## «ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА»

## Отчёт по лабораторной работе №2

## Выполнил: Ланин Д.М. ФИТ НГУ 2 курс

## Преподаватель: Артюхов А. А.

## Цели лабораторной работы:

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 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.

**Для вычислений применяется алгоритм вычисления числа Пи методом Монте-Карло.**

*Далее подробно об алгоритме* : cначала в квадрат с центром в начале координат и со стороной два вписывается круг с единичным радиусом. Затем в этом квадрате случайным образом с равномерным распределением генерируются N точек. Точка может попасть в окружность или нет (условие попадания x2 + y 2 ≤ 1). Далее определяется число M точек, попавших в круг. При достаточно большом числе бросков N, по значениям M и N вычисляется число Пи: Pi = 4 \* M / N.

**Описание методики для определения времени работы программы:**

Для измерения времени исполнения кода я использовал функцию

clock()

 из модуля

time.h

 — она возвращает число таков, измеряемое процессором от начала выполнения программы.  
Глобальная константа

CLOCKS\_PER\_SEC

 хранит число тактов, выполняемое процессором в секунду. Соответственно, чтобы получить время работы программы в секундах достаточно результат работы функции разделить на эту константу:

clock() / CLOCKS\_PER\_SEC;

## Результаты измерений

Измерения проводились для процессора на архитектуре AMD Picasso при зафиксированной частоте на уровне 2.10GHz.

Начало данной работы заключается в написании программы¸ которая содержит в себе алгоритм вычисления определённого интеграла сложной функции методом трапеции. После этого нужно применить различные уровни оптимизации GCC, которые включают в себя:

· O0, где все оптимизации отключены;

· O1, где включены оптимизации для уменьшения размера бинарного исполняемого файла и такие оптимизации, уменьшающие время работы программы, которые не сильно замедляют работу компилятора;

· O2, где включены практически все доступные оптимизации, кроме тех, что ускоряют вычисления за счет увеличения размера кода;

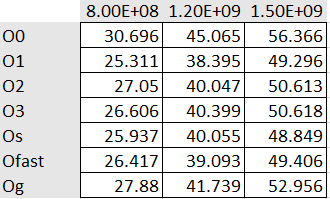
· O3, где включены все оптимизации из уровня O2, к ним добавлены оптимизации времени работы программы, которые могут приводить к увеличению размера бинарного исполняемого файла;

· Os, который служит для оптимизации размера программ, в него включено подмножество оптимизаций из уровня O2;

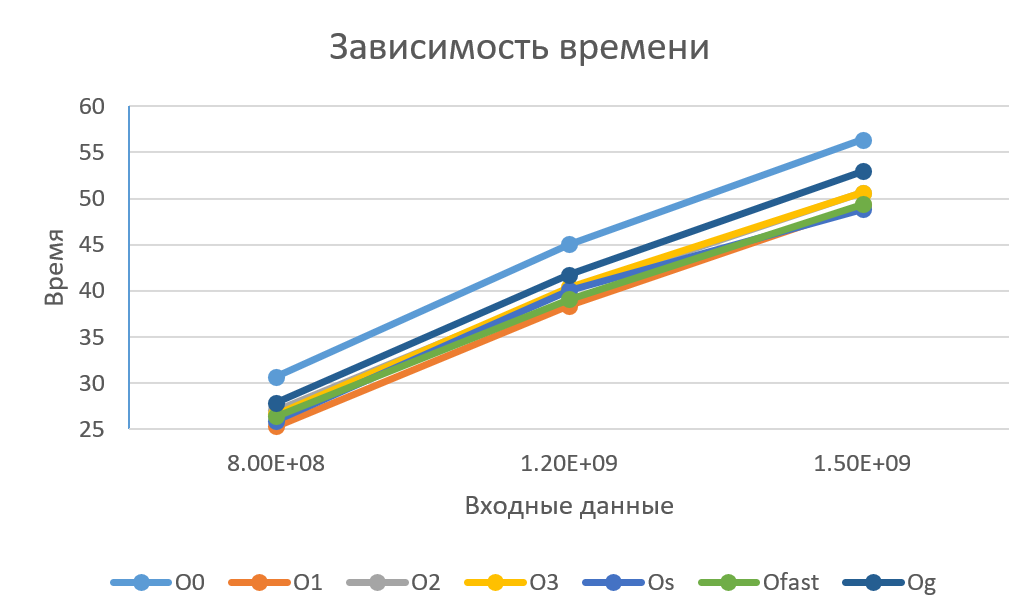
· Og, где происходят все оптимизации, которые сохраняют возможность просмотра стека вызовов, фрагментов исходного текста программы, относящихся к разным уровням этого стека, и возможность приостановки программы для каждой строки исходного текста, содержащей операторы.

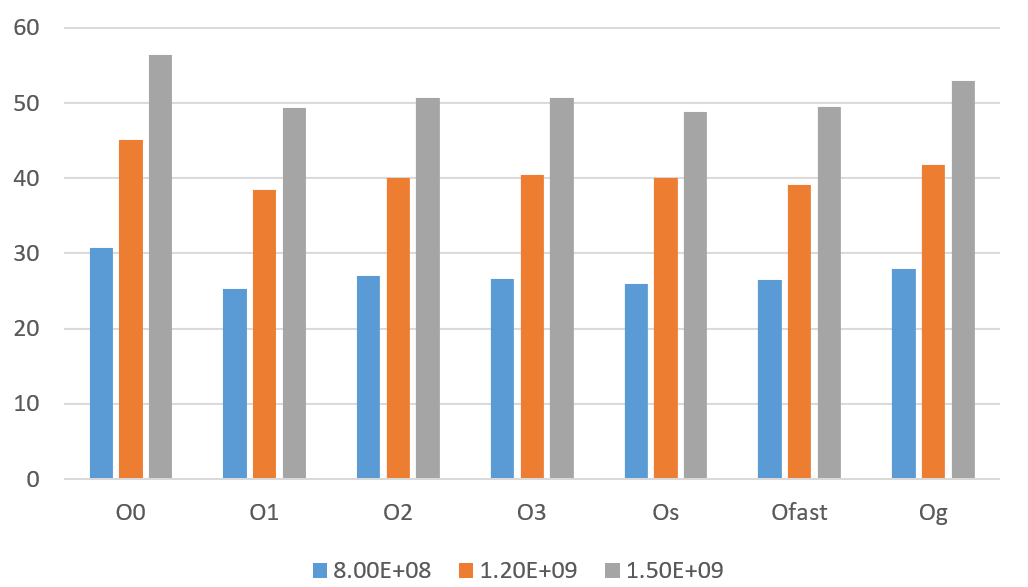
После компиляции программы с каждым из видов оптимизаций нужно измерять время работы программы. Для этого мы вводим сколько интервалов использовать при подсчёте значения.

Ниже представлена таблица со значениями:



По получившимся значениям построим график зависимости





## Вывод:

В ходе выполнения работы были изучены различные уровни оптимизации (O1, O2, O3, Os, Ofast, Og), а также проведены тесты с их применением для оценки корректности работы программы и выявления зависимости времени выполнения от входных данных.