###### 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, 22202 группы

**Бальчинова А.С.**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

В.А. Перепёлкин

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ](#_Toc18443923) 3-6

[ЗАКЛЮЧЕНИЕ 7](#_Toc18443924)

# ЦЕЛИ

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.

# ЗАДАНИЕ

**1.** Написать программу, многократно выполняющую обход массива заданного размера тремя способами.

**2.** Для каждого размера массива и способа обхода измерить среднее время доступа к одному элементу (в тактах процессора). Построить графики зависимости среднего времени доступа от размера массива.

**3.** На основе анализа полученных графиков:

• определить размеры кэш-памяти различных уровней, обосновать ответ, сопоставить результат с известными реальными значениями;

• определить размеры массива, при которых время доступа к элементу массива при случайном обходе больше, чем при прямом или обратном; объяснить причины этой разницы во временах.

# ОПИСАНИЕ РАБОТЫ

1. Реализация программы на языке программирования C++.

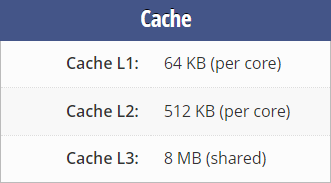
#include <iostream>  
#include <random>  
#include <fstream>  
#include <intrin.h>  
#include <limits.h>  
using namespace std;  
  
void directArrayFill(int \*array, const int len) {  
 for (int i = 0; i < len - 1; i++) {  
 array[i] = i + 1;  
 }  
 array[len - 1] = 0;  
}  
  
void reverseArrayFill(int \*array, const int len) {  
 for (int i = len - 1; i > 0; i--) {  
 array[i] = i - 1;  
 }  
 array[0] = len - 1;  
}  
  
void randomArrayFill(int \*array, const int len) {  
 random\_device dev;  
 mt19937 rng(dev());  
 uniform\_int\_distribution<mt19937::result\_type> dist(0,len - 1);  
 memset(array, -1, sizeof(int) \* len);  
 int tmp;  
 while ((tmp = dist(rng)) == 0) {  
 continue;  
 }  
 array[0] = tmp;  
 int next = tmp;  
 for (int i = 0; i < len - 2; i++) {  
 tmp = dist(rng);  
 for (int j = 0; j < len; j++) {  
 const int ind = (tmp + j) % len;  
 if (array[ind] == -1 && next != ind) {  
 array[next] = ind;  
 tmp = ind;  
 break;  
 }  
 }  
 next = tmp;  
 }  
 array[next] = 0;  
}  
  
void arrayTraversal(const int \*array, const int len, const int numberOfTraversals) {  
 for (int k = 0, i = 0; i < len \* numberOfTraversals; i++) {  
 k = array[k];  
 }  
}  
  
int main() {  
 const int nMin = 256; //2^8 - 1кб  
 const int nMax = 256 \* 1024 \* 32; //2^23 - 32мб  
 ofstream file("data.txt");  
 int \*array = new int[nMax];  
 const int k = 100;  
 for (int n = nMin; n <= nMax; n \*= 2) {  
 unsigned long long start = 0, end = 0;  
  
 directArrayFill(array, n);  
 unsigned long long t = LONG\_LONG\_MAX;  
 for (int j = 0; j < 1; j++) {  
 arrayTraversal(array, n, 1); // прогрев кэша  
 start = \_\_rdtsc();  
 arrayTraversal(array, n, k);  
 end = \_\_rdtsc();  
 t = min(t, end - start);  
 }  
 file << "N = " << n/256 << " KB. Direct traversal: " << (t)/(k \* n) << " CPU cycles\n";  
  
 reverseArrayFill(array, n);  
 t = LONG\_LONG\_MAX;  
 for (int j = 0; j < 1; j++) {  
 arrayTraversal(array, n, 1); // прогрев кэша  
 start = \_\_rdtsc();  
 arrayTraversal(array, n, k);  
 end = \_\_rdtsc();  
 t = min(t, end - start);  
 }  
 file << "N = " << n/256 << " KB. Reverse traversal: " << (t)/(k \* n) << " CPU cycles\n";  
  
 randomArrayFill(array, n);  
 t = LONG\_LONG\_MAX;  
 for (int j = 0; j < 1; j++) {  
 arrayTraversal(array, n, 1); // прогрев кэша  
 start = \_\_rdtsc();  
 arrayTraversal(array, n, k);  
 end = \_\_rdtsc();  
 t = min(t, end - start);  
 }  
 file << "N = " << n/256 << " KB. Random traversal: " << (t)/(k \* n) << " CPU cycles\n";  
 }  
 delete[] array;  
}

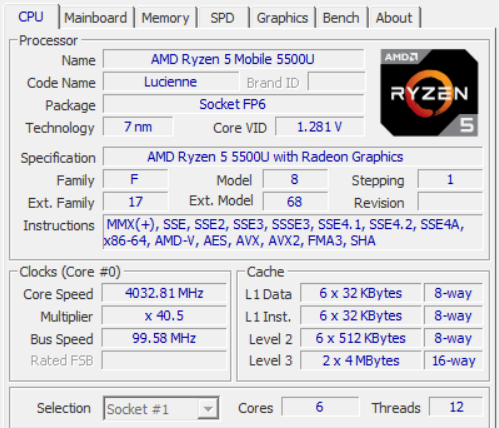
1. Запуск программы и результат.

Заметные скачки времени чтения: от 2МБ к 4МБ и от 4МБ к 8КБ.

Исходя из полученных значений, сразу можно сделать вывод, что размер кэша L3 составляет не менее 4 Мб, размер кэша L2 составляет от 2 до 4 мб.

Также можно сделать вывод, что размер L1 кэша не менее 32 Кб, так как время чтения элемента массива для массивов размера менее 64 Кб одинаковы не зависимо от способа обхода.  
При размере массива в 64 Кб время чтения для случайного обхода начинает расти. Для 128 Кб время чтения больше лишь на 1 такт, потому можно предположить, что размер L1 кэша не менее 128 Кб, то же самое для 256 Кб.  
Заметно увеличение времени чтения для 512 Кб, потому можно сделать вывод, что размер кэша L1 менее 512 КБ.  
В итоге имеем: L1 ∈ [192КБ;512КБ), L2 ∈ [2МБ; 4МБ), L3 ∈ [4МБ, 16МБ).





# ЗАКЛЮЧЕНИЕ

Исследовали зависимости времени доступа к данным в памяти от их объема, а также от порядка их обхода.