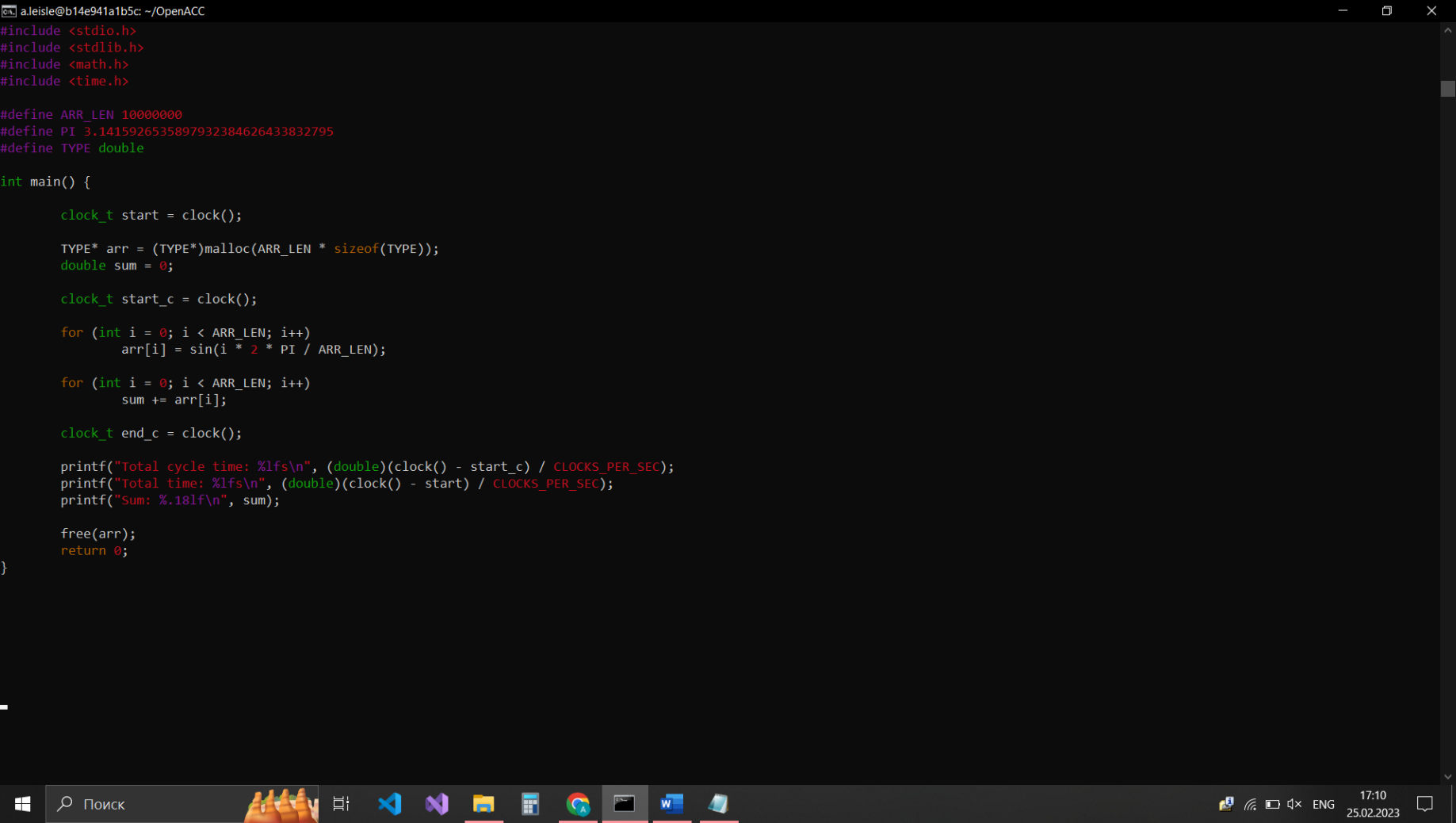


Рис.6 – Программа “double\_Sthread.c”

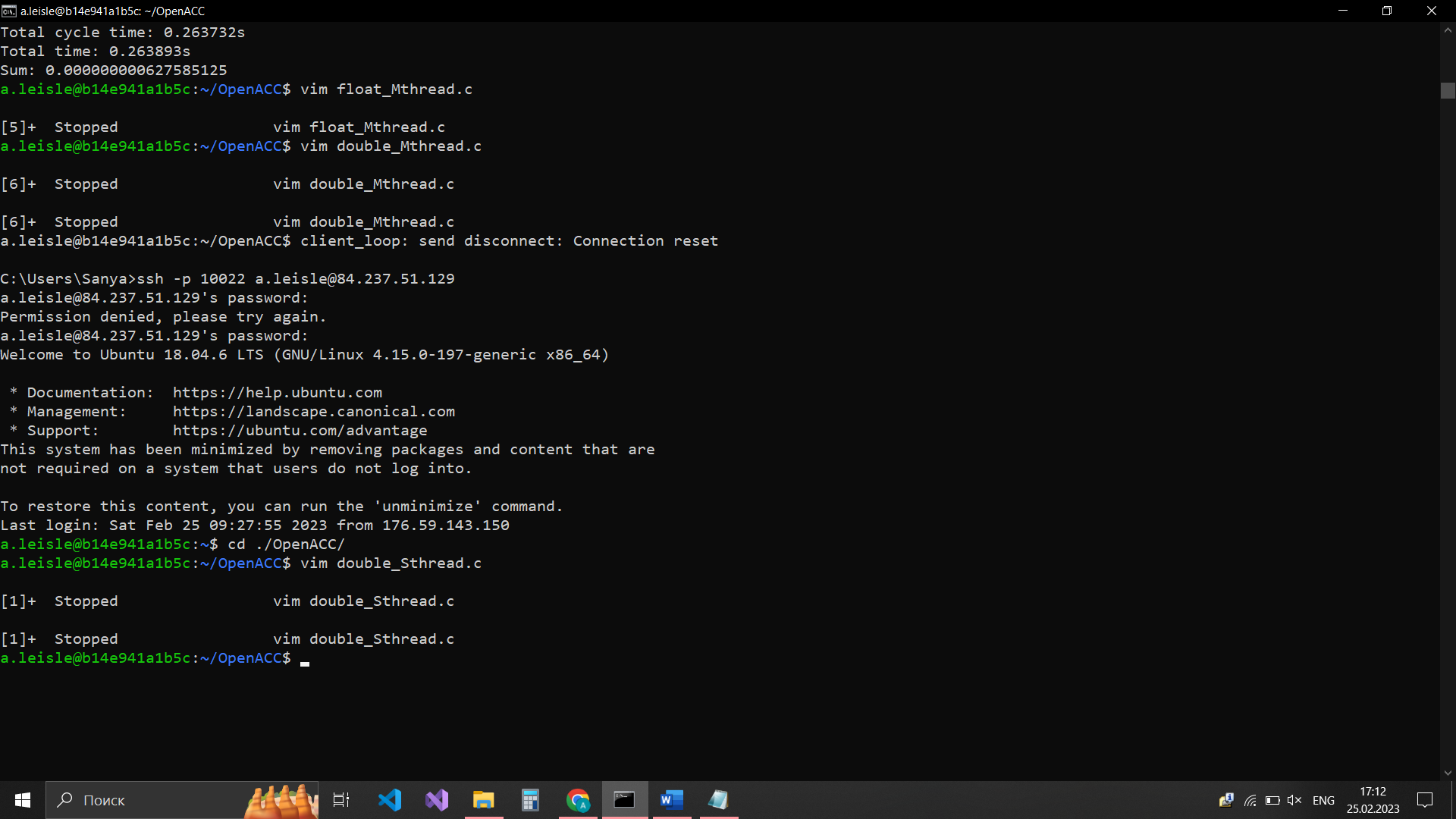


Рис.7 – Результат программы “double\_Sthread.c” на одном ядре CPU

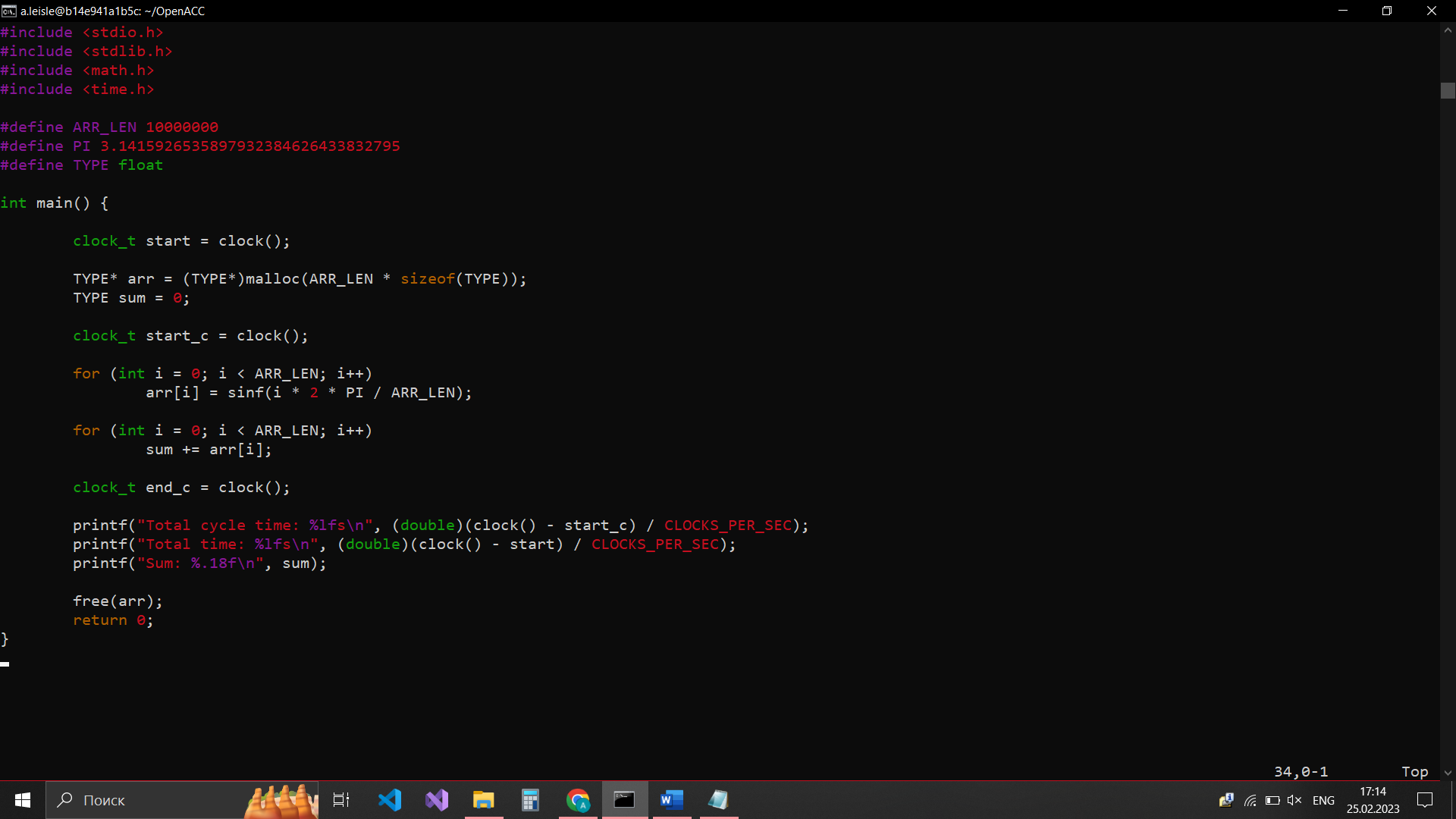


Рис.8 – Программа “float\_Sthread.c”

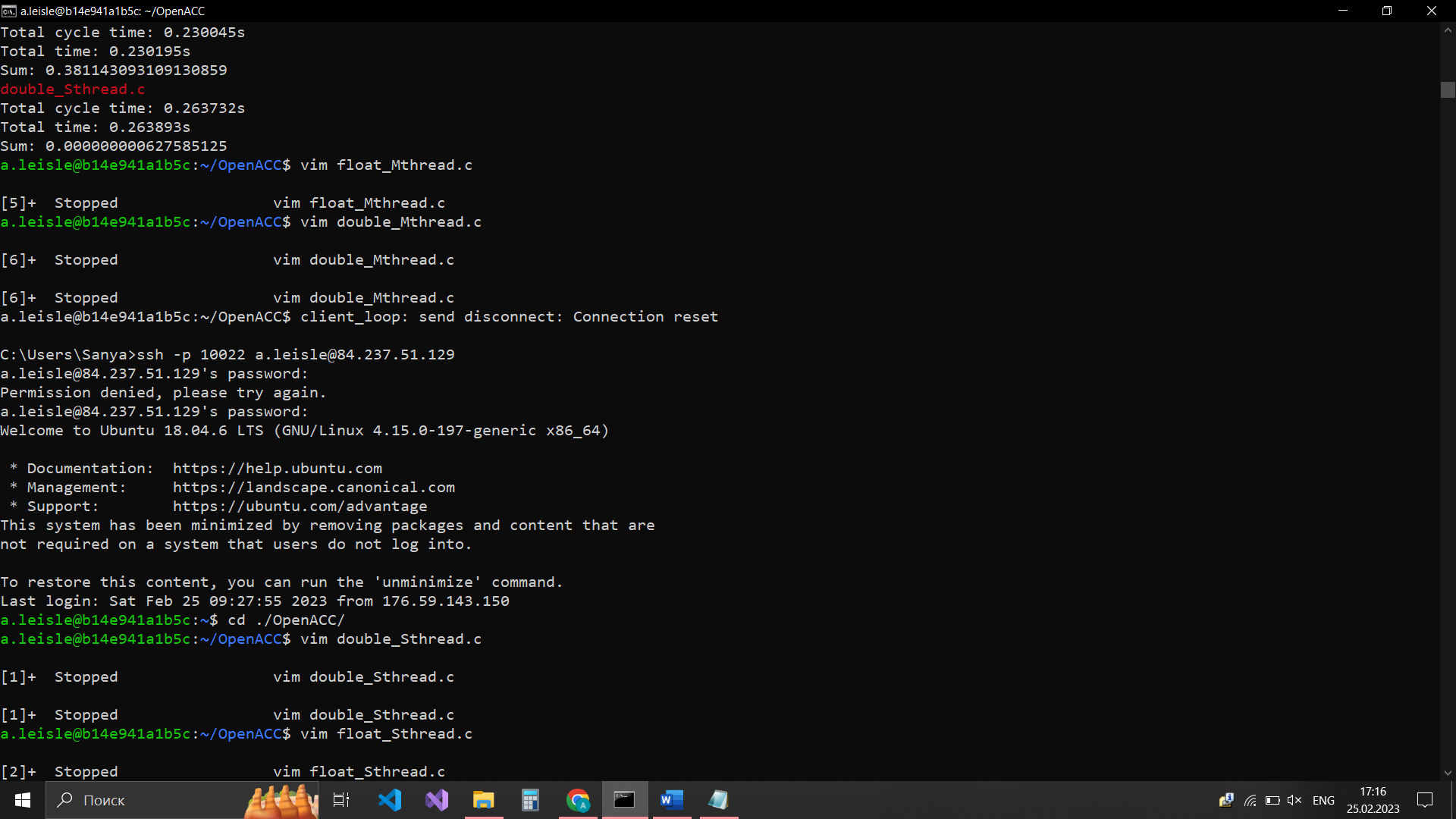


Рис.9 – Результат программы “float\_Sthread.c” на одном ядре CPU

Общее время выполнения циклов при выполнении на одном ядре для “float\_Sthread.c” и “double\_Sthread.c” составляет соответственно 0.230045c и 0.263732c, а общее время работы программы составляет 0.230195с и 0.263893c. Значения сумм у обеих программ равны 0.381143093109130859 и 0.000000000627585125.

Диаграмма 3 - Время выполнения программы “float\_Sthread.c” на одном ядре CPU

Диаграмма 4 - Время выполнения программы “double\_Sthread.c” на одном ядре CPU

При выполнении данных программ на нескольких ядрах CPU, мы получаем общее время выполнения для «float\_Mthread.c» - 0,018c, общее время выполнения циклов – 0,017c, а сумма получается равной

-0.328125000000000000.

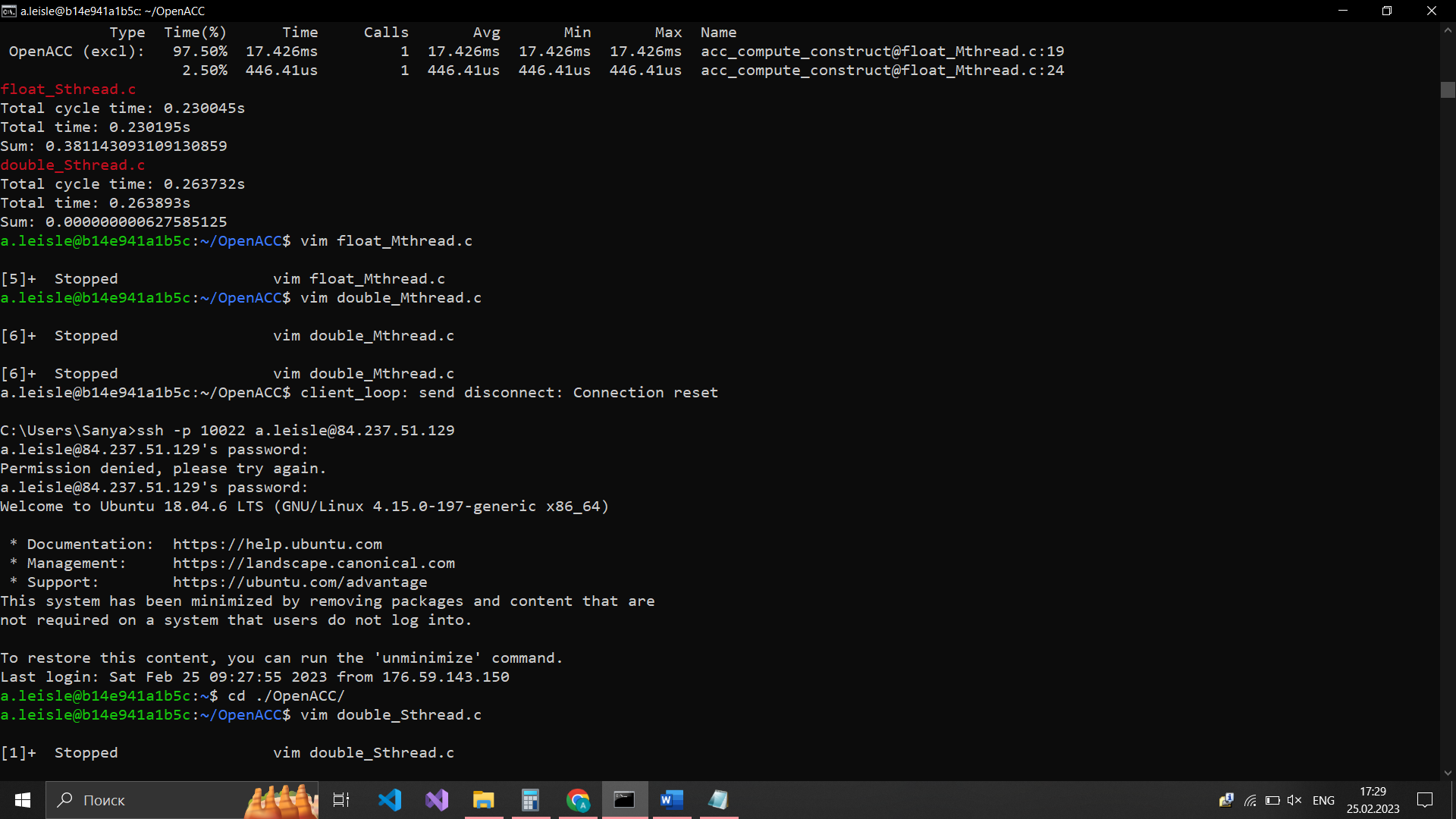


Рис.10 – Профилирование “float\_Mthread.c” на нескольких ядрах CPU

Диаграмма 5 - Время выполнения “float\_Mthread.c” на нескольких ядрах CPU

Для файла «double\_Mthread.c» на нескольких ядрах, мы получаем общее время выполнения – 0,023c, общее время выполнения циклов – 0,021c, а сумма получается равной -0.328125000000000000.

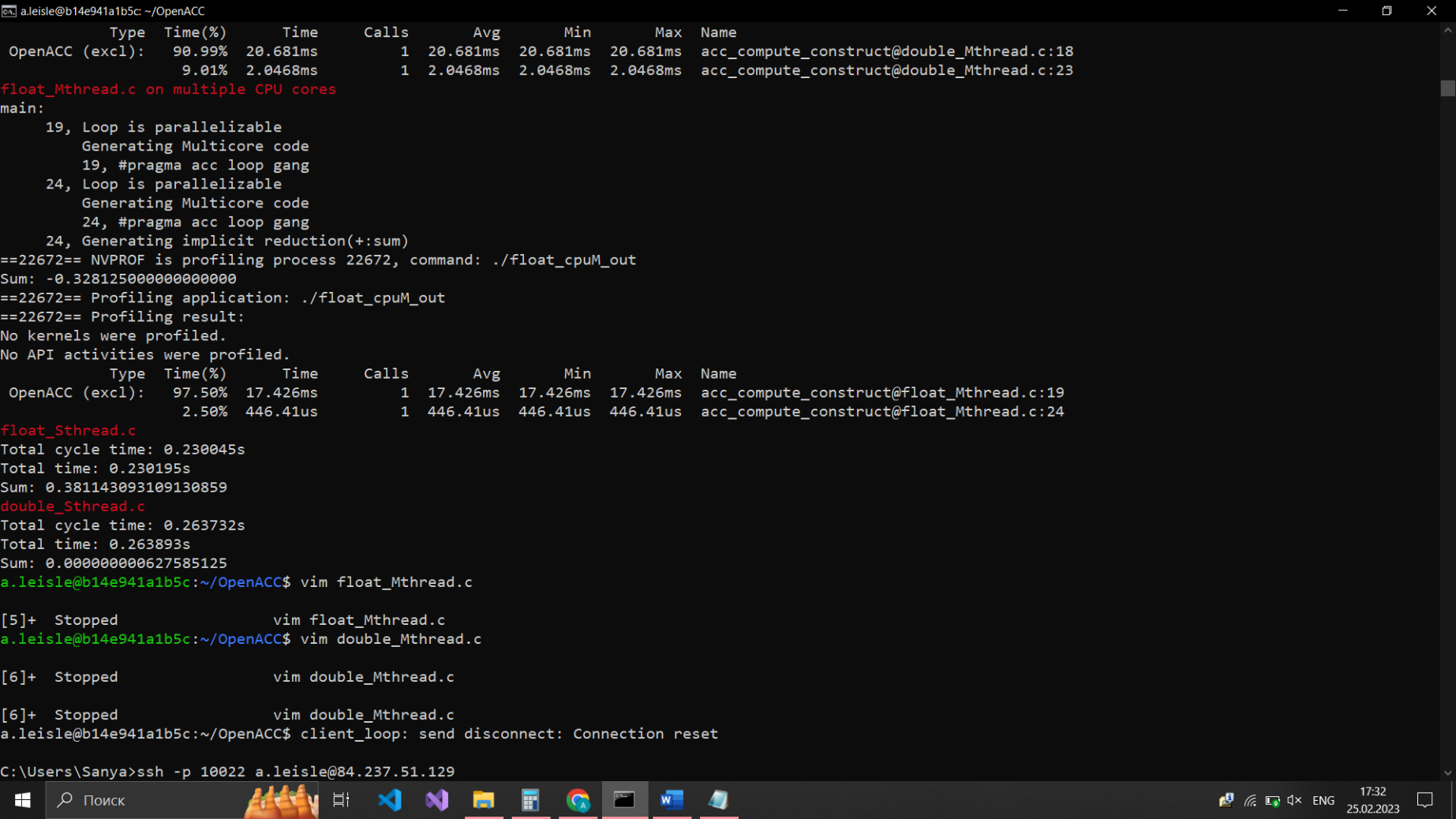


Рис.11 – Профилирование “double\_Mthread.c” на нескольких ядрах CPU

Диаграмма 6 - Время выполнения “double\_Mthread.c” на нескольких ядрах CPU

В результате, видно, что при исполнении кода на нескольких ядрах GPU основное время выполнения программы тратится на копирование данных и выделение памяти, при выполнении на CPU основное время программы тратится на выполнение циклов.

Исходя из полученных данных, для решения данной задачи разумнее выбрать исполнение кода на нескольких ядрах CPU, так как общее время выполнения оказалось меньше всего. Однако, если в задаче будет стоять вычисление большего количества данных, то разумнее будет использовать выполнение на нескольких ядрах GPU.

Ссылка на github: https://github.com/HerrPhoton/OpenACC.git