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

«ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА»

студента 2 курса, 23209 группы
Инокова Семёна Шухратовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
А. Ю. Кудинов

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.

2. Получение базовых навыков работы с компилятором GCC.

3. Исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы

ЗАДАНИЕ

1. Написать на языке C или C++ алгоритм вычисления числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда.

2. Программу скомпилировать компилятором GCC с уровнями оптимизации **-O0, -O1, -O2, -O3, -Os, -Ofast, -Og** под архитектуру процессора x86.

3. Для каждого из семи вариантов компиляции измерить время работы программы при нескольких значениях N.

ОПИСАНИЕ РАБОТЫ

1. Для написания кода, который реализует алгоритм вычисления числа Пи был выбран язык C.

2. Произвёл замеры времени при различных n, нашёл n, при котором время составляет около 30 секунд (4.350.000.000).

3. Произвёл замеры времени для каждого из уровней компиляции при различных N.

4. Показал на графике зависимость времени выполнения программы от N при каждых уровнях компиляции.

5. Сравнил размеры бинарных файлов.

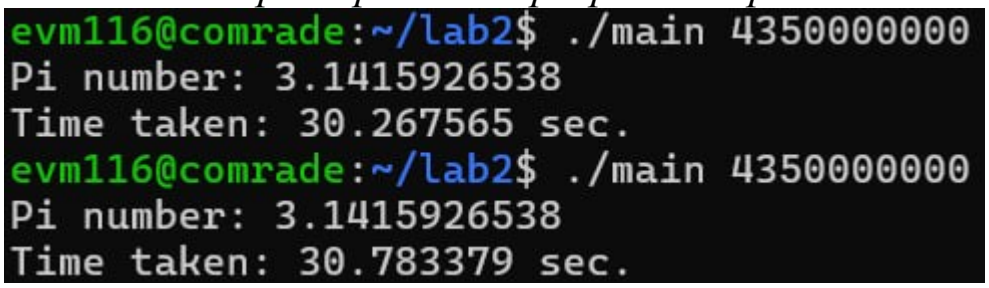
ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы я познакомился с различными оптимизирующими преобразованиями и уровнями оптимизации компилятора GCC. Я изучил влияние настроек оптимизирующего компилятора GCC на время выполнение программы и её размер.

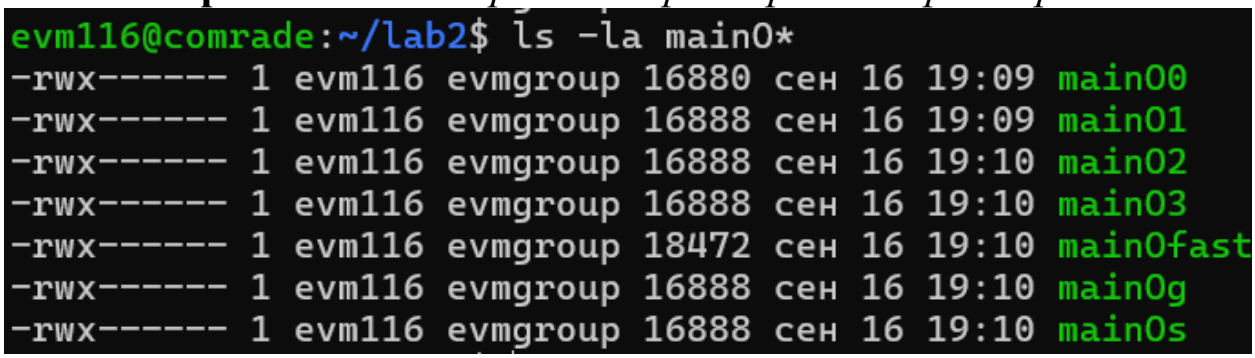
Приложение 1. *Код программы для работы*

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>

double piCalculation(long long n) {
    double pi = 4;

    for (long long i = 1; i <= n; ++i) {
        double x = 4. / (2*i + 1);
        if (i % 2) {
            pi -= x;
        } else {
            pi += x;
        }
    }

    return pi;
}

int main(int argc, char *argv[]) {
    struct timespec start, end;
    long long n = strtoll(argv[1], NULL, 10);

    clock_gettime(CLOCK_MONOTONIC_RAW, &start);
    double pi = piCalculation(n);
    clock_gettime(CLOCK_MONOTONIC_RAW, &end);

    printf("Pi number: %.10lf\n", pi);
    printf("Time taken: %lf sec.\n", end.tv_sec-start.tv_sec + 0.000000001*(end.tv_nsec-start.tv_nsec));

    return 0;
}
```

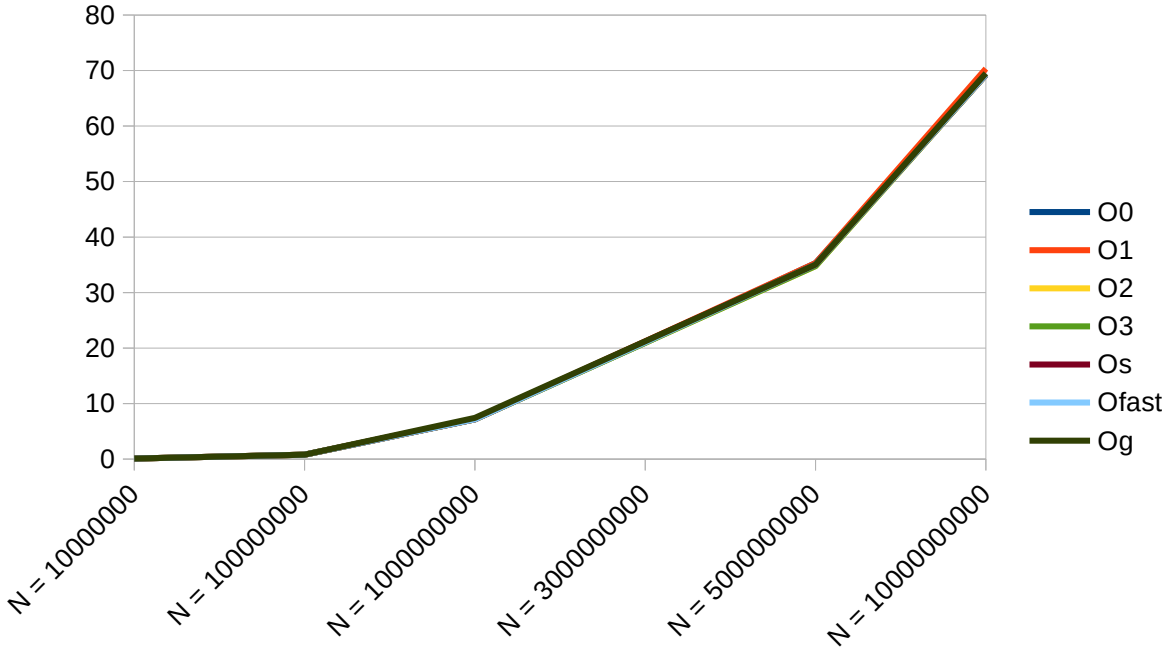
Приложение 2. *Время работы программы при n = 4350000000*



Приложение 3. *Сравнение размеров бинарных файлов*



Приложение 4. *Измерение времени работы при различных n*



N	O0	O1	O2	O3	Os	Ofast	Og
N = 10000000	0.083601	0.085194	0.085129	0.085106	0.083681	0.083369	0.084275
N = 100000000	0.794868	0.794752	0.793612	0.793646	0.793763	0.794942	0.793465
N = 1000000000	7.161636	7.214852	7.161570	7.160400	7.158115	7.158549	7.418033
N = 3000000000	20.965253	21.222606	20.959081	20.962623	21.171998	21.050555	21.217890
N = 5000000000	35.288263	35.291858	34.767023	34.763648	35.039606	35.022141	35.030607
N = 10000000000	69.246933	70.337789	69.252864	69.252843	69.252446	69.254550	69.512713