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 23208

Доценко Никиты Алексеевича

“Направление 09.03.01 – Информатика и вычислительная техника”

Преподаватель:
Иванишкин Д.С.

Новосибирск, 2024 г.

Содержание

1 Цель

1. Экспериментальное определение степени ассоциативности кэш-памяти.

2 Задание

1. 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.

2. По полученному графику определить степень ассоциативности кэшпамяти, сравнить с реальными характеристиками исследуемого процессора.

3 Ход работы

3.1 параметры теста

1. Размер между фрагментами - 8мб (размер L3 кэша)
2. Размер фрагментов - (объем кэш памяти / количество фрагментов)

3.2 График

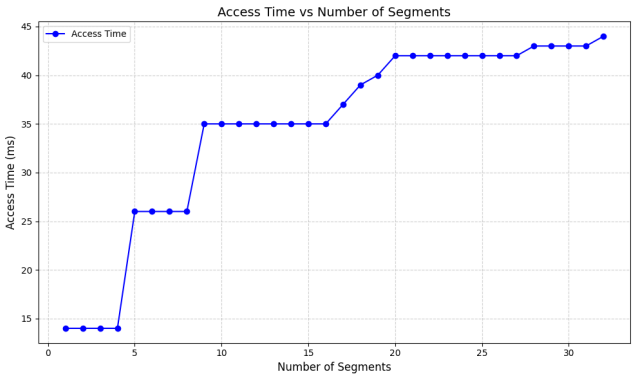

Рис. 1: График зависимости времени доступа от размера массива

3.3 Реальные и полученные в ходе тестирования значения степени ассоциативности кэш-памятей процессора

1. по графику отчётливо видно 3 скачка
   (a) первый скачок относится к TLB-кэшу (ассоциативность - 4)
   (b) второй скачок относится к L1 и L2 кэшу (ассоциативность - 8)
   (c) третий скачок относится к L3 кэшу (ассоциативность - 16)
2. реальные значения:
   (a) TLB - 4
   (b) L1 L2 - 8
   (c) L3 - 16

полное совпадение

3.4 Программа на C++

Ниже приведен листинг компилируемой программы main.cpp

```
#include <iostream>
#include <x86intrin.h>
#include <limits.h>
#include <iomanip>
#include <fstream>

const size_t L3_CACHE_SIZE = 8 * 1024 * 1024;
const size_t L3_ARRAY_SIZE = L3_CACHE_SIZE / sizeof(int);
```

```
using namespace std;

void initialize_array(int* array, int num_segments) {
    int segment_size = L3_ARRAY_SIZE / num_segments;
    int index;

    for (int segment = 0; segment < (num_segments - 1); segment++) {
        index = segment * L3_ARRAY_SIZE;
        for (int i = 0; i < segment_size; i++) {
            array[index + i] = index + i + L3_ARRAY_SIZE;
        }
    }
    index = L3_ARRAY_SIZE * (num_segments - 1);
    for (int i = 0; i < segment_size; i++) {
        array[index + i] = i + 1;
    }
    array[index + segment_size - 1] = 0;
}

size_t time(int* array, int total_size, int num_segments, int iterations) {
    size_t min_access_time = INT_MAX;
    int segment_size = L3_ARRAY_SIZE;

    int current_index = 0;

    for (int run = 0; run < iterations; run++) {

        for (int warmup_iter = 0; warmup_iter < iterations; warmup_iter++) {
            for (int i = 0; i < total_size; i++) {
                volatile int temp = array[i];
            }
        }

        size_t start_time = __rdtsc();
        for (int access_iter = 0; access_iter < iterations; access_iter++) {
            do {
                current_index = array[current_index];
            } while (current_index != 0);
        }
        size_t end_time = __rdtsc();

        size_t access_time = (end_time - start_time) / (segment_size *
            iterations);
        if (access_time < min_access_time) {
            min_access_time = access_time;
        }
    }
    return min_access_time;
}

int main() {
    int num_iterations;
    cout << "Enter␣number␣of␣iterations:␣";
    cin >> num_iterations;

    size_t total_array_size = L3_ARRAY_SIZE * 32;

    int* data_array = new int[total_array_size];
```

```
    ofstream csv_file("output.csv");
    csv_file << "num_segments,access_time" << endl;

    for (int num_segments = 1; num_segments <= 32; num_segments++) {
        cout << setw(2) << num_segments << " : ";
        initialize_array(data_array, num_segments);
        size_t access_time = time(data_array, total_array_size, num_segments
            , num_iterations);
        cout << access_time << endl;


        csv_file << num_segments << "," << access_time << endl;
    }


    csv_file.close();

    delete[] data_array;

    csv_file.close();
    return 0;

}
```

4 Вывод

В ходе выполнения работы были изучены принципы отображения оперативной памяти в кэш, понятие кэш-буксировки, кэш-банка. Были найдены степени ассоциативности кэшей различных уровней. Проведено сравнение полученых данных с реальными показателями