###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«Параллельная реализация решения системы линейных алгебраических уравнений с помощью OpenMP»

студента 2 курса, 22208 группы

**Лебедева Антона Андреевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А.Ю. Кудинов

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение Листинг 6](#_Toc18443925)

# ЦЕЛЬ

Разработать и исследовать параллельные программы решения СЛАУ методом простой итерации с применением одной из библиотек, реализующих стандарты OpenMP.

# ЗАДАЧА

# Написать последовательную программу, реализующую итерационный алгоритм решения системы линейных алгебраических уравнений вида Ax=b, распараллелить с помощью OpenMP.

# Реализовать два варианта программы:

# Вариант 1: для каждого распараллеливаемого цикла создается отдельная параллельная секция #pragma omp parallel for,

# Вариант 2: создается одна параллельная секция #pragma omp parallel, охватывающая весь итерационный алгоритм.

# Замерить время работы двух вариантов программы при использовании различного числа процессорных ядер: от 1 до числа доступных в узле. Построить графики зависимости времени работы программы, ускорения и эффективности распараллеливания от числа используемых ядер. Исходные данные и параметры задачи подобрать таким образом, чтобы решение задачи на одном ядре занимало не менее 30 секунд.

# Провести исследование на определение оптимальных параметров #pragma omp for schedule(...) при некотором фиксированном размере задачи и количестве потоков.ОПИСАНИЕ РАБОТЫ

*Выбираю значение параметра N таким, чтобы время работы программы было порядка 30 секунд относительно -O0.*

*Для меня это:*

**

команды для ее компиляции:

*gcc -O0 Source.c -o dO0 -Wall*

*скомпилирую компилятором GCC с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og по порядку:*

|  |  |
| --- | --- |
| *уровень оптимизации* | *Время выполнения программы* |
| *-O0* | *34.203* |
| *-O1* | *13.868* |
| *-O2* | *12.791* |
| *-O3* | *12.734* |
| *-Os* | *17.330* |
| *-Ofast* | *12.717* |
| *-Og* | *13.123* |

*Проверяю то же самое, но с другим N:*

****

|  |  |
| --- | --- |
| *уровень оптимизации* | *Время выполнения программы* |
| *-O0* | *62.221* |
| *-O1* | *23.300* |
| *-O2* | *23.231* |
| *-O3* | *23.254* |
| *-Os* | *31.797* |
| *-Ofast* | *23.244* |
| *-Og* | *24.113* |

*Можно прийти к выводу что лучшие уровни оптимизации, в моём случае это: -O2, -O3 и -Ofast. Но если бы мне пришлось выбирать, я бы выбрал именно -O2, т.к. вес с компилированной программы обычно больше у -O3, а у*

*-Ofast менее точная компиляция математических функций.*

**ЗАКЛЮЧЕНИЕ**

# *Изучил основные функции оптимизирующего компилятора, и некоторые примеры оптимизирующих преобразований и уровней оптимизации.*

# *Получил базовые навыки работы с компилятором GCC.*

# *Исследовал влияния оптимизационных настроек компилятора GCC на время исполнения программы.*

# Листинг

#include <windows.h>

#include <stdio.h>

#include <time.h>

double funcPi(long long int N) {

long double pi = 0;

int isMinus = 1;

for (long long int n = 0; N >= n; n++) {

pi += (long double)((long double)(4 \* isMinus) / (long double)(2 \* n + 1));

isMinus \*= -1;

}

return pi;

}

int main() {

//cout.precision(30);

long long int N = 4099900000;

printf("%llf\n", funcPi(N));

//cout << "runtime = " << clock() / 1000.0 << endl;

printf("%llf", clock() / 1000.0);

}