МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Физтех-школа Радиотехники и компьютерных технологий

Лабораторная работа

Исследование зависимости времени выполнения программы от уровня оптимизации

Выполнил:
Рогов Анатолий
Б01-406

г. Долгопрудный
2025

Цель работы: 1) построение модели множества Мандельброта с различными способами вычисления; 2) измерение времени выполнения программы с каждым из способов и проанализировать исследуемую зависимость.

В работе используются: графическая библиотека SFMl 2.6.1, библиотека time, библиотека immintrin.

Во многих случаях скорость выполнения программы является очень важным фактором при её разработке. Если программа должна обрабатывать видеокадры или сетевые пакеты в масштабе реального времени, то медленно работающая программа не сможет обеспечить необходимую функциональность. Посмотрим, как можно ускорить работу программ с помощью нескольких приемов оптимизации.

Чтобы написать эффективную программу, необходимо, во-первых, подобрать наилучший набор алгоритмов и структур данных и, во-вторых, написать код, который компилятор сможет превратить в эффективный выполняемый код. Внешне незначительные изменения в исходном коде программы могут радикально повлиять на успех ее оптимизации компилятором. В-третьих (это особенно касается преимущественно вычислительных задач), необходимо разделить задачи на части, которые можно вычислять параллельно, используя преимущества многоядерных и многопроцессорных систем.

Первый шаг в оптимизации программы – устранение ненужной работы, чтобы код выполнял намеченную задачу с максимальной эффективностью. Под этим подразумевается устранение ненужных вызовов функций, условных проверок и обращений к памяти. Эти оптимизации не зависят от каких-либо конкретных свойств целевой машины.

Понимая, как работает процессор, мы сможем сделать второй шаг в оптимизации программы и использовать способность процессоров выполнять несколько инструкций параллельно.

В данной работе оптимизация кода по построению множества Мандельброта выглядит простым и линейным процессом применения серии преобразований в определенном порядке. Основные версии функции по построению множества:

1. Базовая версия - непосредственные арифметические преобразования для каждой отдельной точки;

2-3. Версия на массивах - представление точек в виде последовательностей из 4 элементов с применением арифметических операций для каждой из них.

4. Intrinsics версия - применение встроенных возможностей процессора для полной векторизации вычислений.

Множество Мандельброта. Множество Мандельброта — это фрактальное множество на комплексной плоскости, которое определяется простой итерационной формулой, но демонстрирует чрезвычайно сложную и красивую структуру. Оно было впервые описано Бенуа Мандельбротом в 1980 году и стало одним из самых известных примеров математических фракталов.

Множество Мандельброта строится на основе итераций комплексного квадратичного отображения: $z_{n+1} = z_n^2 + c$, где: z — комплексная переменная (начальное значение $z_0 = 0$), c — комплексная константа (параметр, определяющий точку на плоскости).

Правило принадлежности:
Точка c принадлежит множеству Мандельброта, если последовательность z_n не стремится к бесконечности при $n \to \infty$. На практике итерации выполняют до определённого предела (например, $n_{max} = 1000$), и если $|z_n|$ остаётся меньше некоторого порога (обычно 2), то c считается принадлежащей множеству.

Intrinsics functions. Интринсики (intrinsics) — это специальные функции или конструкции которые предоставляют низкоуровневый доступ к возможностям процессора, минуя стандартные механизмы компилятора. Они используются для оптимизации критически важных участков кода, где важны производительность и контроль над аппаратными ресурсами. Использование SIMD-инструкций (AVX, SSE, SIMD) позволяет выполнять одну операцию над несколькими данными одновременно (векторизация), например, обработка изображений и физические расчеты.

Ход работы Измерим приборную погрешность:

$$\bar{t} = 0,0057 \pm 0,0001 \text{ c } (1,8\%)$$

№	Время t, c
1	0,0061
2	0,0072
3	0,0080
4	0,0076
5	0,0059
6	0,0036
7	0,0064
8	0,0035
9	0,0049
10	0,0033

```c
#include <stdio.h>
#include <time.h>

int main(int argc, char* argv[]) {
    printf("PROGRAM TIME: %lg sec\n", (double)
       clock() / CLOCKS_PER_SEC);
    return 0;
}
```

Для проверки правильности применения соответствующих оптимизаций будем сравнивать результаты вычислений множества Мандельброта с помощью графической библиотеки. Будем следить за изменениями полученного изображения, чтобы исправить оптимизации, нарушившие ход выполнения программы. Все последующие измерения времени будем проводить без использовани графической библиотеки, рассчитывая каждую из точек по 300 раз. Окончательный результат поделим на 300 для получения времени на рассчет одной точки.

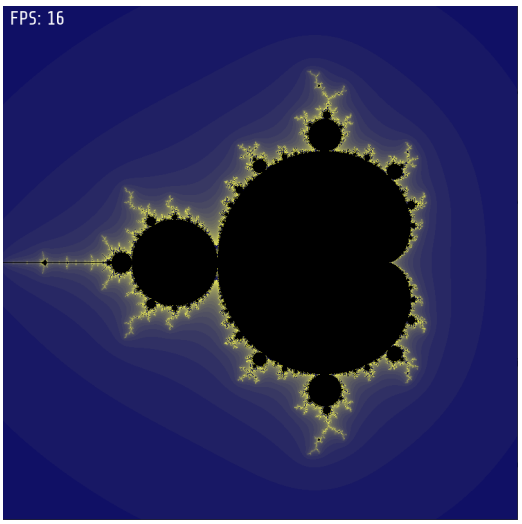


Рис. 1. Базовая версия

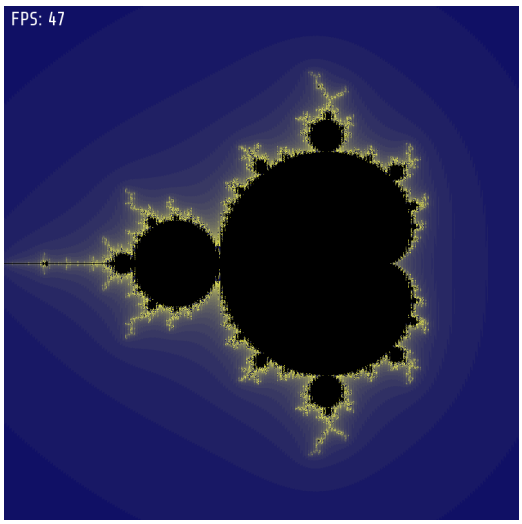


Рис. 2. Intrinsic версия

1 Базовая версия

Таблица 1: v1.1 Base -O3

№	Время t, с
1	18,81
2	18,74
3	18,80
4	18,80
5	18,72
6	18,81
7	18,75
8	18,80
9	18,82
10	18,73

$\bar{t} = 18,77 \pm 0,33$ с $(1,8\%)$
$F\bar{P}S = 16,86 \pm 0,16$ $(0,96\%)$

Таблица 2: v1.2 Base ~~-O3~~

№	Время t, с
1	39,79
2	39,78
3	39,88
4	39,74
5	40,02
6	39,87
7	39,58
8	39,58
9	39,59
10	39,64

$\bar{t} = 39,75 \pm 0,71$ с $(1,8\%)$
$F\bar{P}S = 8,46 \pm 0,02$ $(0,27\%)$

2 Версия на массивах

Таблица 3: v2.1 Vector -O3

№	Время t, с
1	10,04
2	10,48
3	10,14
4	10,49
5	10,15
6	10,53
7	10,05
8	10,05
9	10,51
10	10,14

$\bar{t} = 10,26 \pm 0,22$ с $(2,2\%)$
$F\bar{P}S = 29,12 \pm 0,94$ $(3,2\%)$

Таблица 4: v2.2 Vector ~~-O3~~

№	Время t, с
1	255,34
2	255,16
3	255,00
4	255,83
5	255,65

$\bar{t} = 254,99 \pm 4,52$ с $(1,8\%)$
$F\bar{P}S = 1,77 \pm 0,01$ $(0,56\%)$

3 Intrinsic версия

Таблица 5: v2.1 Vector -O3

№	Время t, с
1	6,23
2	6,15
3	6,18
4	6,18
5	6,23
6	6,17
7	6,20
8	6,19
9	6,19
10	6,18

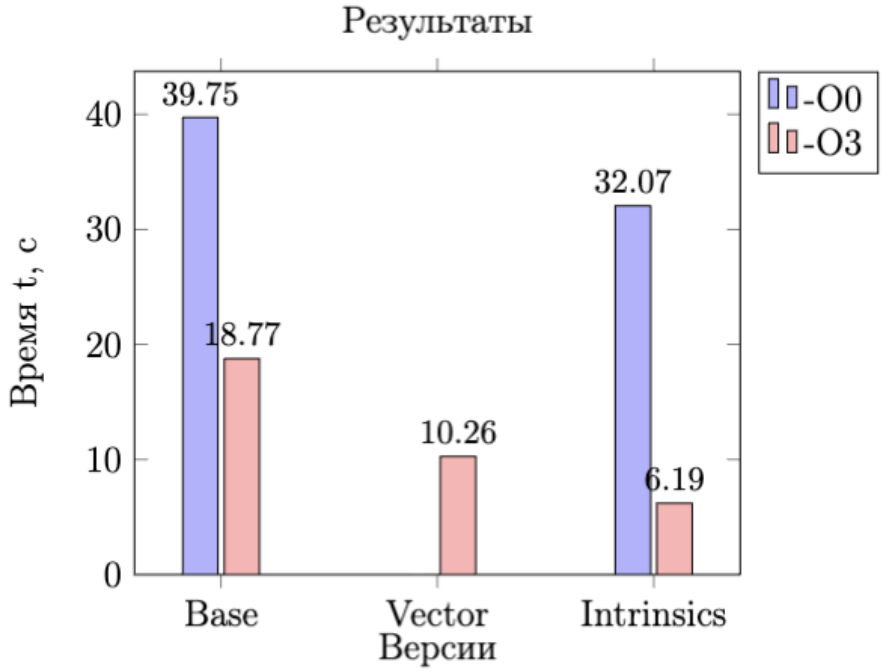
$\bar{t} = 6,19 \pm 0,11$ с $(1,8\%)$
$F\bar{P}S = 47,35 \pm 4,23$ $(8,9\%)$

Таблица 6: v2.2 Vector ~~-O3~~

№	Время t, с
1	32,12
2	31,83
3	32,17
4	32,22
5	32,16
6	32,16
7	31,94
8	32,18
9	32,11
10	31,88

$\bar{t} = 32,08 \pm 0,57$ с $(1,8\%)$
$F\bar{P}S = 10,17 \pm 0,01$ $(0,1\%)$

Анализ результатов В силу значительных отличий результатов измерения времени для vector версии опустим её значения на итоговой гистограмме:



Вывод: Я воссоздал математическую модель множества Мандельброта с помощью графической библиотеки SFML 2.6.1. Измерил время выполнения программы на каждом из уровней оптимизации. Итоговый прирост оптимизаций, проделанных в работе, составляет: $\frac{18.78}{6.19} \approx 3$. Полученный результат попадает в диапазон ожидаемых значений для оптимизаций без применения многопоточности!