###### 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

«ВЛИЯНИЕ КЭШ-ПАМЯТИ НА ВРЕМЯ ОБРАБОТКИ МАССИВОВ»

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. *Исследование зависимости времени доступа к данным в памяти от их объема.*
2. *Исследование зависимости времени доступа к данным в памяти от порядка их обхода.*

# ОПИСАНИЕ РАБОТЫ

*Описание выполненной работы:*

1. *Написать программу, многократно выполняющую обход массива заданного размера тремя способами.*
2. *Для каждого массива и способа обхода измерить среднее время доступа к одному элементу (в тактах процессора). Построить графики зависимости среднего времени доступа от размера массива.*

*Результаты:*

# 

График зависимости:

# ЗАКЛЮЧЕНИЕ

*В ходе лабораторной работы, я исследовал зависимость времени доступа к данным в памяти от их объема, а также от порядка их обхода.*

# Приложение 1. *Листинг*

#include <iostream>

#include <vector>

#include <random>

#include <algorithm>

#include <chrono>

#include <x86intrin.h> // для \_\_rdtsc

using namespace std;

const int MIN\_CACHE\_INT = 256; // 256 int = 1 KB

const int MAX\_CACHE\_INT = 32 \* 1024 \* 1024 / sizeof(int); // 32 MB

void initialize\_array(vector<int>& arr, const string& mode) {

int n = arr.size();

if (mode == "direct") {

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

arr[i] = (i + 1) % n;

}

}

else if (mode == "reverse") {

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

arr[i] = (i - 1 + n) % n;

}

}

else if (mode == "random") {

vector<int> indices(n);

iota(indices.begin(), indices.end(), 0);

random\_device rd;

mt19937 gen(rd());

shuffle(indices.begin(), indices.end(), gen);

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

arr[indices[i]] = indices[(i + 1) % n];

}

}

}

uint64\_t measure\_access\_time\_cycles(const vector<int>& arr, int iterations) {

uint64\_t n = arr.size();

volatile int k = 0;

// "Прогрев" кэша

for (int iter = 0; iter < iterations; iter++) {

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

k = arr[k];

}

}

uint64\_t start = \_\_rdtsc();

for (int iter = 0; iter < iterations; iter++) {

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

k = arr[k];

}

}

uint64\_t end = \_\_rdtsc();

return (end - start) / (n \* iterations);

}

int main(int argc, char\* argv[]) {

if (argc != 2) {

cerr << "Too few arguments!" << endl;

}

int iterations = stoi(string(argv[1]));

int i = 0;

for (int size = 256; size <= MAX\_CACHE\_INT; size \*= 2) {

i++;

vector<int> arr(size);

cout << size / 256 << ": ";

// Прямой обход

initialize\_array(arr, "direct");

cout << measure\_access\_time\_cycles(arr, iterations) << ' ';

// Обратный обход

initialize\_array(arr, "reverse");

cout << measure\_access\_time\_cycles(arr, iterations) << ' ';

// Случайный обход

initialize\_array(arr, "random");

cout << measure\_access\_time\_cycles(arr, iterations) << ' ';

cout << endl;

}

return 0;

}