**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи №1

На тему: «Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Windows»

З дисципліни: «Операційні системи»

**Лектор** : ст.викл каф.ПЗ

Грицай О.Д.

**Виконала:** ст.гр.ПЗ-23

Кохман О.В.

**Прийняла:** ст.викл каф.ПЗ

Грицай О.Д.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2022 р.

\_ \_ \_ \_ \_ \_ \_ \_ .

Львів – 2022

**Тема:** Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Windows.

**Мета:** Ознайомитися з процесами та потоками в операційній системі Windows. Навчитися працювати із системними утилітами, що дають можливість отримувати інформацію про процеси, потоки, використовувану пам’ять та іншу необхіжнк інформацію.

**Індивідуальне завдання**

1. За допомогою утиліти «Диспетчер задач» та Process Explorer отримати повну інформацію про процеси: ідентифікатор процесу, завантаження ЦП (центрального процесора), час ЦП, базовий пріоритет, стан процесу, пам'ять-використання, пам'ять-зміни, пам'ять-максимум, помилок сторінки, об’єкти USER, код сеансу, об’єм віртуальної пам’яті, лічильник дескрипторів, лічильник потоків.

2. За допомогою утиліти Process Explorer отримати додаткову інформацію про процеси та їхні потоки.

3. Використовуючи «Диспетчер задач» та Process Explorer змінити пріоритет будь-якого процесу, від низького до «реального часу»; задати відповідність виконання процесів на окремих ядрах центрального процесора; виконати завершення процесу.

4. Використовуючи Process Explorer призупинити процес і відновити його роботу.

5. Скомпілювати файл main.cpp представлений нижче і запустити виконуваний файл на різній кількості активних процесорів (ядер). Знайти для даної програми величини A , S , p при різних вхідних значеннях величини n.

#include <iostream>

#include <thread>

#include <mutex>

#include <vector>

#include <algorithm>

#include <cmath>

#define DEF\_THREAD\_COUNT 4

const int N = 50000000;

namespace MCConcurrency

{

inline int threadsCount(int iterationsCount = -1)

{

int numThreads = static\_cast<int>(std::thread::hardware\_concurrency());

if (numThreads == 0)

numThreads = DEF\_THREAD\_COUNT;

if (iterationsCount == 0)

return 1;

if (iterationsCount > -1)

numThreads = std::min(static\_cast<int>(numThreads), iterationsCount);

return numThreads;

}

template<typename Function>

void parallel\_for(int first, int last, int threadsNumbers, const Function& l)

{

int numThreads = threadsNumbers;

int elementsCount = last - first;

//auto numThreads = threadsCount(last - first);

if (numThreads <= 0)

return;

int threadNumber = 0;

std::mutex tasksMutex;

std::vector<std::thread> threads;

const auto chunk = std::max(static\_cast<int>(std::ceil(elementsCount /

static\_cast<double>(numThreads))), 1);

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

{

threads.push\_back(std::thread([&]()

{

int thread\_number;

tasksMutex.lock();

thread\_number = threadNumber;

++threadNumber;

tasksMutex.unlock();

int beg = first + thread\_number \* chunk;

int end = std::min(first + (thread\_number + 1) \* chunk, elementsCount);

for (int ind = beg; ind < end; ind++)

{

l(ind, thread\_number);

}

}));

}

std::for\_each(begin(threads), end(threads), [](std::thread& th) {

th.join();

});

}

}

int main()

{

typedef std::chrono::high\_resolution\_clock Time;

typedef std::chrono::milliseconds ms;

typedef std::chrono::duration<float> fsec;

std::vector<float> arr(N);

auto numThreads = MCConcurrency::threadsCount(N);

std::cout << "Set process af inity (cores count) and press <Enter>";

getchar();

auto t0 = Time::now();

MCConcurrency::parallel\_for(0, N, numThreads, [&](int nIndex, int threadNumber) {

arr[nIndex] = sin(threadNumber) \* cos(threadNumber);

});

auto t1 = Time::now();

fsec fs = t1 - t0;

ms d = std::chrono::duration\_cast<ms>(fs);

std::cout << "Duration: " << d.count() << "ms\n";

return 0;

}

6. Дослідити вплив зміни відповідності ядру на швидкодію процесу. Виконати завдання згідно варіанту, що відповідає порядковому номеру у списку підгрупи

9) Копіювання файлів.

**Теоретичні відомості**

Операційна система - це сукупність програм, які призначені для керування ресурсами комп'ютера й обчислювальними процесами, а також для організації взаємодії користувача з апаратурою.

Ресурси - це логічні й фізичні компоненти комп'ютера: оперативна пам'ять, місце на диску, периферійні пристрої, процесорний час тощо.

Обчислювальним процесом (або завданням) називається послідовність дій, яка задається програмою.

Утиліта - це невелика програма, що виконує конкретну сервісну функцію.

Процес — об'єкт операційної системи, контейнер системних ресурсів, призначених для підтримки виконання програми.

Програма – це статичний набір команд, а процес – контейнер для ресурсів, які використовуються при виконанні екземпляра програми.

У Windows кожен процес має один із таких базових класів пріоритету:

● IDLE\_PRIORITY\_CLASS,

● BELOW\_NORMAL\_PRIORITY\_CLASS,

● NORMAL\_PRIORITY\_CLASS,

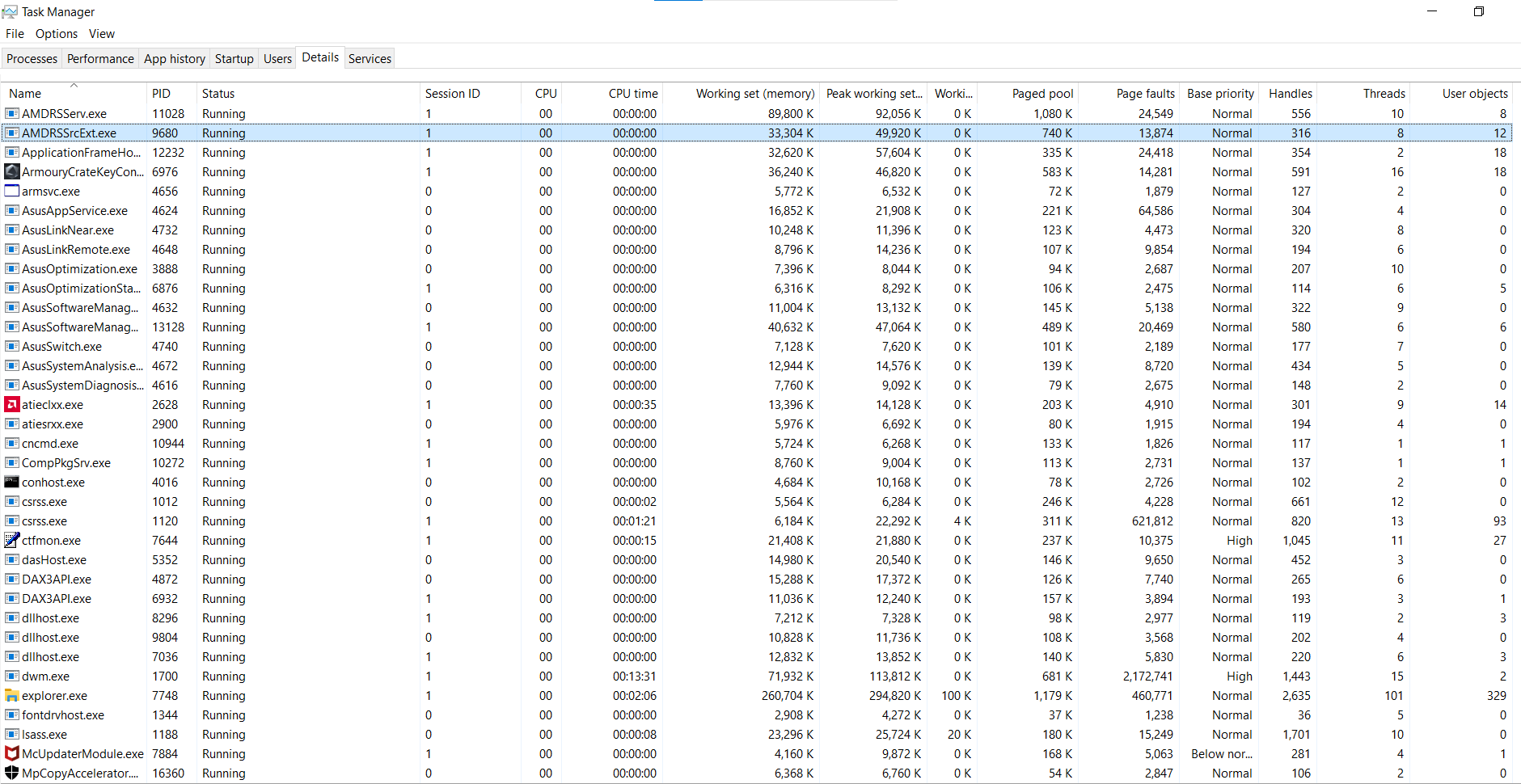
● ABOVE\_NORMAL\_PRIORITY\_CLASS,

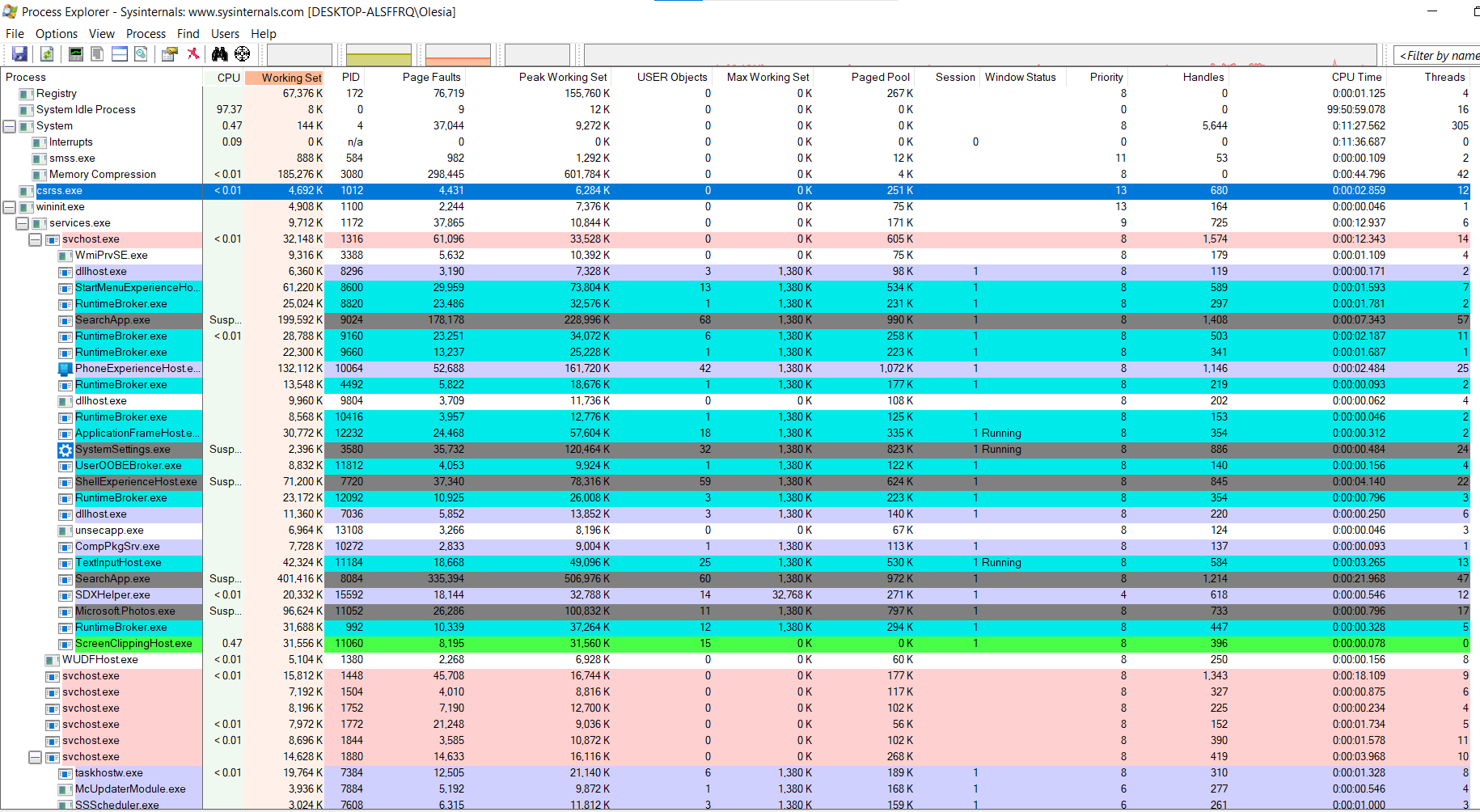
● HIGH\_PRIORITY\_CLASS,

● REALTIME\_PRIORITY\_CLASS.

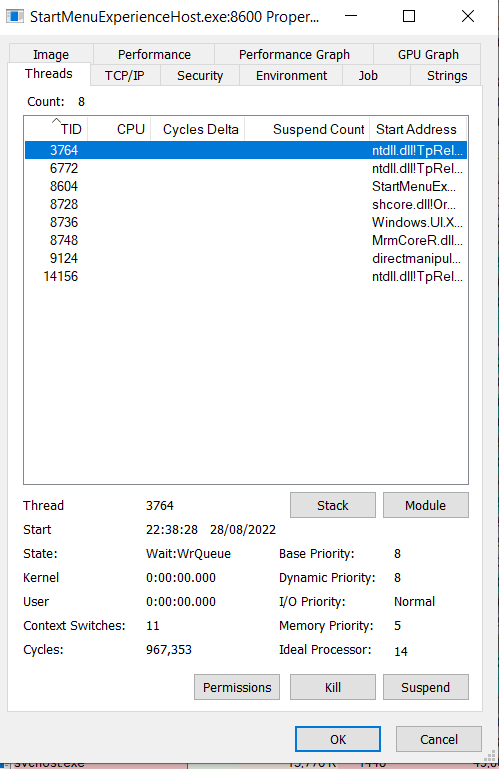
**Протокол роботи**

**1:**

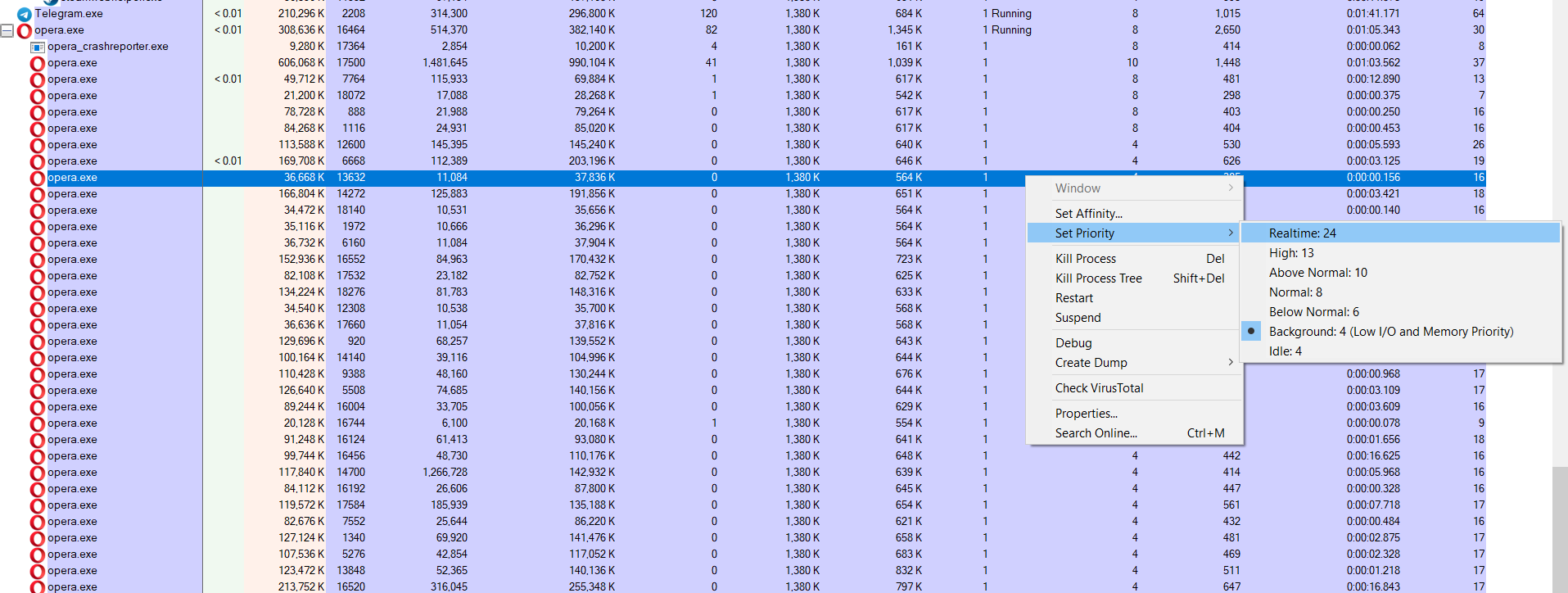
****

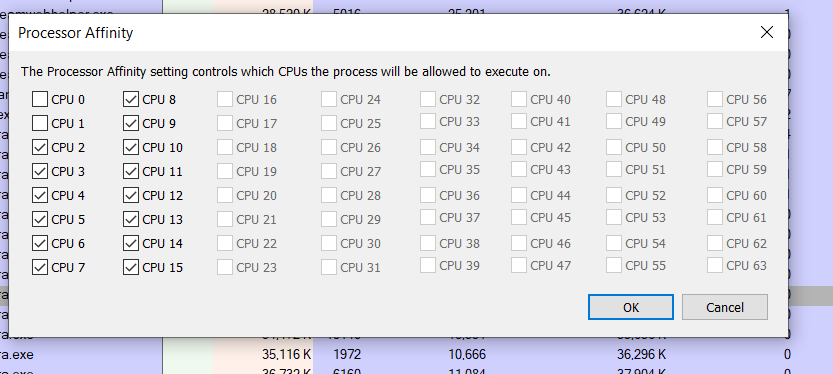
****

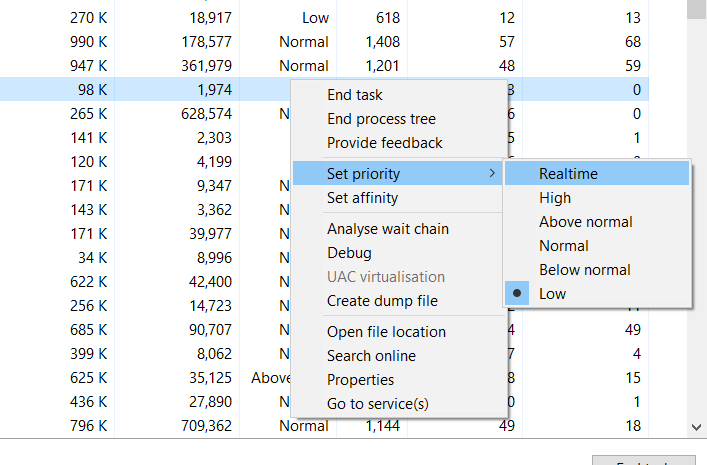
**2:**

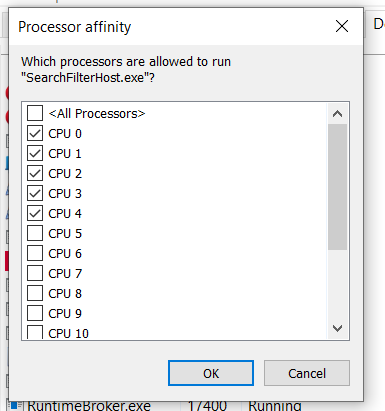
****

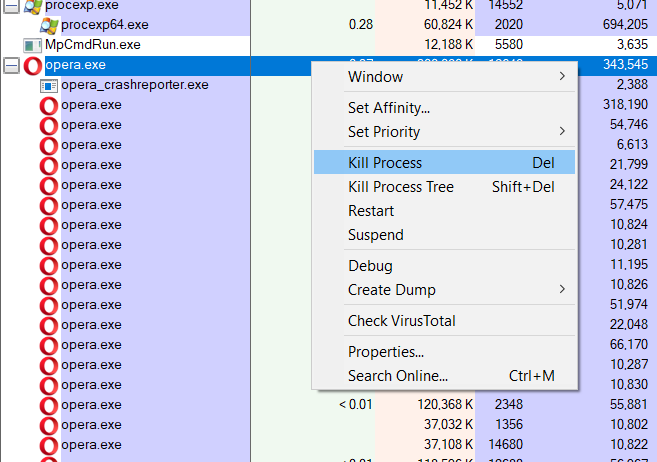
**3:**

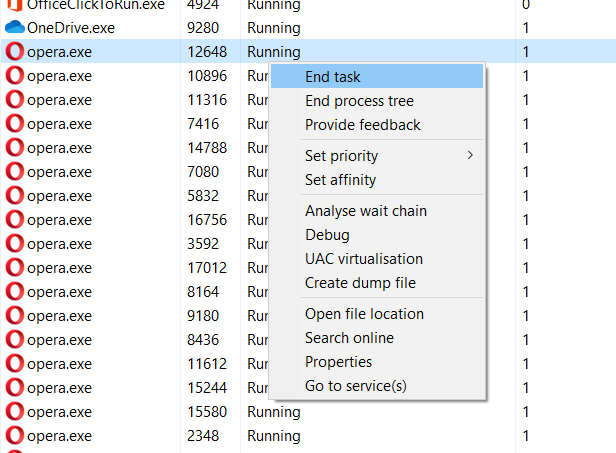
****

****

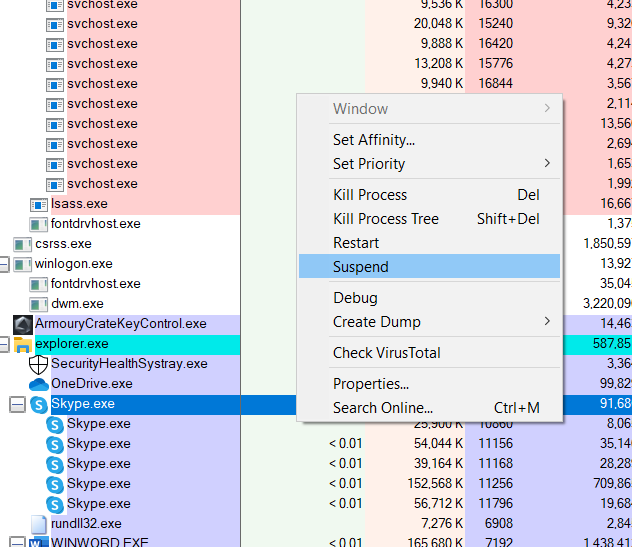
****

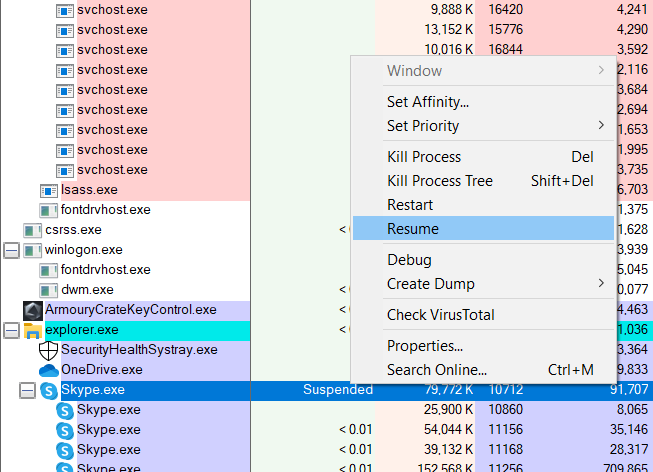
****

****

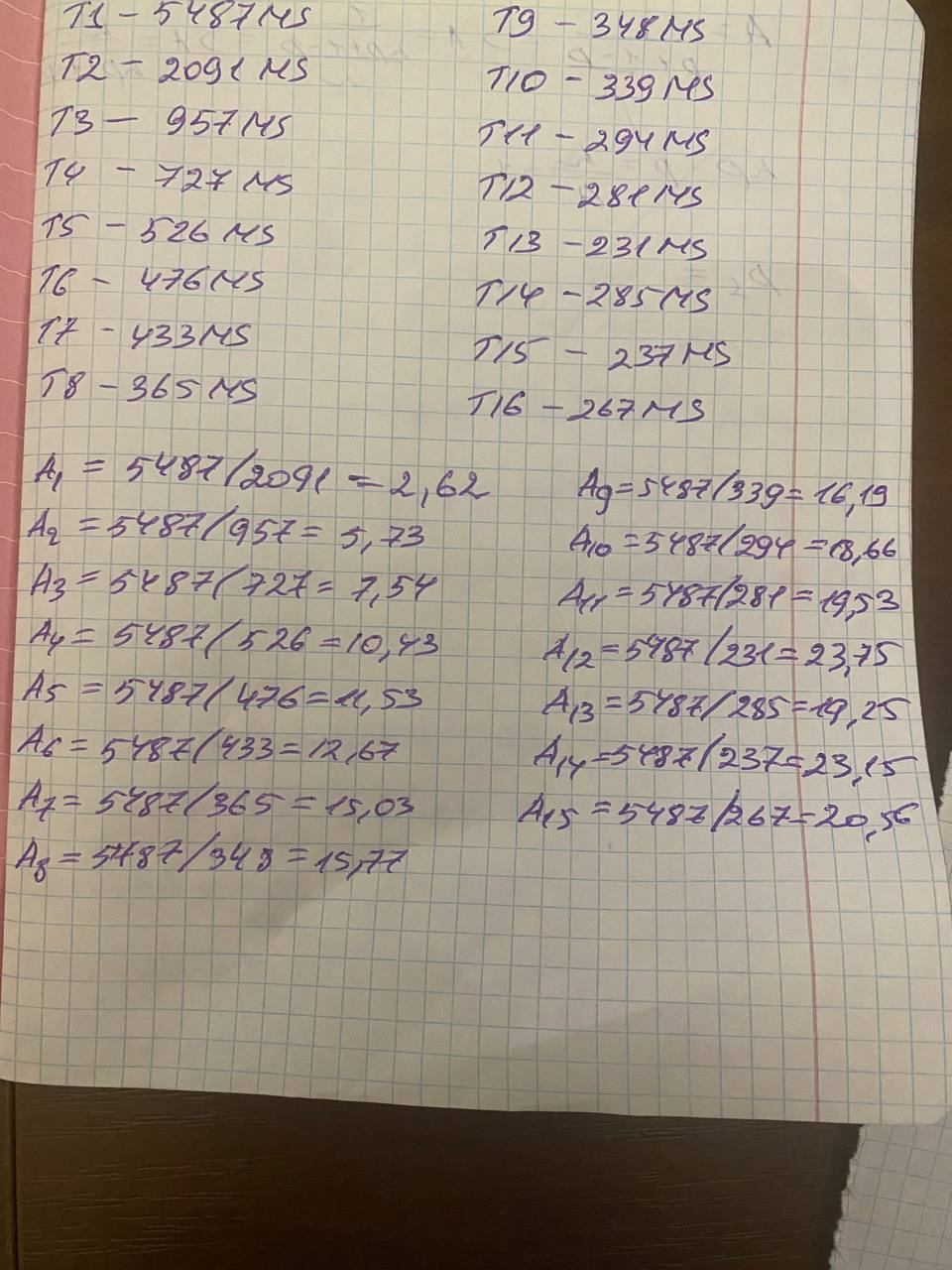
****

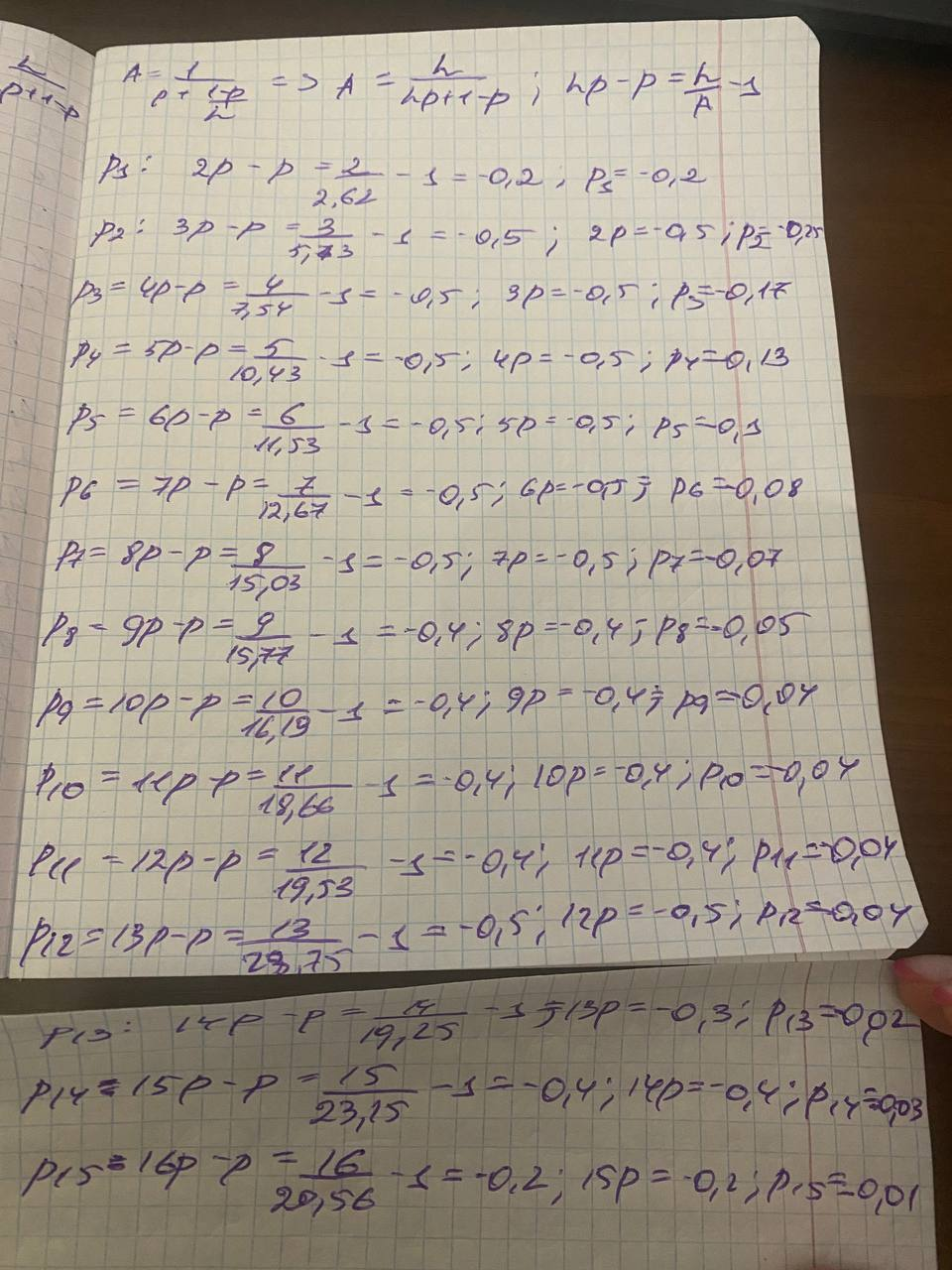
**4:**

****

****

**5:**





**6:**

CPU1 – 8260 ms, CPU2 – 3850 ms, CPU3 – 4070 ms, CPU4 – 3910 ms, CPU5 – 3520 ms, CPU6 – 3610 ms, CPU7 – 3710 ms, CPU8 – 3530 ms, CPU9 – 3190 ms, CPU10 – 3190 ms, CPU11 – 3240 ms, CPU12 – 4200 ms, CPU13 – 3410 ms, CPU14 – 3650 ms, CPU15 – 350 ms, CPU16 – 3610 ms.

**Висновок**

На цій лабораторній роботі я ознайомилась з Task Manager і Process Explorer, дослідила швидкість роботи процесів при різній кількості ядер.