**Основи об'єктно-орієнтованого програмування**

**Лабораторна робота №3а**

**Паралельне мультипоточне програмування**

Хід виконання:

Спочатку створюється клас за допомогою циклу for C# і побачимо, скільки часу знадобиться для завершення виконання. Потім ми напишемо інший клас за допомогою методу Parallel For і побачимо, скільки часу знадобиться для завершення виконання.

В кожному класі створюється однаковий послідовний цикл. Цикл повторюється десять разів, а керуюча змінна циклу збільшується з нуля до дев’яти. На кожній ітерації викликається метод DoSomeIndependentTask. Метод DoSomeIndependentTask виконує обчислення, яке включається для створення достатньо тривалої паузи, щоб побачити покращення продуктивності паралельної версії.

Створюємо інтерфейс Global з оголошеним абстрактним методом Method

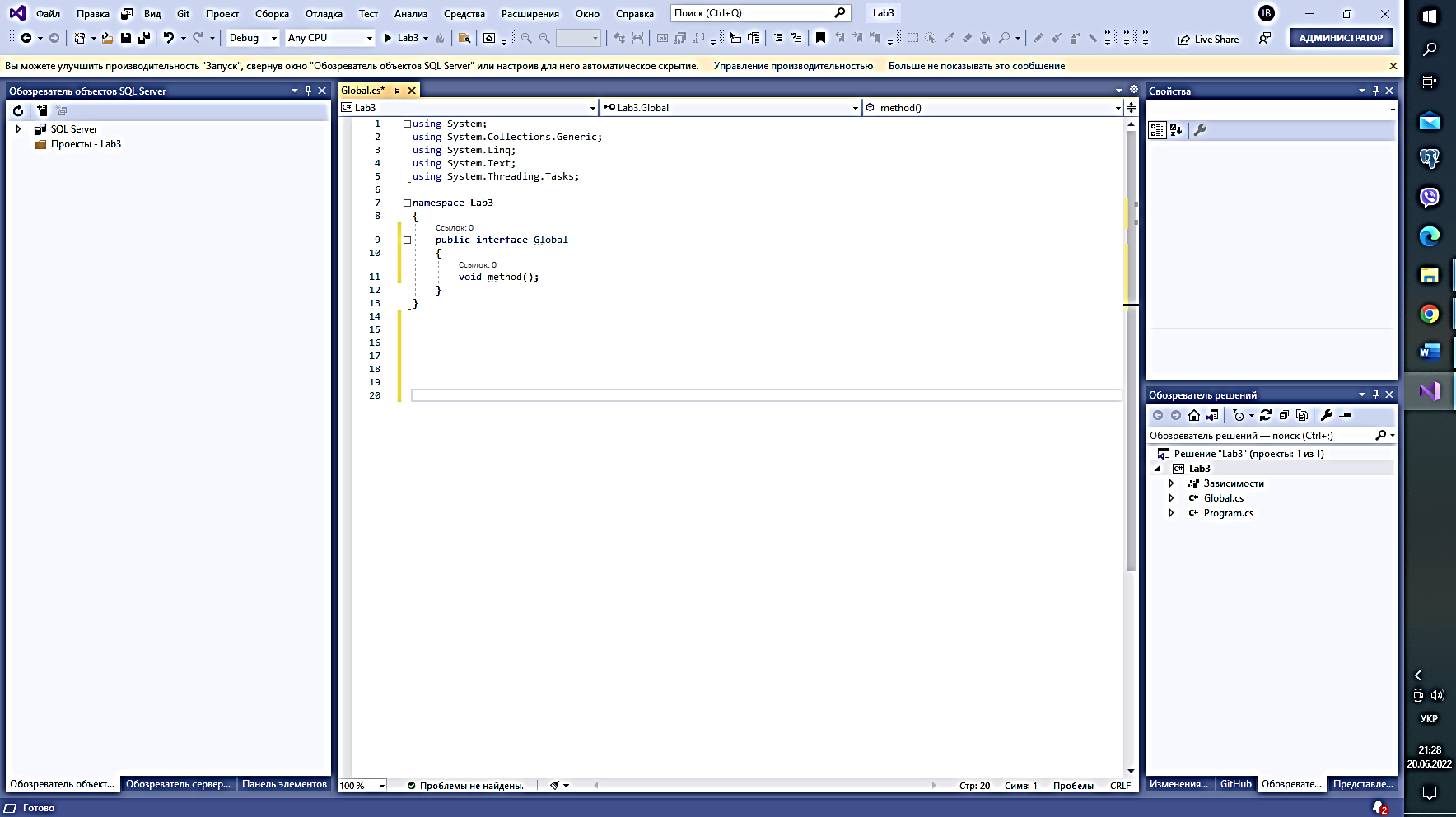


Рис. 1. Скріншот інтерфейсу Global

Створюємо клас Consistent з перезавантаженим методом Method, який виконує розрахунки в послідовному режимі. Клас є нащадком інтерфейсу Global.

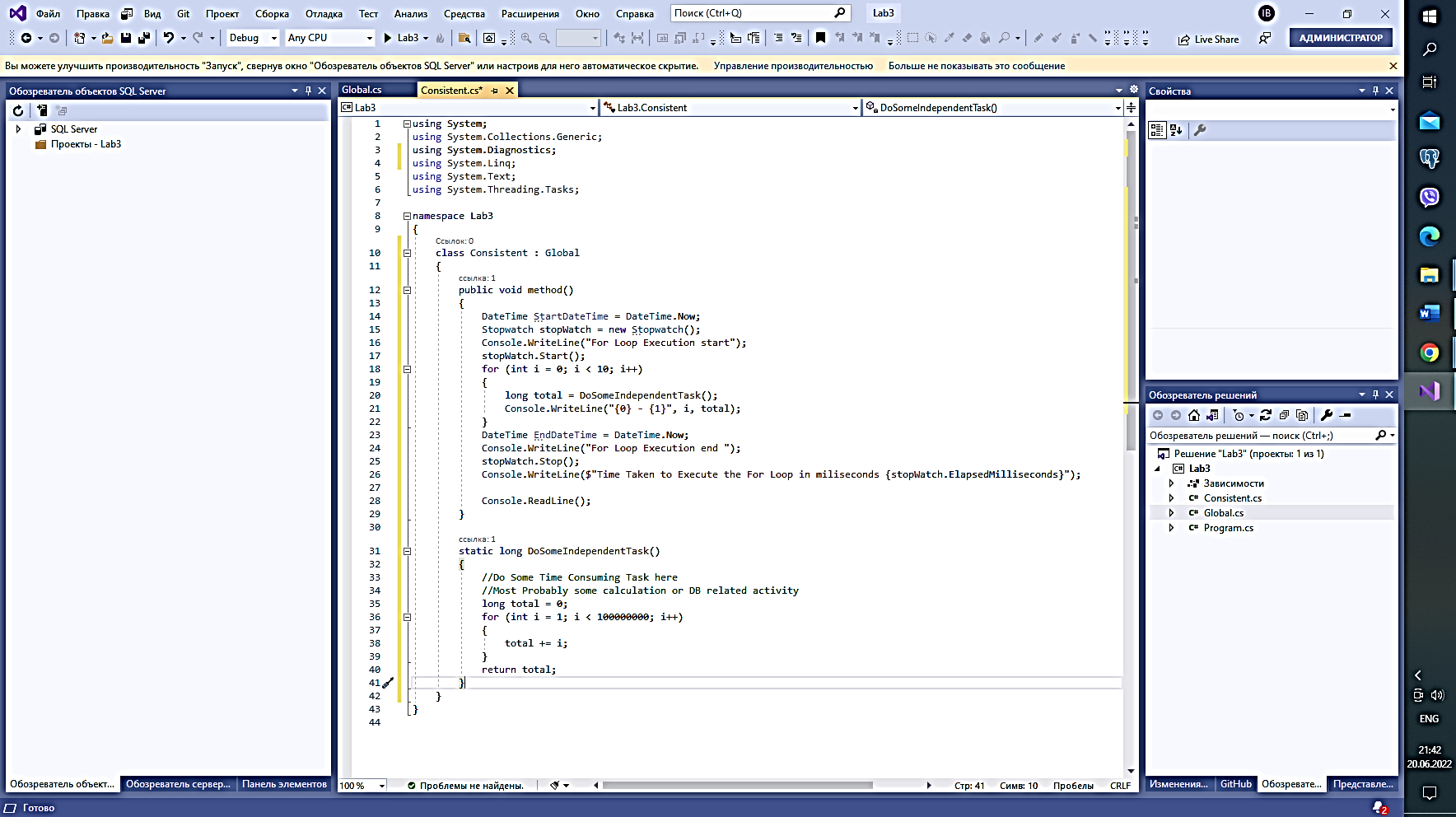


Рис. 2. Скріншот класу Consistent

Створюємо клас Рarallel з перезавантаженим методом Method, який виконує розрахунки в паралельному режимі. Клас є нащадком інтерфейсу Global.

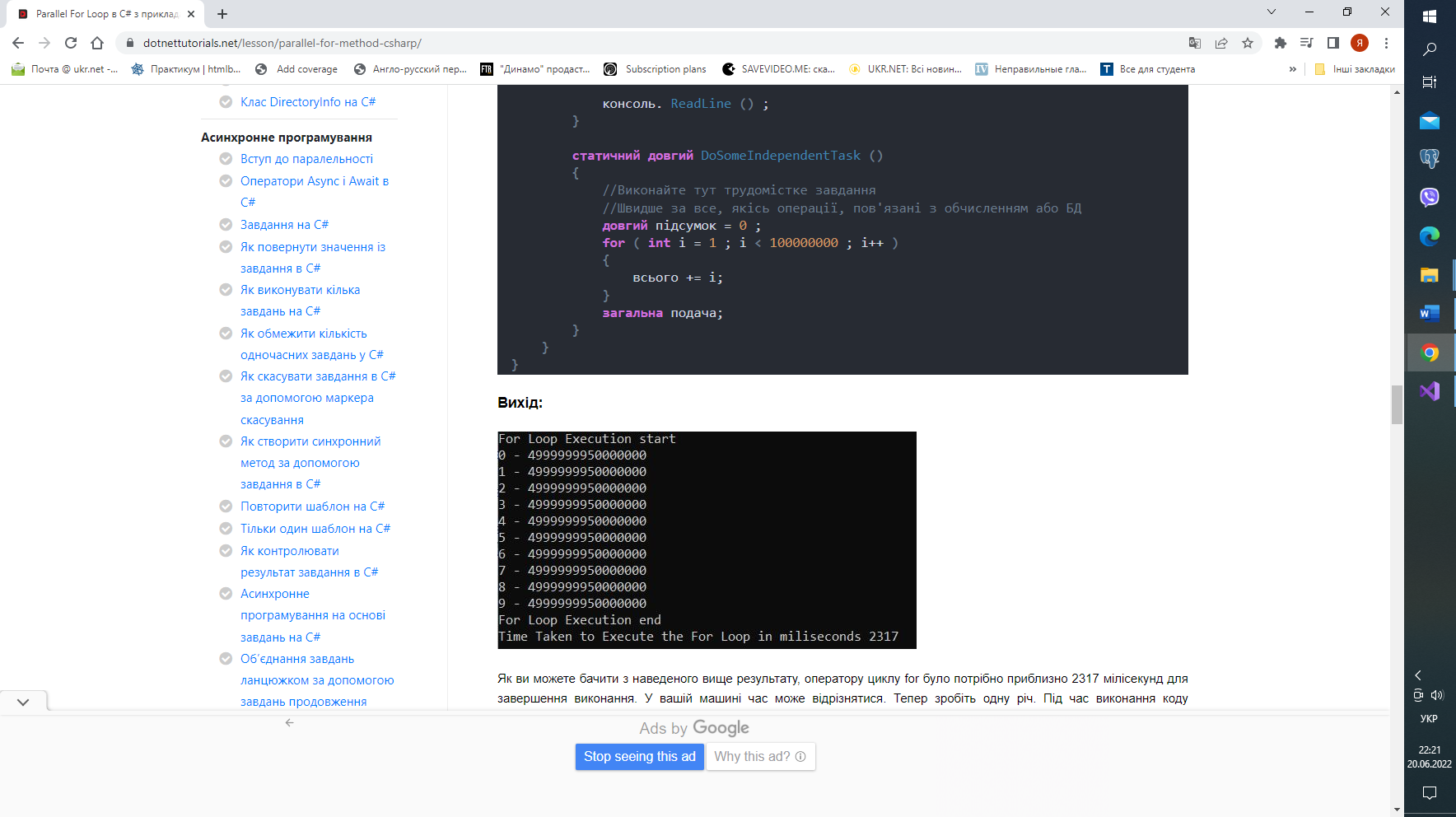


Рис. 3. Скріншот реалізації класу Consistent в послідовному режимі.

Як можна бачити з наведеного вище результату, оператору циклу for було потрібно приблизно 2317 мілісекунд для завершення виконання. Під час виконання коду відкрийте диспетчер завдань і подивіться на використання ЦП. На машині максимум 42%, використання ЦП виконується, як показано на зображенні нижче.

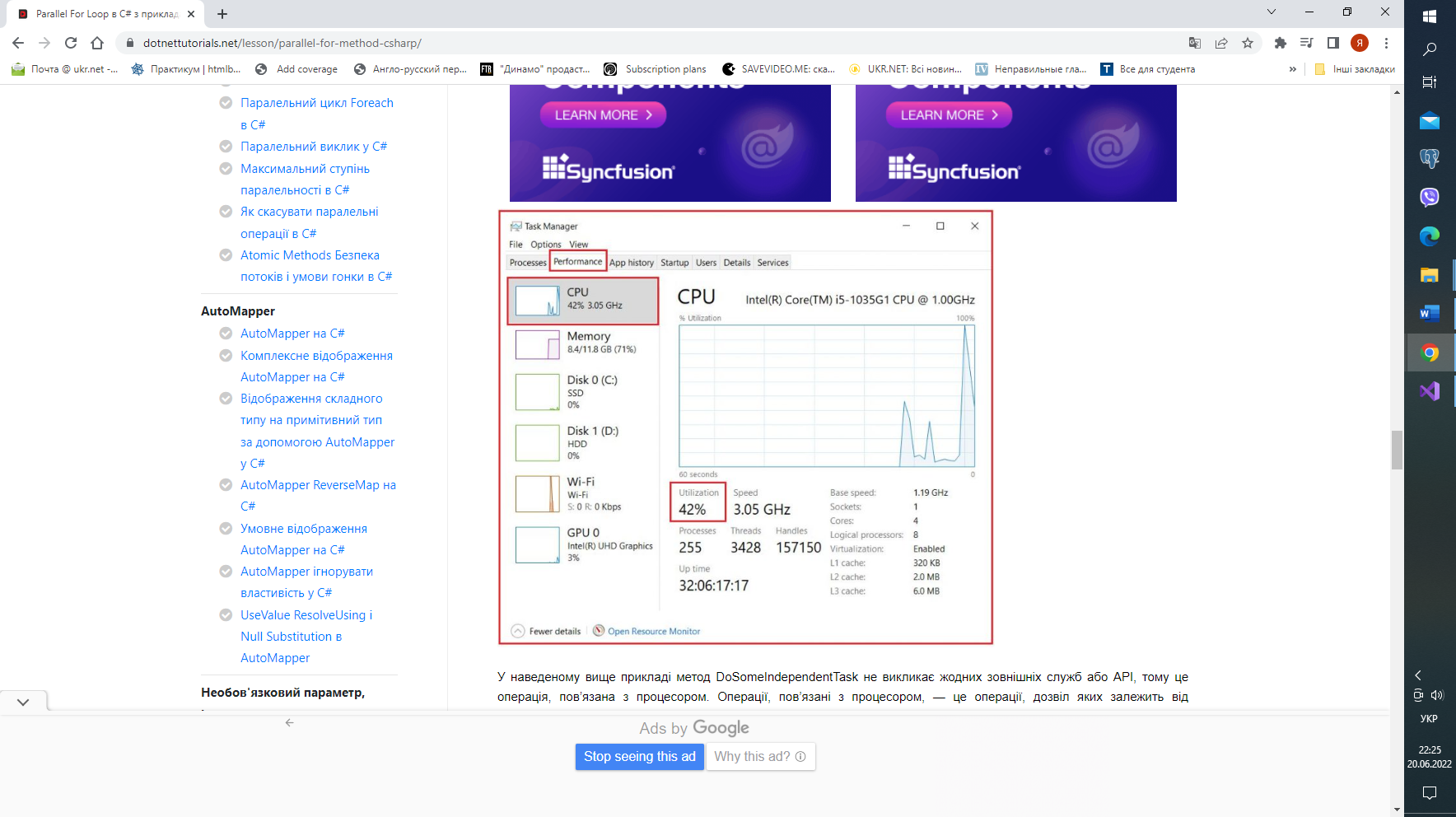


Рис. 4. Скріншот роботи ЦП під час реалізації в послідовному режимі

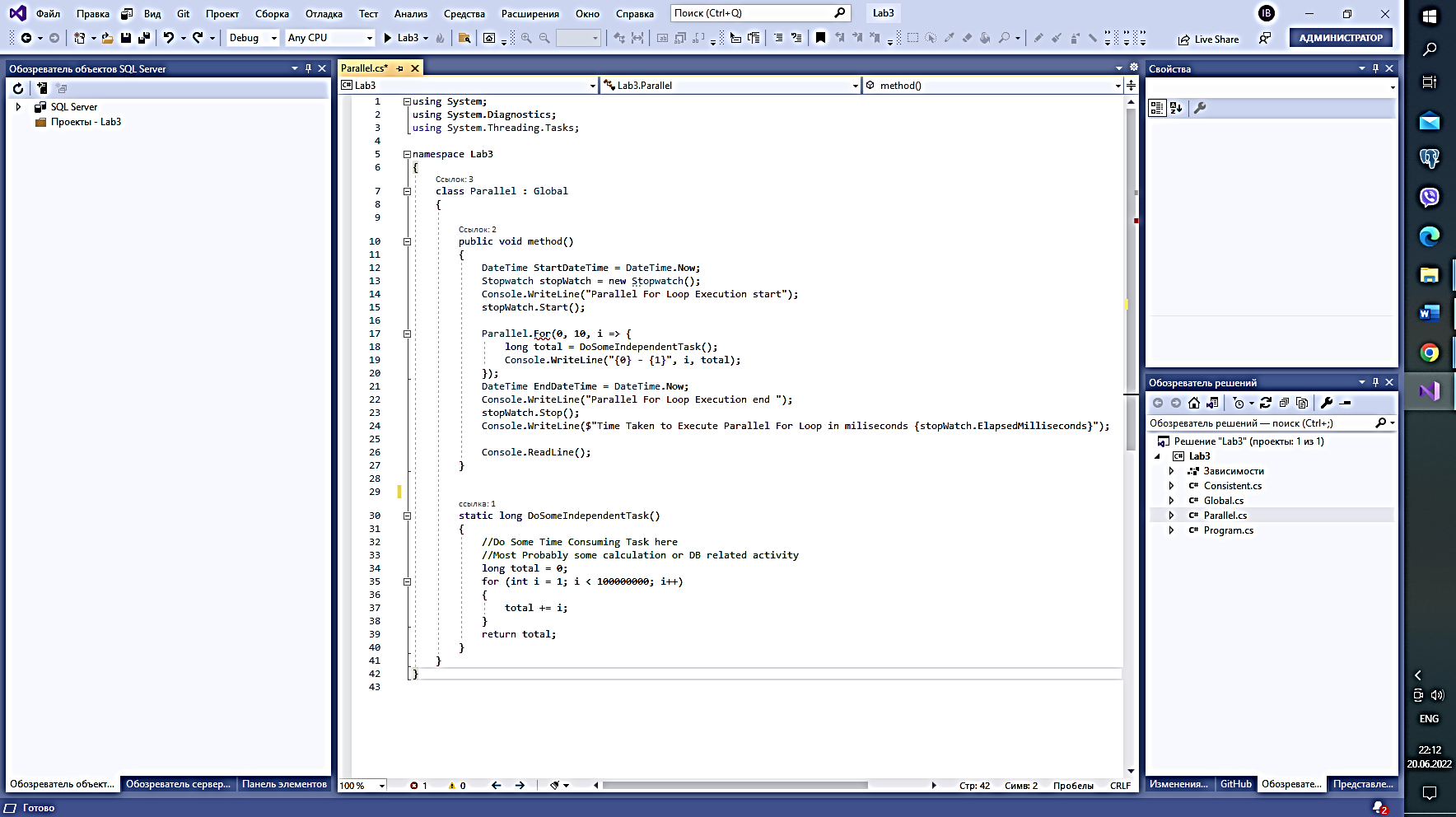


Рис. 5. Скріншот класу Рarallel

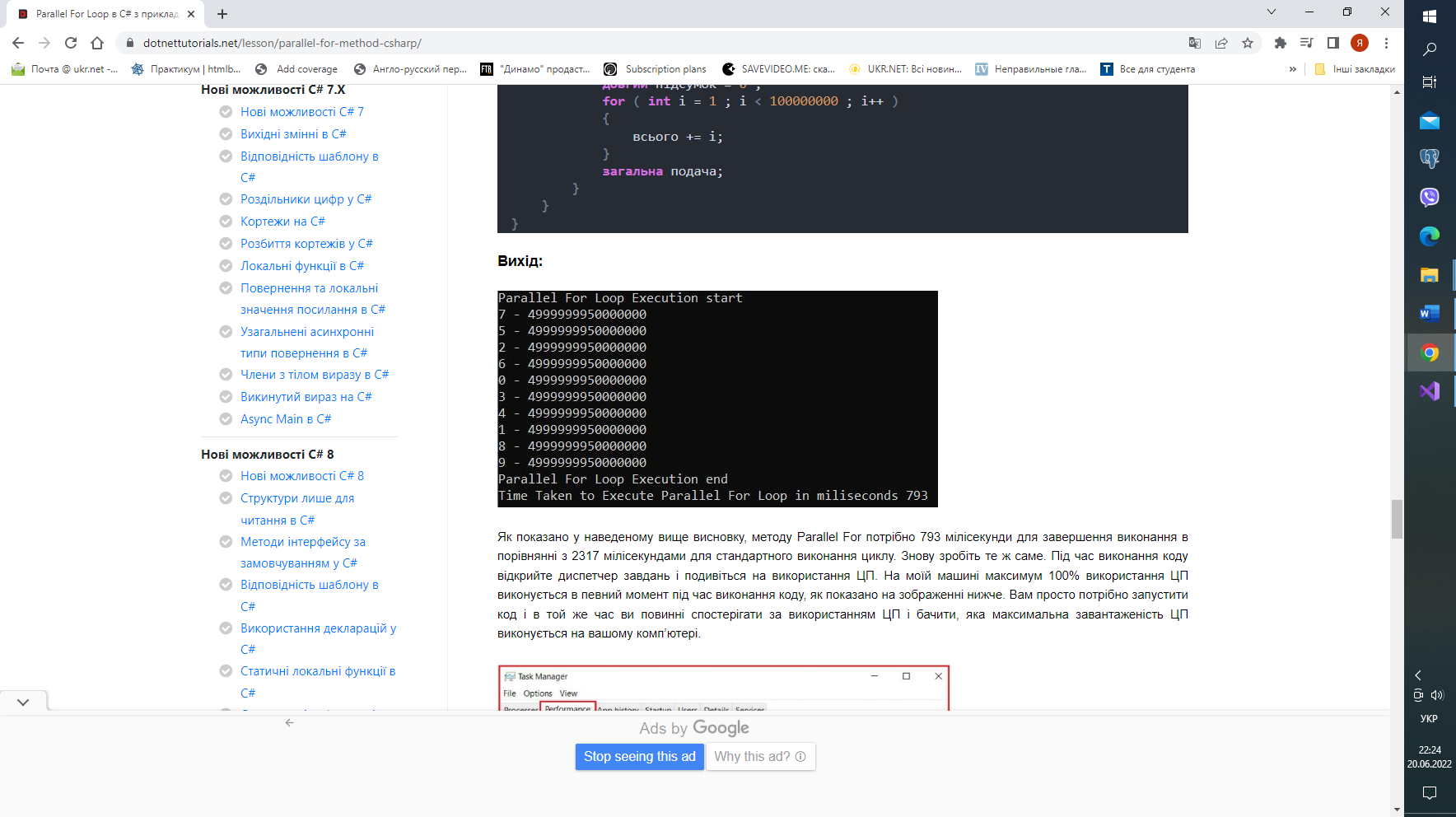


Рис. 5. Скріншот реалізації класу Рarallel в паралельному режимі.

Як показано у наведеному вище висновку, методу Parallel For потрібно 793 мілісекунди для завершення виконання в порівнянні з 2317 мілісекундами для стандартного виконання циклу. На машині максимум 100% використання ЦП виконується в певний момент під час виконання коду, як показано на зображенні нижче.

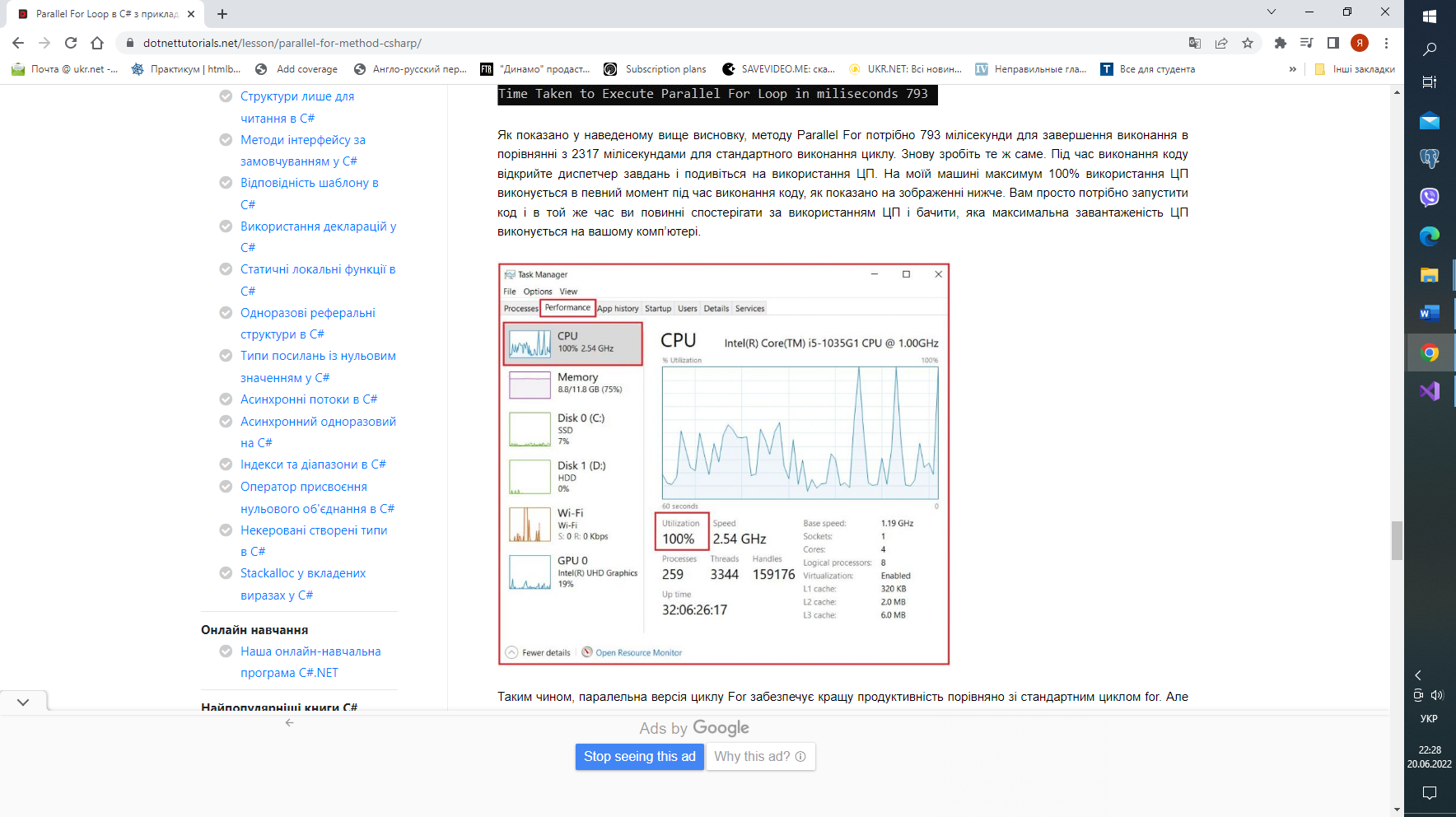


Рис. 8. Скріншот роботи ЦП під час реалізації в паралельному режимі

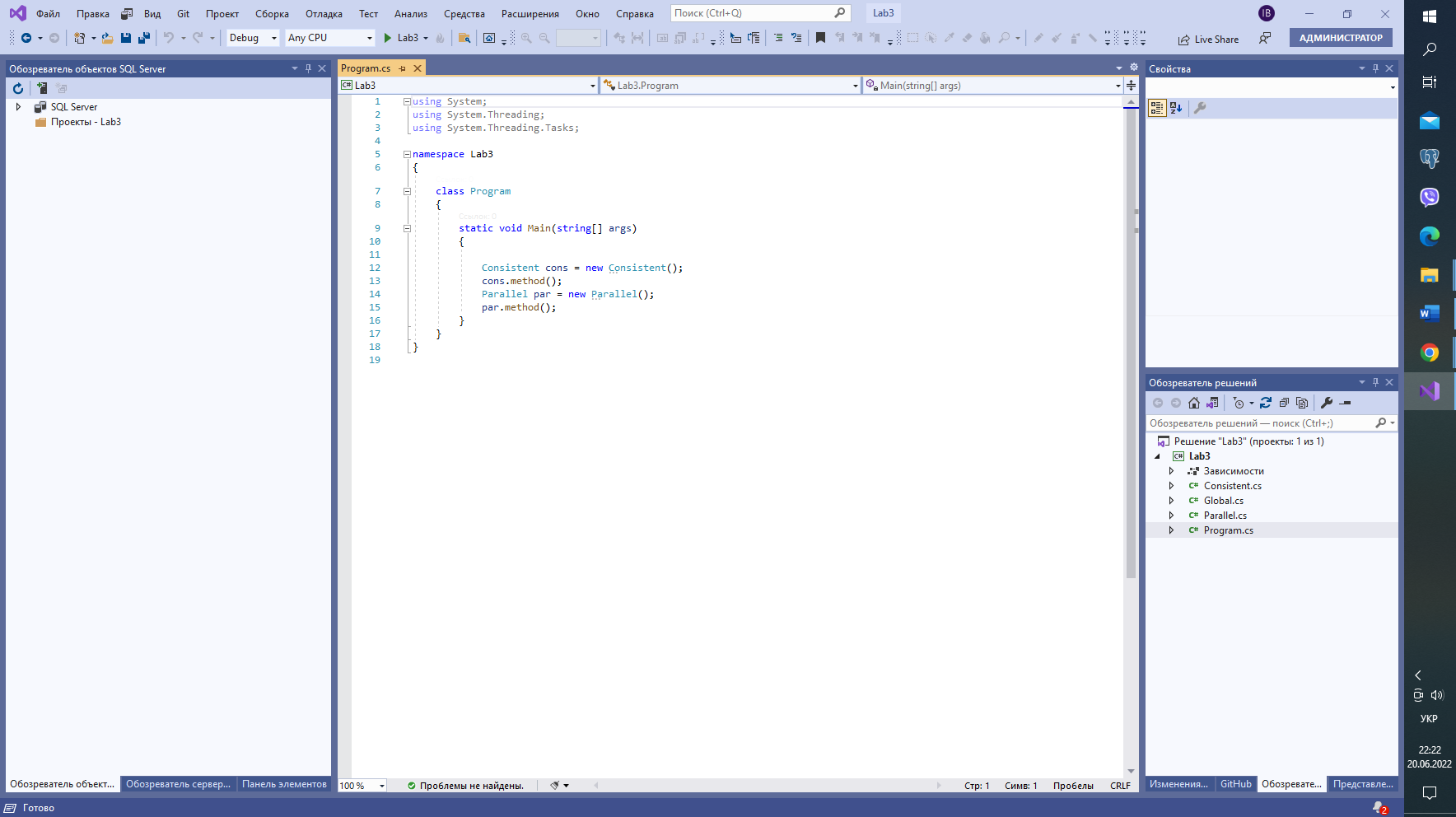


Рис. 6. Скріншот класу Program.

ВИСНОВКИ

Таким чином, паралельна версія реалізація алгоритму перечислення чисел від 1 до 10 забезпечує кращу продуктивність порівняно зі стандартним послідовним циклом приблизно в 3 рази вище, в тому числі за рахунок вищої завантаженості ЦП на 62%.