###### 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 Бородина Артёма Максимовича 2 курса, 19205 группы

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

к.т.н, доцент

А.Ю. Власенко

Новосибирск 2020

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение *1.**Листинг 1* 6](#_Toc18443925)

Приложение *2.**Листинг 2* 7

Приложение *3.**Листинг 3* 8

# ЦЕЛЬ

1. Знакомство с программной архитектурой x86/x86-64.

2. Анализ ассемблерного листинга программы для архитектуры x86/x86-64.

# ЗАДАНИЕ

*1 вариант:*

Алгоритм вычисления числа π с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда.

1. Изучить программную архитектуру x86/x86-64

2. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.

3. Проанализировать полученные листинги.

4. Составить отчет по лабораторной работе.

**ОПИСАНИЕ РАБОТЫ**

1. Изучены основы архитектуры x86, группы регистров, доступные программисту, сопроцессор и векторные расширения.
2. Были сделаны 3 листинга: код программы, вычисляющей число π, код программы на ассемблере, скомпилированной с ключом -O0 и код программы на ассемблере, скомпилированной с ключом -O2.
3. Для просмотра кода программы на ассемблере был использован ресурс godbolt.org.
4. Компиляция обоих листингов на ассемблере проводилась с использованием ключа -m32, позволяющего компилировать код для 32битной архитектуры. Для просмотра кода с синтаксисом AT&T был использован соответствующий параметр. (листинги можно посмотреть по ссылке https://godbolt.org/z/8qss71)
5. При сравнении двух листингов на ассемблере становится заметно, что программа, скомпилированная с ключом -O2, требует гораздо меньше памяти для переменных и делает расчеты более эффективно (например, использование команды lea для расчета простых математических выражений), по сравнению с программой, скомпилированной с ключом -O0. Так же компилятор с ключом -O2 добавил проверку на 0 в функции **LeibnizFormula(int)**, что позволило избежать заход в цикл для расчета числа π, т.к. после инициализации переменных цикл пришлось бы сразу завершать.

**ЗАКЛЮЧЕНИЕ**

После изучения архитектуры x86, были рассмотрены программы на ассемблере, скомпилированные с ключами -O0 и -O2, и проведено последующее сравнение для поиска различий разных уровней оптимизации.

**Приложение 1.** 1ый листинг (код программы)



**Приложение 2.** 2ой листинг (ключ -O0)



**Приложение 3.** 3ий листинг (ключ -O2)

