Технологии параллельных систем и распределенных вычислений

Лабораторная работа №6. Введение в OpenMP

Project -> Build options -> Other options -> добавить флаг "-fopenmp"
Linker settings -> "Add" -> "lgomp"

Установка необходимых программных средств.

Пример 1. Умножение матриц

```
#include <stdio.h>
#include <omp.h>
#define N 4096
double a[N][N], b[N][N], c[N][N];
int main()
{
     int i, j, k;
     double t1, t2;
     // инициализация матриц
      for (i=0; i<N; i++)
           for (j=0; j<N; j++)
                a[i][j]=b[i][j]=i*j;
      t1=omp_get_wtime();
     // основной вычислительный блок
     #pragma omp parallel for shared(a, b, c) private(i, j, k)
      for(i=0; i<N; i++)
     {
           for(j=0; j<N; j++)
          {
               c[i][j] = 0.0;
               for(k=0; k<N; k++)
                    c[i][j]+=a[i][k]*b[k][j];
          }
      }
      t2=omp_get_wtime();
      printf("Time=%lf\n", t2-t1);
}
```

Пример 2. Вычисление числа π

```
#include <stdio.h>
double f(double y)
{
     return(4.0/(1.0+y*y));
}
int main()
{
     double w, x, sum, pi;
      int i;
      int n = 1000000;
      w = 1.0/n;
      sum = 0.0;
```

```
    #pragma omp parallel for private(x) shared(w)\
reduction(+:sum)
    for(i=0; i < n; i++)
    {
        x = w*(i-0.5);
        sum = sum + f(x);
    }
    pi = w*sum;
    printf("pi = %f\n", pi);
}
```

Задание:

1. Скомпилировать первый пример. Объяснить принцип работы алгоритма. Запустить код при числе потоков 1, 2, 3, 4. Построить график зависимости времени работы программы от числа потоков.
2. Скомпилировать второй пример. Объяснить принцип работы алгоритма. Запустить код при числе потоков 1, 2, 3, 4. Построить график зависимости времени работы программы от числа потоков.
3. Разработать параллельный алгоритм и найти сумму чисел от 1 до N, где N задает пользователь. При фиксированном N построить график зависимости времени работы программы от числа потоков.