###### 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»

студентки \_\_\_\_2\_\_\_\_\_ курса, \_\_22208\_\_ группы

**Жуйко Снежаны Евгеньевны**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кудинов А.Ю.

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc153716185)

[ЗАДАНИЕ 3](#_Toc153716186)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc153716187)

[Заключение 6](#_Toc153716188)

# 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.*

*4. Составить отчет по лабораторной работе.*

# ОПИСАНИЕ РАБОТЫ

Листинг реализованной программы:

#include <iostream>

#include <fstream>

#include <string>

#include <exception>

#include <cfloat>

#include <intrin.h>

int main() {

    unsigned int generalCacheSize = 256 \* 1024 / sizeof(unsigned int);

    unsigned int arraySize = 256 \* generalCacheSize / sizeof(unsigned int);

    unsigned int\* array = new unsigned int[arraySize];

    unsigned int fragmentsOffset = generalCacheSize;

    std::ofstream ofs("results.csv");

    for (int fragmentsCount = 1; fragmentsCount < 33; ++fragmentsCount) {

        unsigned int fragmentSize = generalCacheSize / fragmentsCount;

        for (unsigned int i = 0; i < fragmentSize; ++i) {

            unsigned int k = i;

            for (int j = 1; j < fragmentsCount; ++j) {

                array[k] = k + fragmentsOffset;

                k += fragmentsOffset;

                if (k > arraySize) {

                    throw std::exception();

                }

            }

            array[k] = (i + 1) % fragmentSize;

        }

        unsigned long long i;

        volatile unsigned int k;

        unsigned long long timeTaken;

        unsigned long long minTime = ULLONG\_MAX;

        for (int j = 0; j < 100; ++j) {

            unsigned long long start = \_\_rdtsc();

            for (k = 0, i = 0; i < arraySize; ++i) {

                k = array[k];

            }

            unsigned long long end = \_\_rdtsc();

            timeTaken = (end - start) / arraySize;

            if (timeTaken < minTime) {

                minTime = timeTaken;

            }

        }

        std::cout << fragmentsCount << ',' << minTime << '\n';

        ofs << fragmentsCount << ',' << minTime << '\n';

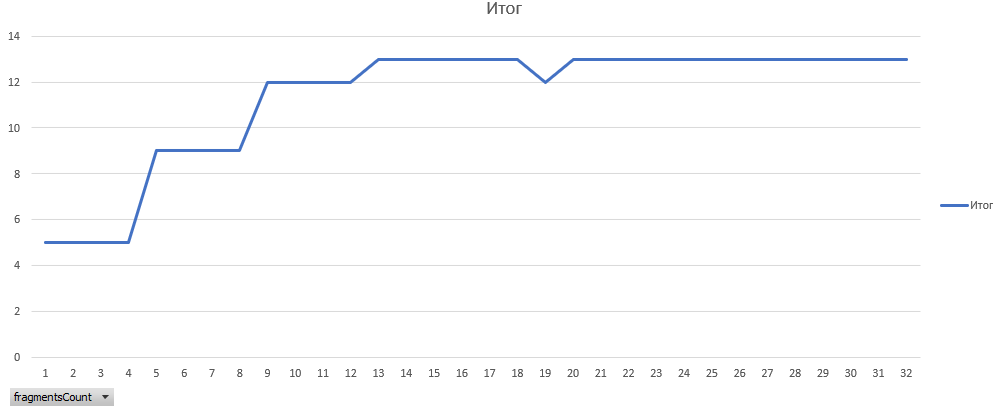
    }

    delete array;

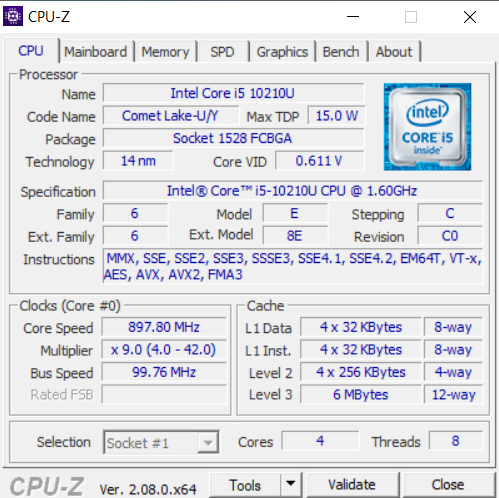
}

Строка компиляции: g++ lab9.cpp -o lab9 -O1

График зависимости среднего времени доступа к элементу массива от числа фрагментов:



Реальные значения степени ассоциативности кэш-памятей процессора:



Полученные в ходе тестирования значения степени ассоциативности кэш-памятей процессора:

L1 – 8-way, L2 – 4-way, L3 – 12-way.

Параметры теста: размер фрагментов - generalCacheSize / fragmentsCount, величина смещения – generalCacheSize, где generalCacheSize –  256 Kb, fragmentsCount **-** количество фрагментов от 1 до 32.

# Заключение

Для экспериментального определения степени ассоциативности кэш-памяти можно организовать обход данных в памяти, который вызывает

«буксование» кэш-памяти. Для этого нужно организовать доступ в

память по адресам, отображаемым на одно и то же множество в кэш-памяти.

Увеличивая количество конфликтующих обращений, в какой-то момент оно

превысит степень ассоциативности кэш-памяти и данные начнут

вытесняться из кэш-памяти – возникнет кэш-буксование. Буксование кэш-памяти приводит к увеличению времени выполнения программы, благодаря

чему можно определить степень ассоциативности кэш-памяти.

Обход фрагментов данных организуем таким образом, чтобы подряд происходили обращения к элементам разных фрагментов, отстоящим на заданное смещение. Так как размер кэш-памяти кратен размеру банка, то расстояние между началами фрагментов можно взять равным ему.

Эффект буксования кэша – это ситуация неэффективного использования кэш-памяти, когда блоки данных, к которым происходят обращения, отображаются на одно и то же множество кэш-строк.

Достоинства ассоциативного кэша: эффективное заполнение, недостатки: долгий поиск и большой тэг.

Достоинства кэша прямого отображения: быстрый поиск, небольшой тэг, недостатки – неэффективное заполнение.

Достоинства множественно-ассоциативного кэша: компромисс, недостаток: в сложности реализации.