###### 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 Бородина Артёма Максимовича 2 курса, 19205 группы

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

к.т.н, доцент

А.Ю. Власенко

Новосибирск 2020

**СОДЕРЖАНИЕ**

ЦЕЛЬ 3

ЗАДАНИЕ 3

ОПИСАНИЕ РАБОТЫ 4

ЗАКЛЮЧЕНИЕ 5

Приложение *1.**Скрипт* 6

Приложение *2.**График* 7

Приложение *3.**Хеш-суммы* 8

# 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 вариант:*

Алгоритм вычисления числа π с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда.

1. Написать программу на языке C или C++, которая реализует выбранный алгоритм из задания.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 30-60 секунд.

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

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

6. Составить отчет по лабораторной работе.

**ОПИСАНИЕ РАБОТЫ**

1. Изучено понятие компилятора, его характеристики и основные функции, а также уровни компиляции, примеры оптимизирующих преобразований в GCC и их способы указания.
2. Для замеров времени была использована ранее написанная программа для вычисления числа π с помощью формулы Лейбница.
3. Был написан bash скрипт (Приложение 1) способный выдавать таблицу формата .csv с результатами замеров.
4. Мною были проведены по 11 замеров времени работы программы (Приложение 2) на каждый уровень оптимизации GCC:

при N от 3\*108 до 4\*109 (с шагом в 3 \* 108).

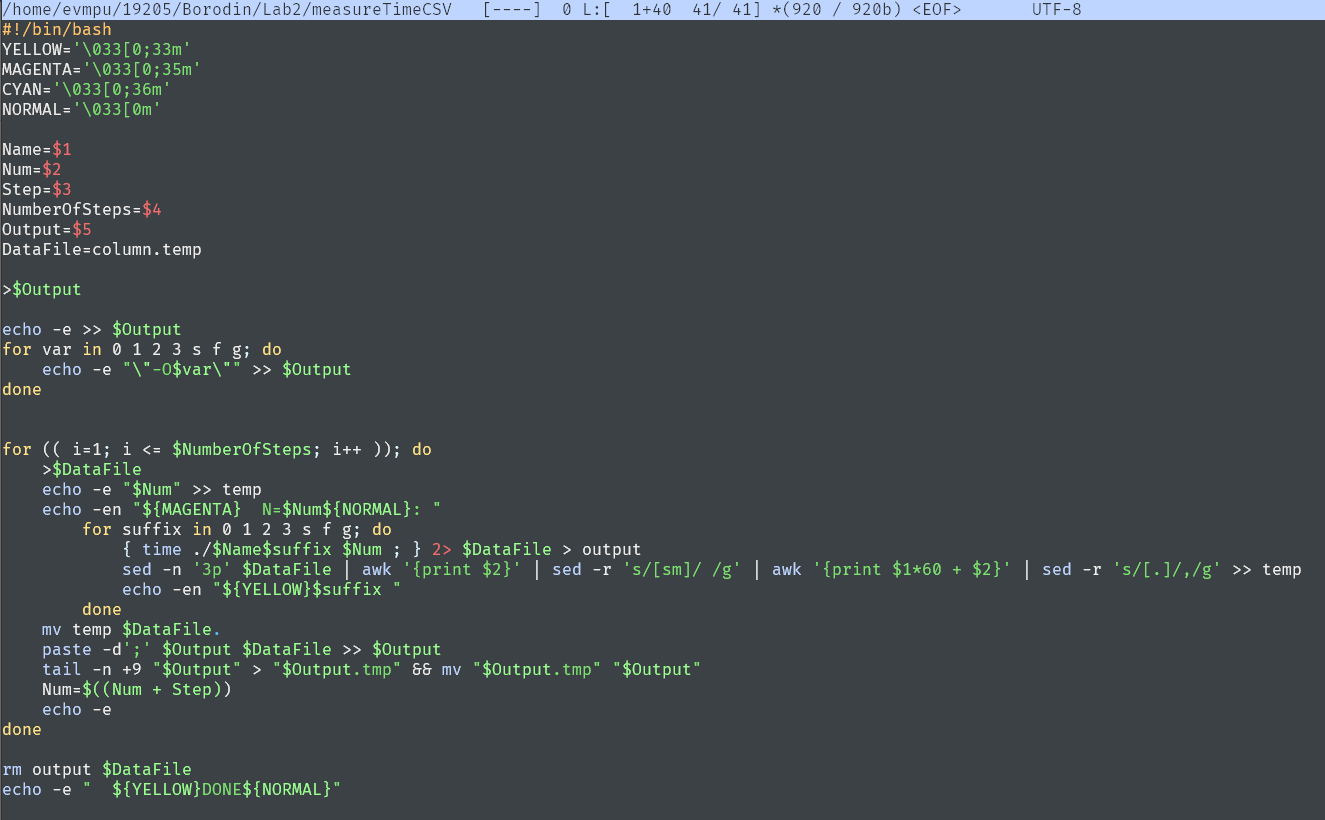
1. Определение времени работы программы осуществлялось с помощью unix-утилиты time.
2. Был построен график зависимости времени от N для каждого уровня оптимизации GCC.
3. Из полученных данных видно, что время всех уровней оптимизации растет линейно с увеличение N. O2 и O3 имеют почти одинаковое время так как их исходные файлы одинаковые. Это можно увидеть, проверив их хеш-сумму (Приложение 3). Самым быстрым по времени, как и самым большим по размеру, уровнем оптимизации оказался Ofast. Время O1 и Og почти равно, потому что они оба компилируются почти без каких-либо оптимизаций. Уровень Os по времени сравним с O2 и O3, но при этом имеет наименьший размер.

**ЗАКЛЮЧЕНИЕ**

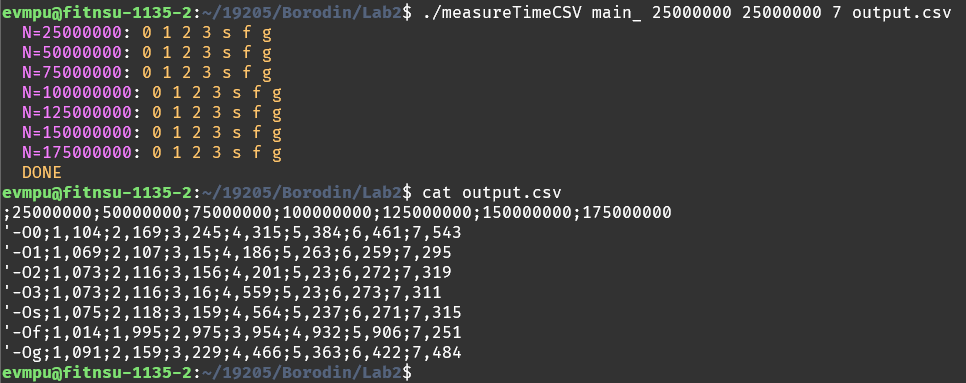
В ходе выполнения лабораторной работы были изучены основы компилятора, уровни оптимизации и примеры оптимизации. Было исследовано влияние уровней оптимизации на время исполнения программы.

**Приложение 1.** Скрипт

*Скрипт принимает на вход 5 параметров: название файла, начальное число, шаг, кол-во шагов, файл вывода*



*Пример вывода:*



**Приложение 2.** График

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N, млн | -O0 | -O1 | -O2 | -O3 | -Os | -Ofast | -Og |
| 400 | 17,139 | 16,613 | 16,655 | 16,737 | 16,65 | 15,677 | 17,107 |
| 700 | 29,945 | 28,981 | 29,089 | 30,394 | 30,448 | 27,389 | 29,802 |
| 1000 | 43,93 | 41,44 | 43,019 | 41,53 | 41,535 | 39,107 | 42,554 |
| 1300 | 55,568 | 53,851 | 53,969 | 55,342 | 53,971 | 50,819 | 55,013 |
| 1600 | 68,373 | 66,262 | 66,445 | 66,455 | 66,481 | 64,852 | 68,052 |
| 1900 | 81,185 | 78,669 | 78,844 | 78,806 | 78,858 | 74,248 | 80,797 |
| 2200 | 95,189 | 91,338 | 91,895 | 91,895 | 91,882 | 86,801 | 94,332 |
| 2500 | 108,148 | 103,801 | 104,41 | 104,462 | 104,404 | 98,628 | 106,69 |
| 2800 | 121,118 | 116,099 | 116,156 | 116,926 | 117,012 | 110,46 | 119,534 |
| 3100 | 134,091 | 128,534 | 129,44 | 129,435 | 129,421 | 122,288 | 132,336 |
| 3400 | 147,064 | 140,968 | 142,018 | 141,885 | 141,935 | 134,067 | 145,751 |
| 3700 | 160,037 | 153,388 | 154,473 | 154,713 | 154,455 | 145,894 | 157,941 |
| 4000 | 177,045 | 165,838 | 166,997 | 166,985 | 166,963 | 157,774 | 174,829 |

**Приложение 3.** Хеш-суммы

