###### 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**

«ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64»

студента 2 курса, группы 22201

Рабецкого Валерия Дмитриевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Матвеев А.А.

Новосибирск 2023

1. **Цель**

Изучить аспекты работы с языком ассемблера, ознакомиться с архитектурой x86-64 и провести детальный анализ ассемблерного кода программы первой лабораторной работы.

1. **Задание**

Изучить программную архитектуру x86-64. Ознакомиться с набором регистров процессора, основными арифметико-логическими командами, командами арифметического сопроцессора. Изучить способы адресации памяти, работу со стеком, так же понять принцип вызова подпрограмм, передачу параметров в подпрограммы и возврат результатов. Для программы на языке С сгенерировать ассемблерные листинги для архитектуры x86-64, после чего провести анализ ассемблерных листингов, сопоставляя команды языка С с машинными командами. После анализа процессорных инструкций продемонстрировать использование ключевых особенностей архитектуры x86-64 на конкретных участках кода, после чего сравнить различия в программах с оптимизациями и без.

**Листинг программы**

#include <iostream>

#include <cmath>

double PI(long long n)

{

double pi;

pi = 0;

double temp = 0;

for (int i = 0; i < n; i++)

{

temp = ((pow(-1, i)) / (2 \* i + 1));

pi += temp;

}

pi = 4 \* pi;

return pi;

}

int main()

{

long long n = 600000000;

double pi = 0;

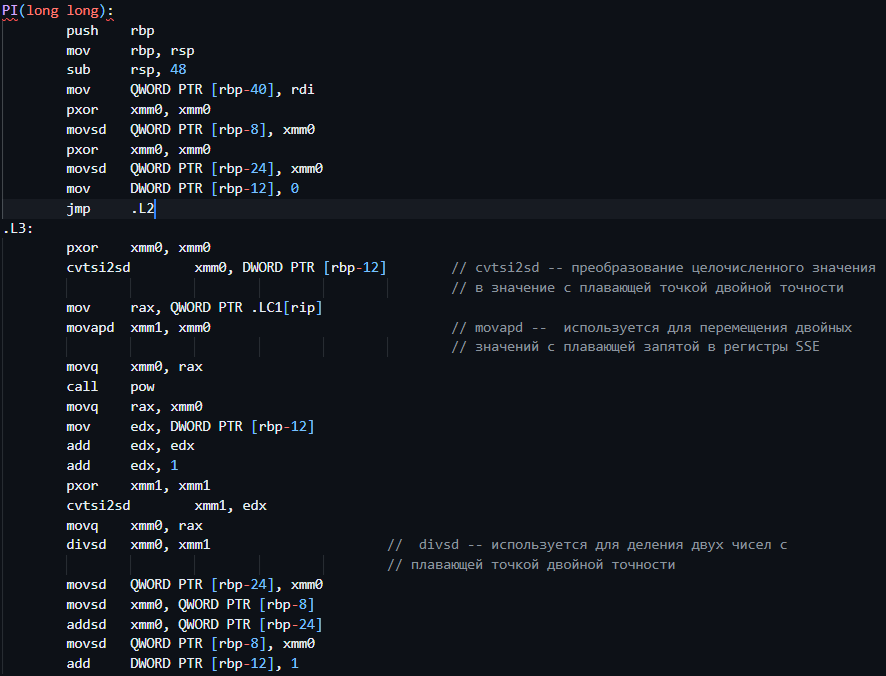
pi = PI(n);

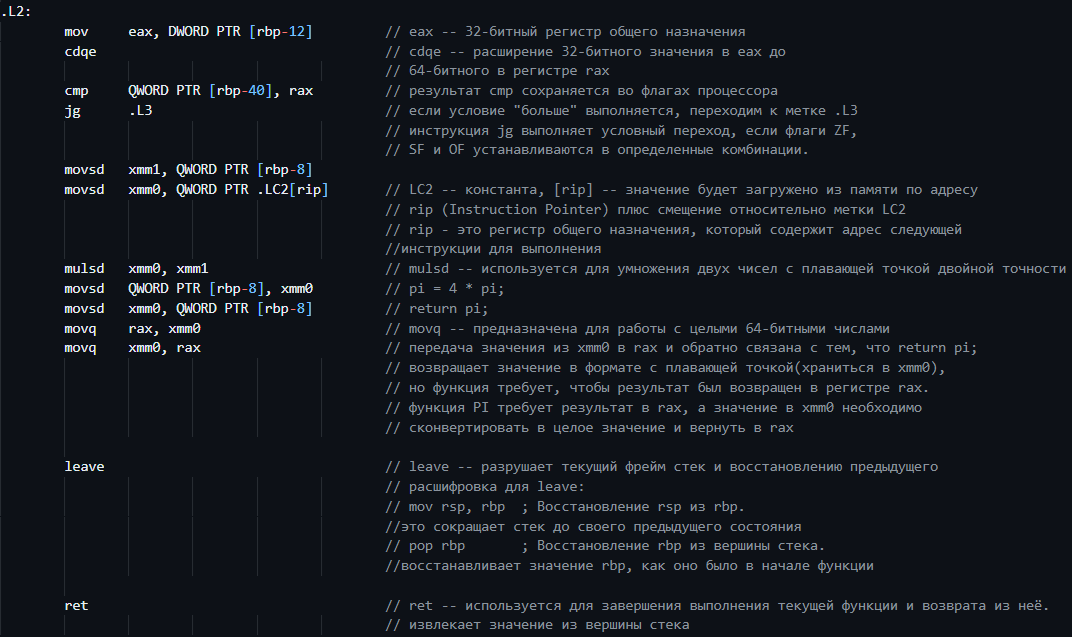
printf("pi = %lf", pi);

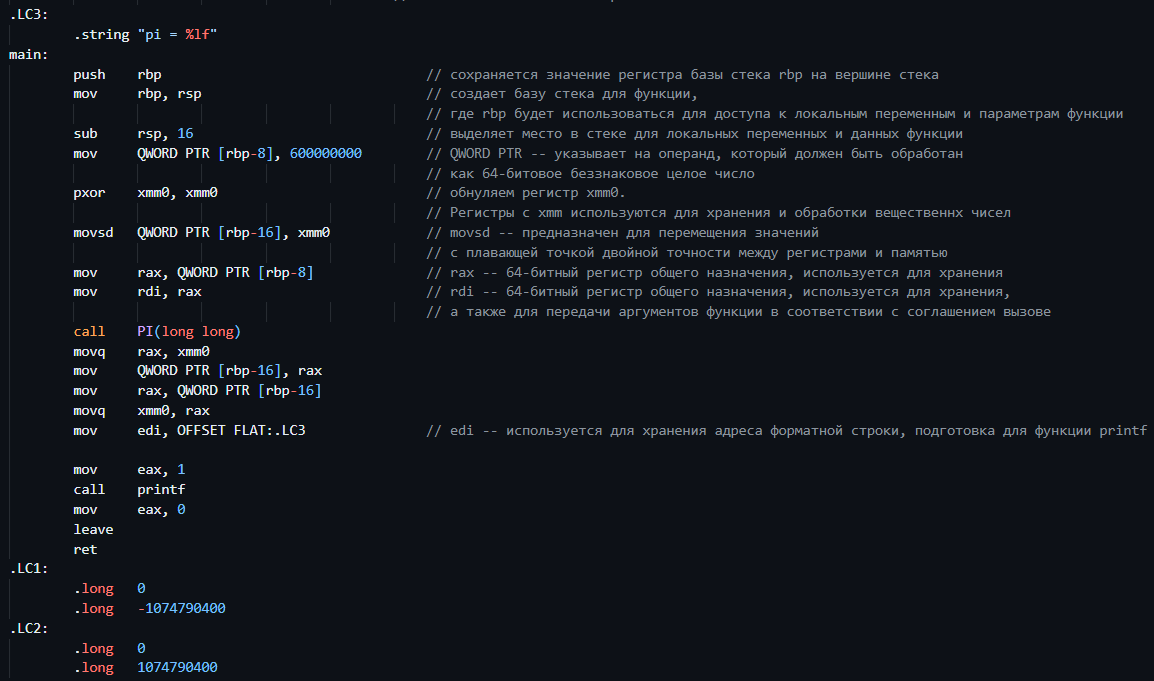
return 0;

}

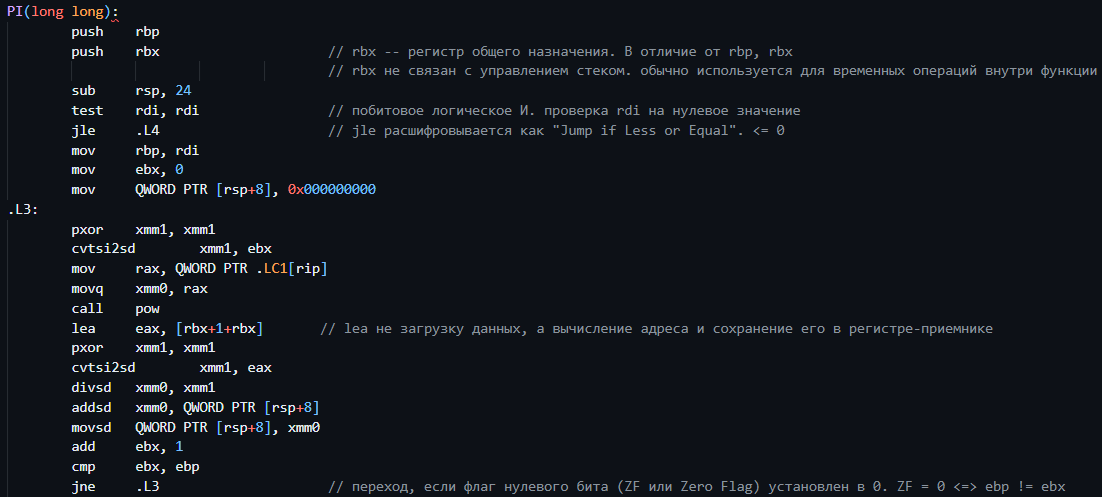
**Ассемблерный листинг для архитектуры x86-64**

****

****

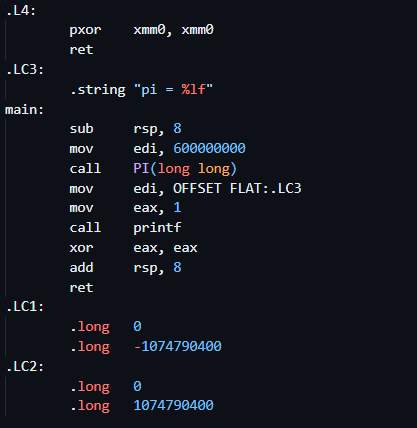
****

**Ассемблерный листинг для архитектуры x86-64 с оптимизацией O1**

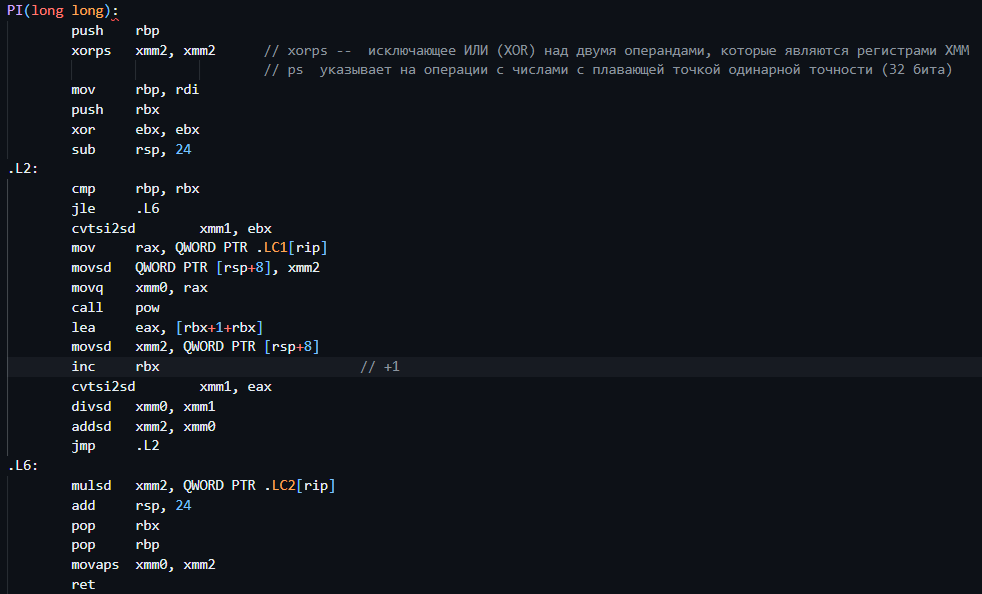
****

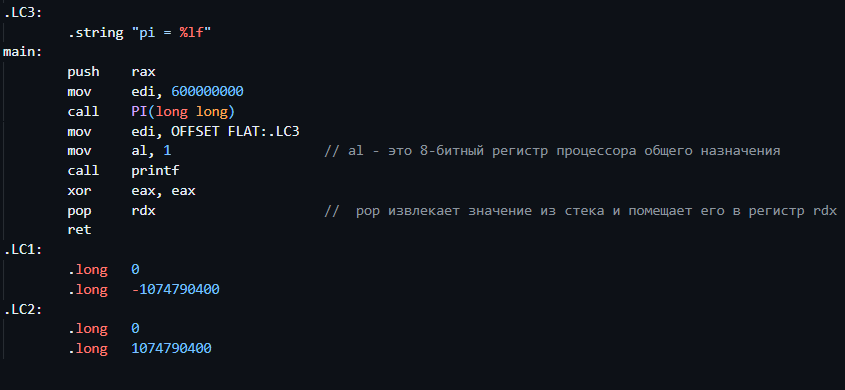
****

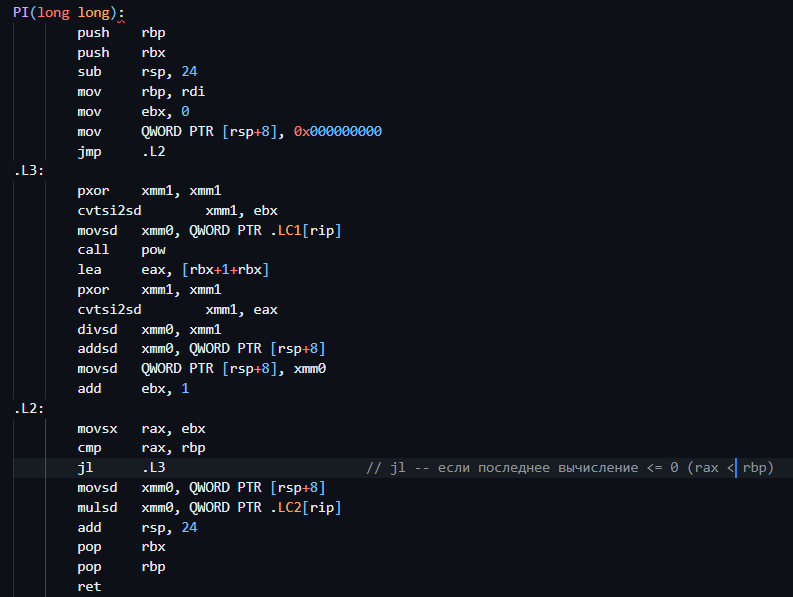
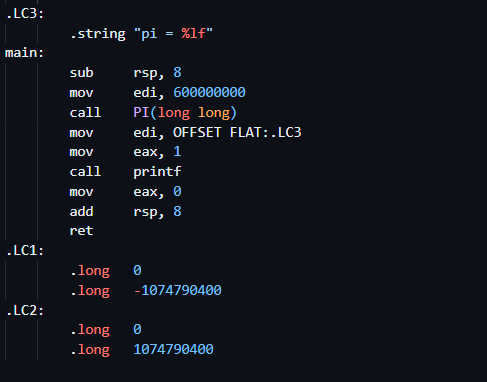
**Ассемблерный листинг для архитектуры x86-64 с оптимизацией O2, O3**

**** ****

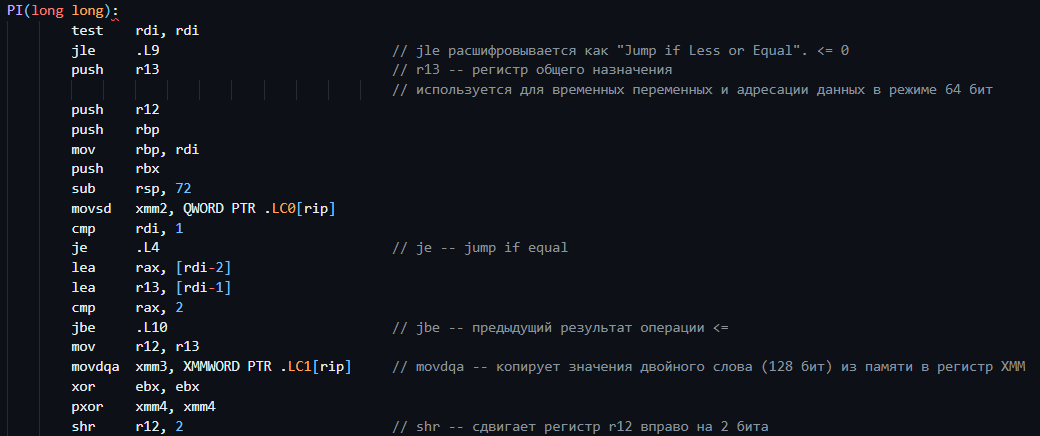
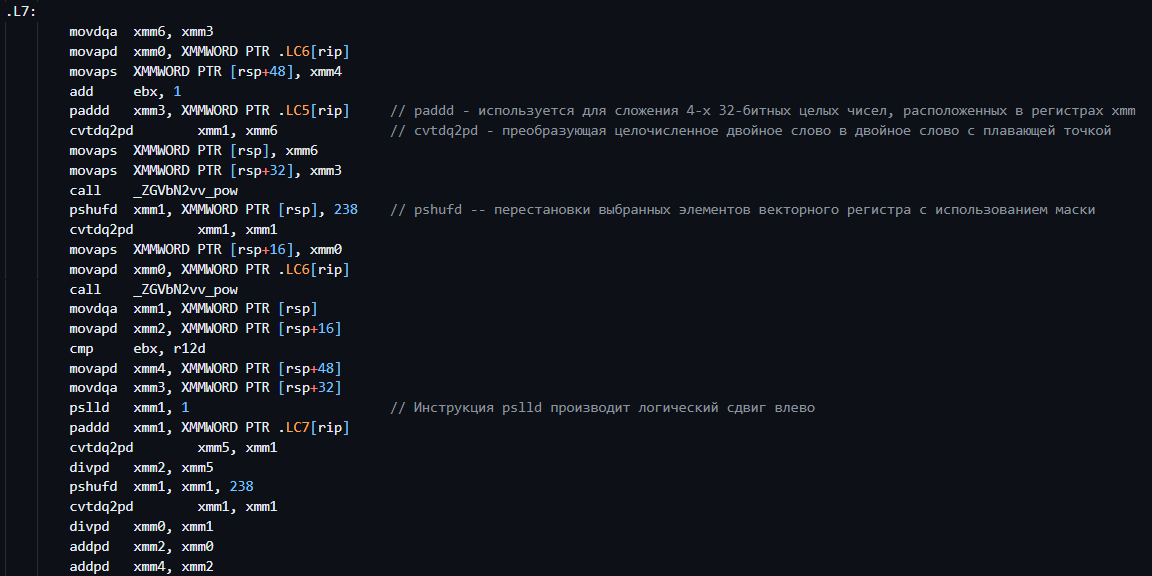
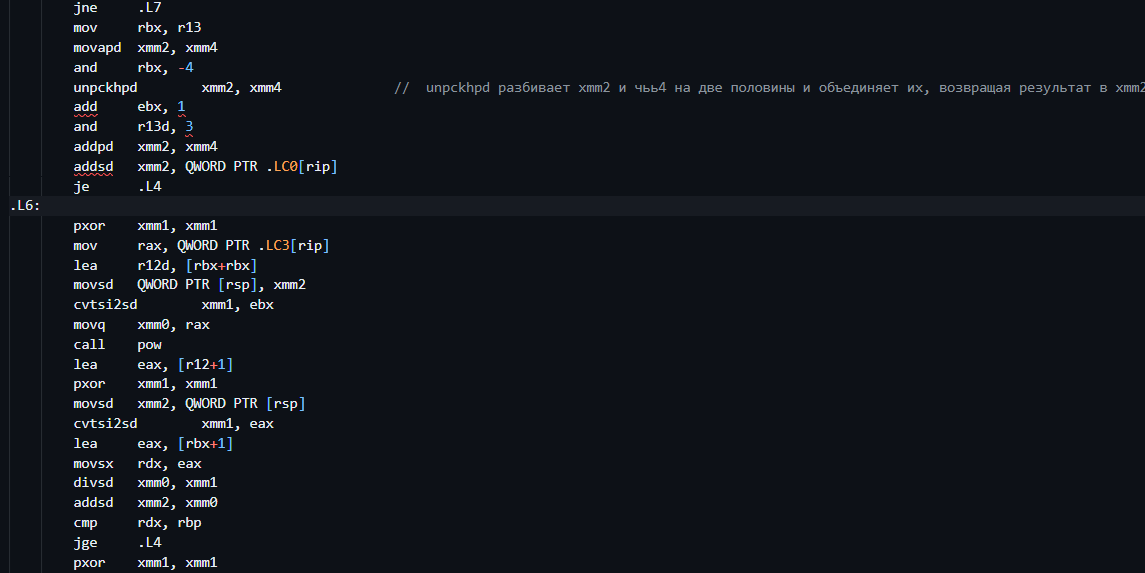
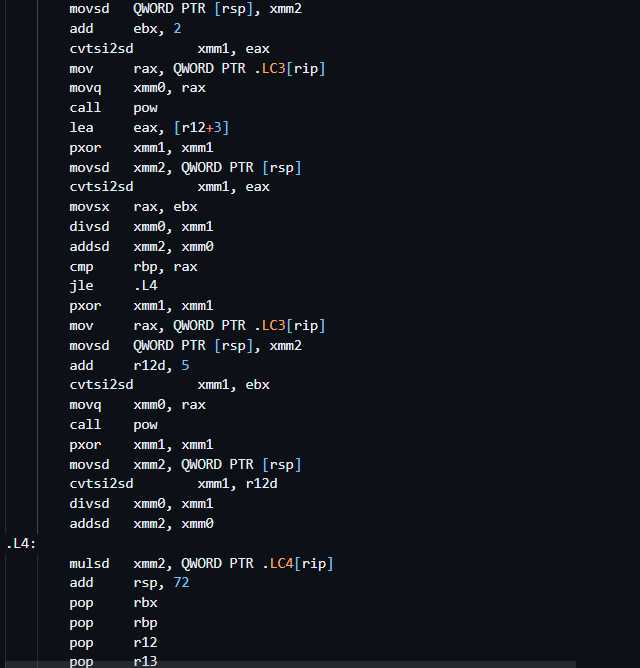
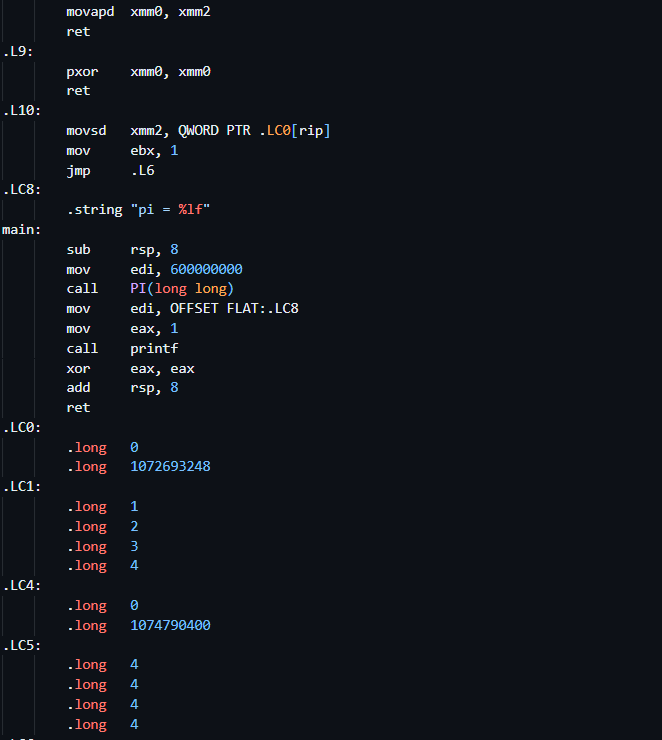
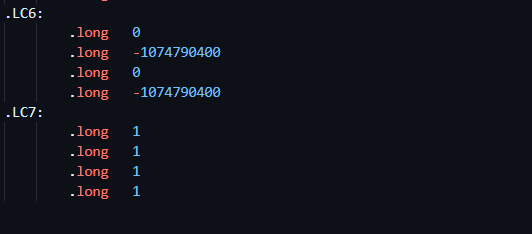
**Ассемблерный листинг для архитектуры x86-64 с оптимизацией Os**

****

****

**Ассемблерный листинг для архитектуры x86-64 с оптимизацией Og** **** 

**Ассемблерный листинг для архитектуры x86-64 с оптимизацией Ofast**

**** ****    

**Выводы**

По результатам проведённого анализа и сопоставления команд языка С и команд языка ассемблера было выяснено, что оптимизированном коде значительно уменьшено количество обращений в стек, больше используются регистры общего назначения и некоторые команды выполняются заранее. Код функции помещён в основное тело программы. Многие вычисления производятся с помощью специальных низкоуровневых алгоритмов работающих с битами.