**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 ARM» Студента 2 курса, 21202 группы

**Куращенко Льва Владиславовича**

## Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

## Перепелкин В.А.

Новосибирск 2022

# ЦЕЛЬ

* *Знакомство с программной архитектурой ARM;*
* *Анализ ассемблерного листинга программы для архитектуры ARM;*

# ЗАДАНИЕ

1. *Изучить основы программной архитектуры ARM.*
2. *Написать программу на языке C, которая реализует алгоритм вычисления числа π разложением в ряд*.
3. *Для программы сгенерировать ассемблерные листинги для архитектуры ARM, используя различные уровни комплексной оптимизации.*
4. *Проанализировать полученные листинги и сделать следующее:*
   * *Сопоставьте команды языка Си с машинными командами;*
   * *Определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти);*
   * *Описать и объяснить оптимизационные преобразования, выполненные компилятором;*
   * *Продемонстрировать использование ключевых особенностей архитектуры ARM на конкретных участках ассемблерного кода.*
5. *Составить отчет по лабораторной работе. Отчет должен содержать следующее:*
   * *Титульный лист;*
   * *Цель лабораторной работы;*
   * *Полный компилируемый листинг реализованной программы и команды для ее компиляции;*
   * *Листинг на ассемблере с описаниями назначения команд с точки зрения реализации алгоритма выбранного варианта;*
   * *Вывод по результатам лабораторной работы*

**Листинг программы на Си**

*#include <stdio.h>*

*int main(){*

*long long int N = 2100000000;*

*double pi = 0;*

*double delta = 0;*

*for (int i = 0; i < N; i++){*

*delta = (double)4/(double)(2\*i+1);*

*if (i % 2 == 1)*

*delta = -delta;*

*pi += delta;*

*}*

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

*return 0;*

*}*

**Листинг ассемблера ARM с ключом O0**

.LC0:

.ascii "%lf\000"

main:

push {r7, lr} //На стек два регистра

sub sp, sp, #32 //Выделить 32 байта

add r7, sp, #0 //Прибавить ноль, записать в r7

adr r1, .L6 //записать в r1 .L6 адрес N

ldrd r0, [r1] //Загрузить в r0 значение N

strd r0, [r7] //Сохранить r0 по адресу r7

mov r0, #0 //Записать ноль в r0

mov r1, #0 //Также обнуляем r1

strd r0, [r7, #24] //На 24 бита пишем ноль

mov r0, #0

mov r1, #0

strd r0, [r7, #16]

movs r1, #0

str r1, [r7, #12] //Записываем i

b .L2

.L4:

ldr r1, [r7, #12]

lsls r1, r1, #1

adds r1, r1, #1

vmov s15, r1 @ int

vcvt.f64.s32 d17, s15

vmov.f64 d18, #4.0e+0

vdiv.f64 d16, d18, d17

vstr.64 d16, [r7, #16]

ldr r1, [r7, #12]

cmp r1, #0

and r1, r1, #1

it lt

rsblt r1, r1, #0

cmp r1, #1

bne .L3

vldr.64 d16, [r7, #16]

vneg.f64 d16, d16

vstr.64 d16, [r7, #16]

.L3:

vldr.64 d17, [r7, #24]

vldr.64 d16, [r7, #16]

vadd.f64 d16, d17, d16

vstr.64 d16, [r7, #24]

ldr r1, [r7, #12]

adds r1, r1, #1

str r1, [r7, #12]

.L2:

ldr r1, [r7, #12]

asrs r0, r1, #31

mov r2, r1

mov r3, r0

ldrd r0, [r7]

cmp r2, r0

sbcs r1, r3, r1

blt .L4

ldrd r2, [r7, #24]

movw r0, #:lower16:.LC0

movt r0, #:upper16:.LC0

bl printf

movs r3, #0

mov r0, r3

adds r7, r7, #32

mov sp, r7

pop {r7, pc}

.L6:

.word 2100000000

.word 0

**Листинг ассемблера ARM с ключом Ofast**

LC0:

.ascii "%lf\000"

main:

vmov.i64 d17, #0 @ float

mov r1, #29952 //Записать в r1

movt r1, 32043 //Записать в r1 32043 в верхнюю половину

push {r3, lr} //Запушить r3, lr

movs r2, #1

movs r3, #0

vmov.f64 d19, #4.0e+0

.L3:

vmov s15, r2 @ int

tst r3, #1 //Сравнить с единицей кинуть флаг

add r3, r3, #1

add r2, r2, #2

vcvt.f64.s32 d18, s15 //Переделать в float 64

vdiv.f64 d16, d19, d18 //Делим четыре на число большое, пишем в d16

it ne //Ставим на следующую команду флаг not equal

vnegne.f64 d16, d16 //delta равно -delta

cmp r3, r1 // сравнить r3 и r1

vadd.f64 d17, d17, d16

bne .L3

vmov r2, r3, d17

movw r0, #:lower16:.LC0

movt r0, #:upper16:.LC0

bl printf

movs r0, #0

pop {r3, pc}