# ЭВМ и периферийные устройства

## Отчёт по лабораторной работе №1

## «ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА»

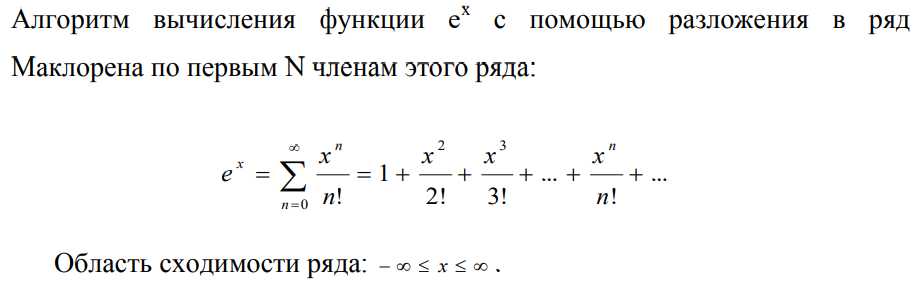
## Выполнил: Скопинцев Н.А. ФИТ НГУ 2 курс

## Преподаватель: Иванишкин Д.С.

## Цель 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.

## Вариант задания



## Результаты измерений

В таблице ниже приведены результаты работы программы и время,

потраченное на исполнение алгоритма при разных ключах компиляции:

|  |  |  |  |
| --- | --- | --- | --- |
| Оптимизация\N | 4.000.000.000 | 6.000.000.000 | 10.000.000.000 |
| -O0 | 31.617393 | 47.155917 | 79.030256 |
| -O1 | 15.309448 | 22.837941 | 38.419855 |
| -O2 | 15.311317 | 22.636621 | 37.895664 |
| -O3 | 15.571909 | 22.840007 | 37.966020 |
| -Os | 15.309687 | 22.862945 | 37.733032 |
| -Ofast | 15.310768 | 22.838019 | 37.669104 |
| -Og | 15.090132 | 22.836791 | 38.215444 |

\*Дополнительно:

Время работы с разными уровнями оптимизациями на локальном сервере ( ноутбук на x86 архитектуре, микроархитектура – Kaby Lake)

|  |  |
| --- | --- |
| Оптимизация\N | 4000000000 |
| -O0 | 54.449195 |
| -O1 | 30.887781 |
| -O2 | 31.009573 |
| -O3 | 30.890805 |
| -Os | 31.122271 |
| -Ofast | 30.909696 |
| -0g | 31.296318 |

## Листинг программы

#include <stdio.h>

#include <time.h>

#define N 4000000000

double Exp(int x) {

double tmp = x;

double res = 1;

long long n = N;

for (long long i = 1; i <= n; i++) {

res += tmp;

tmp = (tmp \* x) / (i + 1);

//printf("%Lf\n", res);

}

return res;

}

int main() {

struct timespec start, end;

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

double e = Exp(7);

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

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

printf("Time taken: %lf sec.\n", end.tv\_sec - start.tv\_sec

+ 0.000000001 \* (end.tv\_nsec - start.tv\_nsec));

return 0;

}

## Вывод

В результате проведенных исследованиях можно убедиться, что при разных значениях N (от 4 млрд. до 10 млрд.) самый первый уровень оптимизации сокращает выполнения программы больше, чем в 2 раза. Следующие же уровни оптимизации практически не дают выигрыша во времени работы программы, дельта всего несколько десятых секунд.

Стоит отметить, что уровни оптимизации **-Os, -Ofast** и **-Og** дают нестабильные результаты, в независимости от выбора значения N. Так, например, при выборе более мелкого N уровень -**Og** оказался самым быстрым, а при выборе большего N – самым медленным (не считая -**O0**).

В ходе тестирования никаких ошибок из-за уровней оптимизации не происходило, программа работала корректно.