Ибраев Ерлан Иржанович 184-1 МОиАИС

**Цель работы:** Получить общие представления о потоках, о способах их создания и использования.

**Информация о компьютере:** Intel® Core™ i7-8750H, 6 ядер, 2.20GHz

**Таблица 1. Сравнение времени последовательных и параллельных вычислений.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Организация алгоритма | Последовательная обработка | | | Параллельная обработка | | |
| общее время работы | время работы одного метода | порядок  вывода | общее время работы | время работы одного метода | порядок  вывода |
| 1 блок | 1974 | 1974 | A | 1817 | 1817 | A |
| 2 блока | 3889 | 1944 | A, B | 2457 | 1227 | A, B |
| 3 блока | 5916 | 1972 | A, B, C | 2643 | 881 | B, A, C |
| 4 блока | 7802 | 1950 | A, B, C, D | 2833 | 708 | C, A, B, D |
| 5 блоков | 8938 | 1787 | A, B, C, D, E | 2872 | 574 | C, A, B, D, E |
| 6 блоков | 11277 | 1879 | A, B, C, D, E, F | 2898 | 483 | B, F, D, A, E, C |
| 7 блоков | 12647 | 1806 | A, B, C, D, E, F, G | 2969 | 424 | C, E, A, F, B, D, G |
| 8 блоков | 15321 | 1915 | A, B, C, D, E, F, G, H | 3082 | 385 | F, H, A, E, D, B, C, G |

**Таблица 2. Влияние приоритета потока на время вычислений.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Число блоков | Среднее время работы одного экземпляра метода | | | | |
| Lowest | BelowNormal | Normal | AboveNormal | Highest |
| 1 блок | 1721 | 2102 | 1921 | 1992 | 2132 |
| 2 блока | 1070 | 1114 | 1133 | 1202 | 1157 |
| 3 блока | 870 | 836 | 882 | 844 | 857 |
| 4 блока | 685 | 706 | 697 | 706 | 690 |
| 5 блоков | 573 | 586 | 573 | 573 | 572 |
| 6 блоков | 488 | 479 | 486 | 482 | 478 |
| 7 блоков | 423 | 428 | 432 | 425 | 420 |
| 8 блоков | 380 | 372 | 371 | 384 | 373 |

**Заключение:**

Таблица 1.

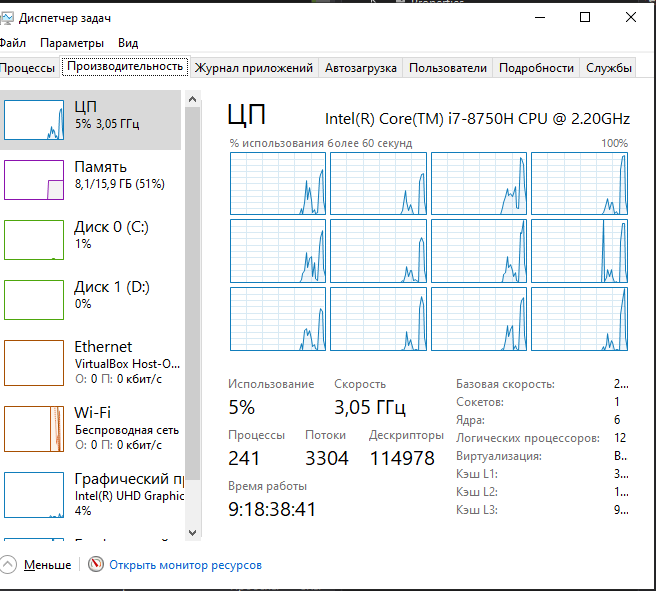
* Параллельное вычисление обрабатывается быстрее чем последовательная.
* Чем больше блоков тем быстрее обрабатывается параллельные вычисления.
* Самое минимальное время параллельной обработки одного метода – при блоках 8.
* Общее время последовательной вычисления линейно увеличивается О(n)=xn.
* Время при 1 блоке незначительно отличается.

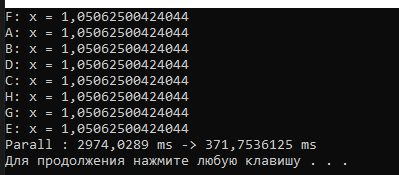
Таблица 2:

* Время вычисления не значительно отчитается от приоритета 8 блоков.

**Скриншоты:**

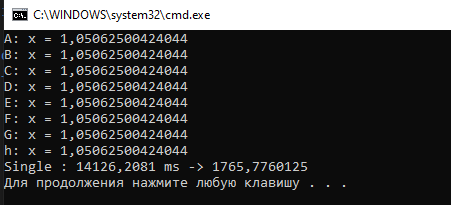
**Параллельная обработка:**

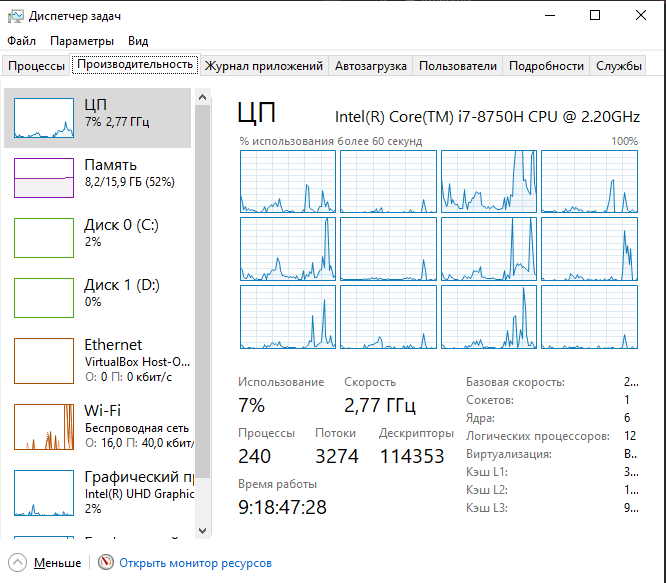




При параллельном вычисления практически равная нагрузка.

**Последовтельная обработка**





**Код:**

static void Work(string s)

{

double x = 1.0;

for (int i=0; i < 5000; i++)

for (int j =0; j < 5000; j++)

{

x = Math.Sqrt(x);

x = x + 0.000000001;

x = Math.Pow(x, 2);

}

Console.WriteLine("{0}: x = {1}", s, x);

}

static void Main(string[] args)

{

DateTime to = DateTime.Now;

Work("A");

Work("B");

Work("C");

Work("D");

Work("E");

Work("F");

Work("G");

Work("h");

double dt = (DateTime.Now - to).TotalMilliseconds;

Console.WriteLine("Single : {0} ms -> {1}", dt, dt / 8);

//var th1 = new Thread(\_ => Work("A"));

//var th2 = new Thread(\_ => Work("B"));

//var th3 = new Thread(\_ => Work("C"));

//var th4 = new Thread(\_ => Work("D"));

//var th5 = new Thread(\_ => Work("E"));

//var th6 = new Thread(\_ => Work("F"));

//var th7 = new Thread(\_ => Work("G"));

//var th8 = new Thread(\_ => Work("H"));

//th1.Start();

//th2.Start();

//th3.Start();

//th4.Start();

//th5.Start();

//th6.Start();

//th7.Start();

//th8.Start();

//th1.Join();

//th2.Join();

//th3.Join();

//th4.Join();

//th5.Join();

//th6.Join();

//th7.Join();

//th8.Join();

//double dt = (DateTime.Now - to).TotalMilliseconds;

//Console.WriteLine("Parall : {0} ms -> {1} ms ", dt, dt / 8);

//var th1 = new Thread(\_ => Work("A"));

//var th2 = new Thread(\_ => Work("B"));

//var th3 = new Thread(\_ => Work("C"));

//var th4 = new Thread(\_ => Work("D"));

//var th5 = new Thread(\_ => Work("E"));

//var th6 = new Thread(\_ => Work("F"));

//var th7 = new Thread(\_ => Work("G"));

//var th8 = new Thread(\_ => Work("H"));

//th1.Priority = ThreadPriority.Highest;

//th2.Priority = ThreadPriority.Highest;

//th3.Priority = ThreadPriority.Highest;

//th4.Priority = ThreadPriority.Highest;

//th5.Priority = ThreadPriority.Highest;

//th6.Priority = ThreadPriority.Highest;

//th7.Priority = ThreadPriority.Highest;

//th8.Priority = ThreadPriority.Highest;

//th1.Start();

//th2.Start();

//th3.Start();

//th4.Start();

//th5.Start();

//th6.Start();

//th7.Start();

//th8.Start();

//th1.Join();

//th2.Join();

//th3.Join();

//th4.Join();

//th5.Join();

//th6.Join();

//th7.Join();

//th8.Join();

//double dt = (DateTime.Now - to).TotalMilliseconds;

//Console.WriteLine("Parall : {0} ms -> {1} ms ", dt, dt / 8);

}