###### 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, 23208 группы

**Веретенникова Андрея Александровича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Д.С. Иванишкин

Новосибирск 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение 1 (2, …). *Наименование приложения* 6](#_Toc18443925)

# ЦЕЛЬ

1. *Знакомство с программной архитектурой x86/x86-64.*
2. *Анализ ассемблерного листинга программы для архитектуры x86/x86-64.*

# ЗАДАНИЕ

*Вариант №1. Алгоритм вычисления числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда.*

# ОПИСАНИЕ РАБОТЫ

*Пошаговое описание выполненной работы:*

1. *Изучить программную архитектуру x86/x86-64:*
   1. *Набор регистров,*
   2. *Основные арифметико-логические команды,*
   3. *Способы адресации памяти,*
   4. *Способы передачи управления,*
   5. *Работу со стеком,*
   6. *Вызов подпрограмм,*
   7. *Передачу параметров в подпрограммы и возврат результатов,*
   8. *Работу с арифметическим сопроцессором,*
   9. *Работу с векторными расширениями.*
2. *Для программы из лабораторной работы №1 «ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ ПРОГРАММ» сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.*
3. *Проанализировать полученные листинги.*
4. *Сравнить различия в программах для архитектуры x86 и архитектуры x86-64.*

*Выбранная методика: сайт для генерации кода в ассемблерный листинг (Godbolt.org), используемый компилятор x86-64 gcc 14.2.*

# ХОД РАБОТЫ

Разбор листинга x86-64:

# 

# 

# 

# 

# 

# Разбор листинга x86:

# 

# 

# 

# 

# 

# ЗАКЛЮЧЕНИЕ

*В ходе лабораторной работы я познакомился с программной архитектурой x86/x86-64 и провел анализ ассемблерного листинга программы для этой архитектуры.*

# Приложение 1. *Листинг x86-64*

pi\_f:

pushq %rbp

movq %rsp, %rbp

movl %edi, -36(%rbp)

pxor %xmm0, %xmm0

movsd %xmm0, -8(%rbp)

movl $1, -12(%rbp)

movl $1, -16(%rbp)

movl $0, -20(%rbp)

jmp .L2

.L3:

pxor %xmm1, %xmm1

cvtsi2sdl -12(%rbp), %xmm1

pxor %xmm2, %xmm2

cvtsi2sdl -16(%rbp), %xmm2

movsd .LC1(%rip), %xmm0

divsd %xmm2, %xmm0

mulsd %xmm1, %xmm0

movsd -8(%rbp), %xmm1

addsd %xmm1, %xmm0

movsd %xmm0, -8(%rbp)

addl $2, -16(%rbp)

negl -12(%rbp)

addl $1, -20(%rbp)

.L2:

movl -20(%rbp), %eax

cmpl -36(%rbp), %eax

jl .L3

movsd -8(%rbp), %xmm0

popq %rbp

ret

.LC2:

.string "Input n: "

.LC3:

.string "%d"

.LC4:

.string "Answer: %0.14f\n"

main:

pushq %rbp

movq %rsp, %rbp

subq $32, %rsp

movl %edi, -20(%rbp)

movq %rsi, -32(%rbp)

movl $.LC2, %edi

movl $0, %eax

call printf

leaq -4(%rbp), %rax

movq %rax, %rsi

movl $.LC3, %edi

movl $0, %eax

call \_\_isoc99\_scanf

movl -4(%rbp), %eax

movl %eax, %edi

call pi\_f

movq %xmm0, %rax

movq %rax, %xmm0

movl $.LC4, %edi

movl $1, %eax

call printf

movl $0, %eax

leave

ret

.LC1:

.long 0

.long 1074790400

# Приложение 2. *Листинг x86*

pi\_f:

push ebp

mov ebp, esp

sub esp, 32

fldz

fstp QWORD PTR [ebp-8]

mov DWORD PTR [ebp-12], 1

mov DWORD PTR [ebp-16], 1

mov DWORD PTR [ebp-20], 0

jmp .L2

.L3:

fild DWORD PTR [ebp-12]

fild DWORD PTR [ebp-16]

fld QWORD PTR .LC1

fdivrp st(1), st

fmulp st(1), st

fld QWORD PTR [ebp-8]

faddp st(1), st

fstp QWORD PTR [ebp-8]

add DWORD PTR [ebp-16], 2

neg DWORD PTR [ebp-12]

add DWORD PTR [ebp-20], 1

.L2:

mov eax, DWORD PTR [ebp-20]

cmp eax, DWORD PTR [ebp+8]

jl .L3

fld QWORD PTR [ebp-8]

leave

ret

.LC3:

.string "Input n: "

.LC4:

.string "%d"

.LC5:

.string "Answer: %0.14f\n"

main:

lea ecx, [esp+4]

and esp, -16

push DWORD PTR [ecx-4]

push ebp

mov ebp, esp

push ecx

sub esp, 20

sub esp, 12

push OFFSET FLAT:.LC3

call printf

add esp, 16

sub esp, 8

lea eax, [ebp-12]

push eax

push OFFSET FLAT:.LC4

call \_\_isoc99\_scanf

add esp, 16

mov eax, DWORD PTR [ebp-12]

sub esp, 12

push eax

call pi\_f

add esp, 16

sub esp, 4

lea esp, [esp-8]

fstp QWORD PTR [esp]

push OFFSET FLAT:.LC5

call printf

add esp, 16

mov eax, 0

mov ecx, DWORD PTR [ebp-4]

leave

lea esp, [ecx-4]

ret

.LC1:

.long 0

.long 1074790400