**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 № 9**

«ИЗМЕРЕНИЕ СТЕПЕНИ АССОЦИАТИВНОСТИ КЭШ-ПАМЯТИ»

студента (ки) \_\_\_\_2\_\_\_\_\_ курса, \_\_21212\_\_ группы

**Олимпиева Юрия Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

**А. А. Артюхов**

2022 год

**Цели**

1. Измерение степени ассоциативности кэш-памяти.

**Задание**

1. Написать программу, выполняющую обход массива с шагом, кратным размеру кэш-памяти, при различном числе фрагментов.  
2. Измерить среднее время доступа к элементу массива в зависимости от числа фрагментов.

3. На основе анализа полученных данных определить ассоциативность кэш-памяти на основании скачков времени доступа к элементу в связи с буксованием кэш-памяти.  
4. Составить отчет по лабораторной работе.

**Описание работы**

* Написал на языке программирования Си++ программу, выполняющую обход массива с шагом, кратным размеру кэш-памяти, при различном числе фрагментов.
* Измерил с помощью функции \_\_rdtsc() время обхода массива в тактах процессора, определил средние значения.
* Построил графики зависимости времени обработки массива в зависимости от числа фрагментов.
* Составил отчёт о выполненной работе.

**Заключение**

В соответствии с экспериментальными результатами проделанной работы установили, что ассоциативность кэш-памяти первого и второго уровня равны 8, ассоциативность кэш-памяти третьего уровня равна 16.

Установленные данные ассоциативности совпали с фактическими показателями в документации процессора.

**Приложение 1**

/\* Copyright 2022 Olimpiev Y. \*/

**#include <cassert>**

**#include <cstdlib>**

**#include <iostream>**

**#include <x86intrin.h>**

**constexpr size\_t maxFragmentsAmount = 32;**

**constexpr size\_t Kb = 1024;**

**constexpr size\_t l1Size = 32 \* Kb;**

**constexpr size\_t l2Size = 512 \* Kb;**

**int main(void) {**

**size\_t cacheSize = l2Size;**

**size\_t step = cacheSize;**

**int\* a = new int[cacheSize \* maxFragmentsAmount];**

**for (int fragmentsAmount = 1; fragmentsAmount <= maxFragmentsAmount; fragmentsAmount++) {**

**size\_t currSize = cacheSize \* fragmentsAmount;**

**for (size\_t i = 0; i < currSize; i++) {**

**a[i] = (1 + i + cacheSize) % currSize;**

**}**

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

**for(int k = 0, i = 0; i < currSize; i++) {**

**k = a[k];**

**}**

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

**//out << (end - start) / currSize << ", ";**

**std::cout << "fragments: " << fragmentsAmount << " " << "\ttics: " << (end - start) / currSize << std::endl;**

**}**

**delete a;**

**return EXIT\_SUCCESS;**

**}**

**Build on Linux Ubuntu:**

g++ -O0 main.cpp -o lab9

**Приложение 2**

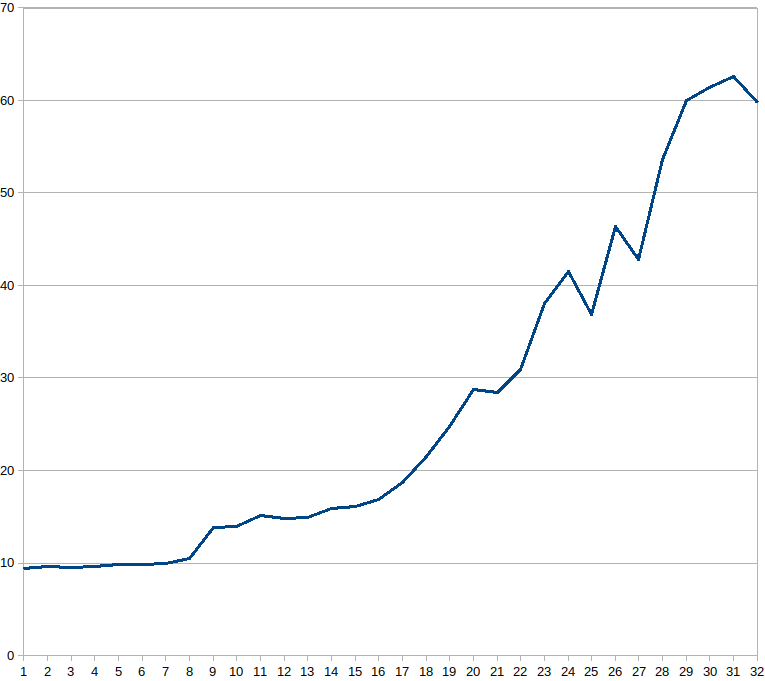
****

График зависимости среднего времени доступа к элементу массива (в тактах процессора) от числа фрагментов.