###### **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

###### **РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### **НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

###### 

###### 

**ЭВМ И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА

Студент:

Овчаренко Дарья Ивановна, группа 23211

Преподаватель:

Ассистент кафедры ПВ ФИТ

Мичуров Михаил Антонович

Новосибирск 2024

# **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**

Вариант задания: 4.

1. Написать программу на языке C или C++, содержащую функцию, которая реализует выбранный алгоритм из задания. Программа должна принимать значение N через параметр в командной строке.

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**

В ходе задания использовался компьютер с архитектурой x86\_64, с операционной системой Ubuntu 22.04.5 LTS и процессором AMD A6-6310 APU with AMD Radeon R4 Graphics.

###### **Пошаговое описание выполненной работы**

1. Был создан файл lab2.c
2. Был подобран такой аргумент n (количество членов, по которым будет выполнено разложение), чтобы время выполнения программы было 30-60 секунд. n = 2 500 000 000. Аргумент x (угол, синус которого нужно почитать) пусть будет равен 30 во всех запусках программ.
3. Скомпилируем программу с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og. Для каждого уровня оптимизации замерим время работы программы при подобранном значении n = 2 500 000 000; 0,5n = 1 250 000 000; 1,5n = 5 000 000 000.
4. Код скомпилирован командой **gcc lab2.c -o lab2**, где lab2 - исполнительный файл.
5. Запуск программы производится с помощью команды **./lab2 argv[1] argv[2].**
6. Команды компиляции c оптимизацией и запуска отразим в таблице (см. Таблицу 1).

Табл1. Команды компиляции и запуска программы

| Уровень оптимизации | Строка компиляции | Строка запуска |
| --- | --- | --- |
| -O0 | gcc -O0 lab2.c -o lab2\_0 | ./lab2\_0 30 2500000000 |
| -O1 | gcc -O1 lab2.c -o lab2\_1 | ./lab2\_1 30 2500000000 |
| -O2 | gcc -O2 lab2.c -o lab2\_2 | ./lab2\_2 30 2500000000 |
| -O3 | gcc -O3 lab2.c -o lab2\_3 | ./lab2\_3 30 2500000000 |
| -Os | gcc -Os lab2.c -o lab2\_s | ./lab2\_s 30 2500000000 |
| -Ofast | gcc -Ofast lab2.c -o lab2\_fast | ./lab2\_fast 30 2500000000 |
| -Og | gcc -Og lab2.c -o lab2\_g | ./lab2\_g 30 2500000000 |

7. Полученные результаты отразим в Таблице 2.

Табл2.Результаты измерений работы программы

| Уровень оптимизации | Время, сек | | |
| --- | --- | --- | --- |
| 0,5n | n | 1,5n |
| Без оптимизации | 21.631513 | 43.580939 | 90.419840 |
| -O0 | 22.700718 | 48.250730 | 89.741782 |
| -O1 | 17.218743 | 34.007242 | 69.994314 |
| -O2 | 17.237169 | 34.515610 | 68.897210 |
| -O3 | 17.782501 | 33.703800 | 65.903225 |
| -Os | 16.482802 | 34.290155 | 66.944533 |
| -Ofast | 14.780583 | 28.945991 | 55.461372 |
| -Og | 17.803346 | 32.819320 | 66.070765 |

8. Графическое представление результатов экспериментов приведено на графиках зависимости различных значений n от времени (см. Рис.1).

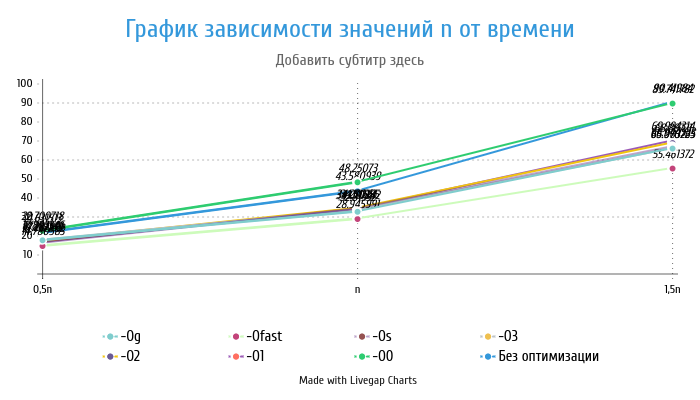
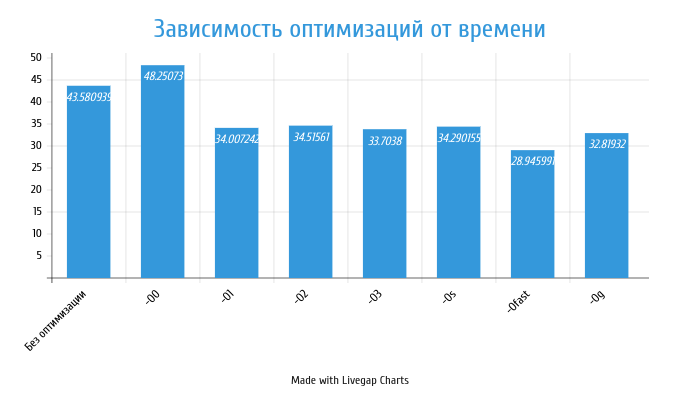


Рис1. График зависимости значений n от времени

  
Рис2. График зависимости оптимизаций от времени

# **ЗАКЛЮЧЕНИЕ**

В нашей работе мы узнали базовые команды для компиляции и запуска программы, изучили основные флаги для оптимизации и как они влияют на время работы программы.

По итогу работы были представлены различные способы оптимизации и определены зависимости этих оптимизаций от времени и n-членов, по которым необходимо разложить синус в ряд Тейлора.

В ходе выполнения задания было выявлено, что код без оптимизации и код с оптимизацией -O0 работают дольше всего; оптимизации -O1, -O2 и -Os работают быстрее упомянутых выше, но они немного медленнее чем -O3 и -Og. Оптимизация -Ofast показала себя быстрее всего.

**ПРИЛОЖЕНИЕ 1.** Листинг программы с библиотечной функцией clock\_gettime

#define \_POSIX\_C\_SOURCE 199309L

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#define PI 3.1415926535897

double CalcSin(double x, long long n);

int main(int argc, char \*\*argv){

struct timespec res, start, end;

if (clock\_getres(CLOCK\_MONOTONIC\_RAW, &res) == 0){ // сравнивается с 0, так как 0 значит успешное выполнение

printf("Timer resolution: %ld sec, %ld nanosec.\n", res.tv\_sec, res.tv\_nsec);

} else {

perror("Call error clock\_getres!");

}

//clock\_gettime (CLOCK\_MONOTONIC\_RAW, &start);

if (argc > 3){

printf("Bad input. Enter x and n in command line.");

return -1;

}

char \*endptr\_x, \*endptr\_n;

long x = strtol(argv[1], &endptr\_x, 10);

long long n = strtol(argv[2], &endptr\_n, 10);

if (\*endptr\_x != '\0'){

printf("Error: Invalid input for x: %s\n", argv[1]);

return -1;

}

if (\*endptr\_n != '\0'){

printf("Error: Invalid input for n: %s\n", argv[2]);

return -1;

}

//double x = atol(argv[1]);

//long long n = atol(argv[2]);

clock\_gettime (CLOCK\_MONOTONIC\_RAW, &start);

double sinx = CalcSin((double)x, n);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("%lf\n", sinx);

//clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Time takken: %lf sec.\n", end.tv\_sec - start.tv\_sec + 0.000000001\*(end.tv\_nsec-start.tv\_nsec));

return 0;

}

double CalcSin(double x, long long n){

double sinx = 0;

x = x \* PI / 180; // перевод градусы в радианы, иначе некорректные вычисления

double sum = x;

for (long long i = 1; i <= 2 \* n - 1; i += 2){

sinx += sum;

sum = (sum \* x \* x \* (-1)) / ((i + 1) \* (i + 2));

}

return sinx;

}

Ссылка на репозиторий с кодом: <https://github.com/dadashasha/nsu_evm/blob/main/lab2>