**Лабораторная №1**

***Практические приемы построения многопоточных приложений***

Цели и задачи: Изучить работу с потоками. Научиться разбивать задачу на части, для последующего их выполнения различными потоками. Решение задачи с помощью WinAPI.

Потоки предоставляют возможность проведения параллельных или псевдопараллельных, в случае одного процессора, вычислений. Потоки могут порождаться во время работы программы, процесса или другого потока. Основное отличие потоков от процессов заключается в том, что различные потоки имеют различные пути выполнения, но при этом пользуются обшей памятью. Таким образом, несколько порожденных в программе потоков, могут пользоваться глобальными переменными, и любое изменение данных одним потоком, будет доступно и для всех остальных.

Существует небольшое число моделей построения многопоточных приложений. Опишем кратко основные. Итеративный параллелизм используется для реализации нескольких потоков (часто идентичных), каждый из которых содержит циклы. Потоки программы, описываются итеративными функциями и работают совместно над решением одной задачи.

**Постановка задачи**

Найти алгебраическое дополнение для каждого элемента матрицы. Входные данные: произвольная матрица А размерности n х n. Заполнение начальной матрицы реализовать в главном потоке приложения.

**Требования**

1. Реализовать последовательный вариант программы для указанного варианта.

2. Реализовать параллельный вариант программы. Количество потоков выполнения должно являться входным параметром задачи.

3. После завершения программа должна выдавать время своей работы. Подобрать размеры матриц таким образом, чтобы время работы последовательного варианта составляло не менее одной секунды.

4. Посчитать параметры качества вашей параллельной программы и построить в Excel графики для количества потоков = [1;16] и количестве задействованных ядер = [1;4] (используйте функцию SetAffinityMask). Всего 12 графиков.

A. Время выполнения

B. Ускорение

C. Эффективность распараллеливания.

5. Результат работы оформить в DOC файле (распечатывать не надо):

A. Титульный лист

B. Постановка задачи

C. Листинг

D. Графики

E. Выводы

**Листинг (Debug x64)**

Время выполнения

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ядра\Потоки | 1 | 2 | 3 | 4 | 8 | 12 | 16 |
| 1 | 3916 | 3758 | 3334 | 3294 | 2977 | 3600 | 3859 |
| 2 | 3185 | 2495 | 2759 | 2454 | 2233 | 2875 | 2380 |
| 3 | 3368 | 2677 | 2453 | 2347 | 2216 | 2251 | 2347 |
| 4 | 3616 | 2664 | 2383 | 2265 | 2263 | 2161 | 2145 |

Ускорение

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ядра\Потоки | 1 | 2 | 3 | 4 | 8 | 12 | 16 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1,2 | 1,5 | 1,2 | 1,3 | 1,3 | 1,3 | 1,5 |
| 3 | 1,2 | 1,4 | 1,4 | 1,4 | 1,4 | 1,6 | 1,7 |
| 4 | 1,1 | 1,4 | 1,4 | 1,5 | 1,4 | 1,7 | 1,8 |

Эффективность распараллеливания

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ядра\Потоки | 1 | 2 | 3 | 4 | 8 | 12 | 16 |
| 1 | 1 | 0,5 | 0,33 | 0,25 | 0,125 | 0,08 | 0,06 |
| 2 | 1,2 | 0,7 | 0,4 | 0,3 | 0,16 | 0,10 | 0,12 |
| 3 | 1,2 | 0,7 | 0,5 | 0,4 | 0,17 | 0,13 | 0,13 |
| 4 | 1,1 | 0,7 | 0,5 | 0,4 | 0,18 | 0,15 | 0,16 |

Вывод

Многопоточная программа работает быстрее чем однопоточная. При 2 ядрах ускорение в среднем 1,2 раза, при 3 ядрах – в 1,4 раза, при 4 ядрах – в 1,5. С увеличением количества ядер уменьшается время работы многопоточной программы. Так как процессор мог быть занят ещё какими-то задачами, результаты могут быть не совсем точными.