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 ПРАКТИЧЕСКОЙ РАБОТЫ

«*Измерение степени ассоциативности кэш-памяти*»

студента 2 курса, группы 22206

***Тропина Никиты Васильевича***

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

доцент, к.т.н.

А.Ю.Власенко

Новосибирск 2023

# СОДЕРЖАНИЕ

[ЦЕЛИ 3](#_Toc154042650)

[ЗАДАНИЕ 3](#_Toc154042651)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc154042652)

[ЗАКЛЮЧЕНИЕ 6](#_Toc154042653)

[ПРИЛОЖЕНИЕ 1. ЛИСТИНГ РЕАЛИЗОВАННОЙ ПРОГРАММЫ. 7](#_Toc154042654)

# ЦЕЛИ

1. Экспериментально определить степень ассоциативности кэш-памяти.

# ЗАДАНИЕ

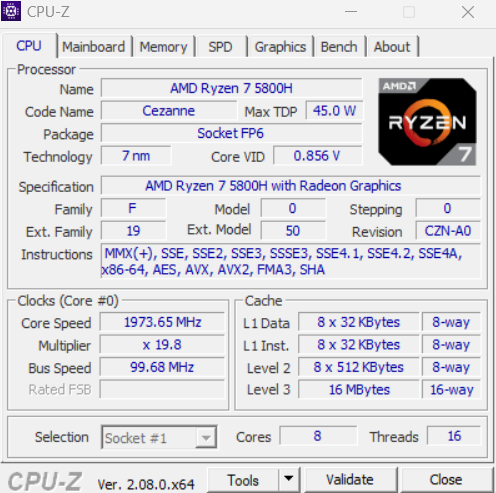
1. Написать программу, выполняющую обход памяти в соответствии с заданием.
2. Измерить среднее время доступа к одному элементу массива (в тактах процессора) для разного числа фрагментов: от 1 до 32. Построить график зависимости времени от числа фрагментов.
3. По полученному графику определить степень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

# ОПИСАНИЕ РАБОТЫ

Была написана программа, которая выделяется массив достаточно большого размера и заполняет его таким образом, чтобы его элементы образовывали связанный список, где значение каждого элемента есть индекс следующего так, чтобы расстояние между к подряд обращающимся элементам равнялось размеру кэш-памяти. Количество фрагментов с таким расстояние меняется от 1 до 32. После этого, измерялось время такого обхода массива. Результаты представлены на диаграмме.

На графике видно замедление после 8-ми фрагментов, что соответствует степени ассоциативности кэш-памяти 1-го и 2-го уровня, а также замедление после 16-ти фрагментов, что соответствует степени ассоциативности кэш-памяти 3-го уровня.

С помощью утилиты CPU-Z были получены настоящие данные о степени ассоциативности кэш-памяти исследуемого процессора.



# ЗАКЛЮЧЕНИЕ

В ходе работы был исследован процессор AMD Ryzen 7 5800H на степень ассоциативности кэш-памяти. Данные, полученные экспериментальным путем, соответствуют действительным значениям. Однако замедление после 16-ти фрагментов очень мало относительно замедления после 8-ми. Следовательно, нельзя сказать, что при настоящей пробуксовке памяти в L3 замедление будет незначительным.

# ПРИЛОЖЕНИЕ 1. ЛИСТИНГ РЕАЛИЗОВАННОЙ ПРОГРАММЫ.

#include <iostream>

#include <intrin.h>

#include <vector>

namespace {

const int offset = 1024 \* 1024 \* 4;

const int arraySize = 1024 \* 1024 \* 4;

}

int warmUp(int seconds)

{

int size = 1024;

std::vector<int> a(size);

std::vector<int> b(size);

for (int l = 0; l < seconds; l++) {

for (int k = 0; k < 64; k++) {

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

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

a[j] = i \* j + 1;

b[j] = a[i] \* j;

a[j] \*= b[j] / 2;

}

}

}

}

if (a[size / 2] == b[size / 2]) {

return 1;

}

else {

return 0;

}

}

void fillArray(int array[], int size, int ways)

{

int sizeOfWay = size / ways;

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

for (int i = 1; i <= ways; i++) {

array[(i - 1) \* offset + j] = offset \* i + j;

}

}

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

array[offset \* (ways - 1) + i] = (1 + i) % sizeOfWay;

}

}

std::vector<double> arrayTraversal(int array[], int size, int maxWays, int nRepetitions)

{

std::vector<double> times(maxWays);

for (int i = 1; i <= maxWays; i++) {

unsigned \_\_int64 curResult = 0;

unsigned \_\_int64 result = UINT64\_MAX;

fillArray(array, size, i);

int k = 0;

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

k = array[k];

}

curResult = \_\_rdtsc();

for (int r = 0; r < nRepetitions; r++) {

k = 0;

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

k = array[k];

}

}

result = \_\_rdtsc() - curResult;

times[i - 1] = result / size / nRepetitions;

}

return times;

}

int main()

{

int size = offset \* 32;

int realElementsNumber = arraySize;

int maxWays = 32;

int nRepetitions = 4;

int\* array = new int[size] {};

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

array[i] = 0;

}

std::cout << warmUp(2) << std::endl;

std::vector<double> times = arrayTraversal(array, realElementsNumber, maxWays, nRepetitions);

for (int i = 0; i < times.size(); i++) {

std::cout << i + 1 << " ways - " << times[i] << " " << std::endl;

}

delete[] array;

}