###### 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

«Измерение степени ассоциативности кэш-памяти»

студента 2 курса, 18205 группы

Гайдамака Андрея Владиславовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А. Ю. Власенко

Новосибирск 2019

ЦЕЛЬ 3

ЗАДАНИЕ 3

ОПИСАНИЕ РАБОТЫ 3

ЗАКЛЮЧЕНИЕ 4

Приложение 1. Код программы 5

# ЦЕЛЬ

Экспериментальное определение степени ассоциативности кэш-памяти.

# ЗАДАНИЕ

1. Написать программу, выполняющую обход памяти в соответствии с заданием.
2. Измерить среднее время доступа к одному элементу массива (в тактах процессора) для разного числа фрагментов: от 1 до 32. Построить график зависимости времени от числа фрагментов.
3. По полученному графику определить степень ассоциативности кэш- памяти, сравнить с реальными характеристиками исследуемого процессора.

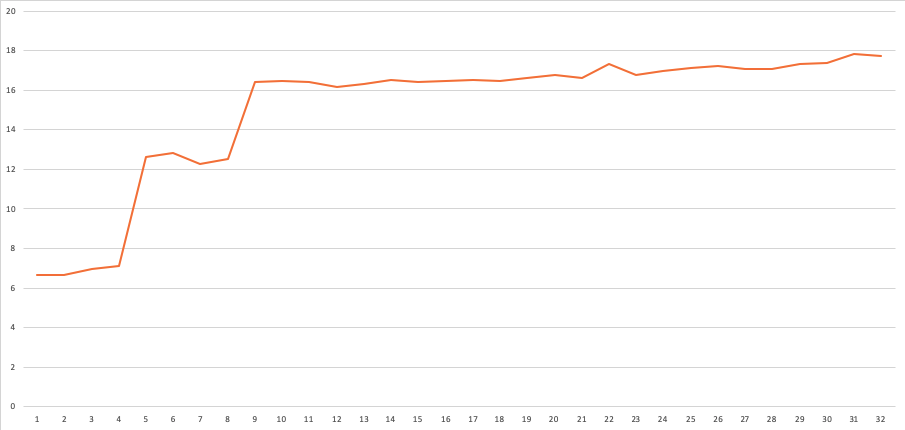
# ОПИСАНИЕ РАБОТЫ

В ходе лабораторной работы было произведено заполнение массива чтобы в соответствии с условиями лабораторной. Был произведен замер времени доступа к элементу массива при разных размерах массива которые зависят от количества фрагментов. Замеры проводились для разного количества фрагментов (от 1 до 32) чтобы определить степень ассоциативности кэш памяти. Все полученные значения записывались в csv файл и по этим данным был построен график зависимости количества тактов для доступа к элементу, от количества фрагментов.

Ассоциативность кэш-памяти 1-го уровня = 4

Ассоциативность кэш-памяти 2-го уровня = 4

Ассоциативность кэш памяти 3-го уровня = 16



# 

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была определена степень ассоциативности кэш-памяти разных уровней. Алгоритм обращался к элементам массива так чтобы воспроизвести эффект буксирования кэш-памяти. Ассоциативность памяти это компромисс, с одной стороны чем она больше тем меньше пробуксовок, а с другой чем она меньше тем быстрее доступ к кэш памяти

**Приложение 1. Код программы**

#include <iostream>  
#include <fstream>  
  
using ull = unsigned long long;  
  
inline ull rdtsc(){  
 unsigned int hi, lo;  
 asm volatile ("rdtsc\n" : "=a" (lo), "=d" (hi));  
 return ((ull)hi << 32) | lo;  
}  
  
double count\_time(const int \* arr, int size){  
 ull min = -1;  
 for (int j = 0; j < 10; ++j){  
 ull start = rdtsc();  
 for (int i = 0, k = 0; i < size; ++i){  
 k = arr[k];  
 }  
 ull end = rdtsc();  
 if (min > (end - start)){  
 min = end - start;  
 }  
 }  
 return static\_cast<double>(min) / size;  
}  
  
int main() {  
 const size\_t CACHE\_SIZE = 12582912;  
 const int INT\_COUNT\_IN\_CACHE = CACHE\_SIZE / sizeof(int);  
 double result[32] = {0};  
 std::ofstream csv\_file("out.csv");  
 if (!csv\_file.is\_open()){  
 throw std::runtime\_error("Cant open csv file");  
 }  
 for (int fragments = 1; fragments <= 32; ++fragments){  
 const int elements = INT\_COUNT\_IN\_CACHE / fragments;  
 const int size = INT\_COUNT\_IN\_CACHE \* fragments;  
 int \* array = new int[size];  
  
 for (int i = 0; i < fragments; ++i){  
 for (int j = 0; j < elements; ++j){  
 array[i \* INT\_COUNT\_IN\_CACHE + j] = (i + 1) \* INT\_COUNT\_IN\_CACHE + j;  
 }  
 }  
  
 for (int i = 0; i < elements; ++i){  
 array[(fragments - 1) \* INT\_COUNT\_IN\_CACHE + i] = i + 1;  
 }  
 array[(fragments - 1) \* INT\_COUNT\_IN\_CACHE + (elements - 1)] = 0;  
  
 result[fragments - 1] = count\_time(array, size);  
  
 }  
 for (int i = 0; i < 32; ++i){  
 csv\_file << i + 1 << ";";  
 }  
 csv\_file << std::endl;  
 for (int i = 0; i < 32; ++i){  
 csv\_file << result[i] << ";";  
 }  
  
  
 return 0;  
}