###### 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**

«ВЛИЯНИЕ КЭШ-ПАМЯТИ НА ВРЕМЯ ОБРАБОТКИ МАССИВОВ»

студента 2 курса, группы 23201

Смирнова Гордея Андреевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Матвеев Алексей Сергеевич

Новосибирск 2024

1. **Цель**

Изучить принцип работы работа кэш-память процессора и на примере обхода элементов массива в разном порядке выяснить размер разных уровней кэш-памяти на конкретном ПК.

1. **Задание**

На основе разных способов обхода циклического массива: прямого, обратного и случайного, сделать среднюю оценку количества тактов для обращения к одному элементу массива. На основе полученных данных выяснить приблизительный размер кэша процессора.

1. **Описание обходов**

Используются следующие обходы массива:

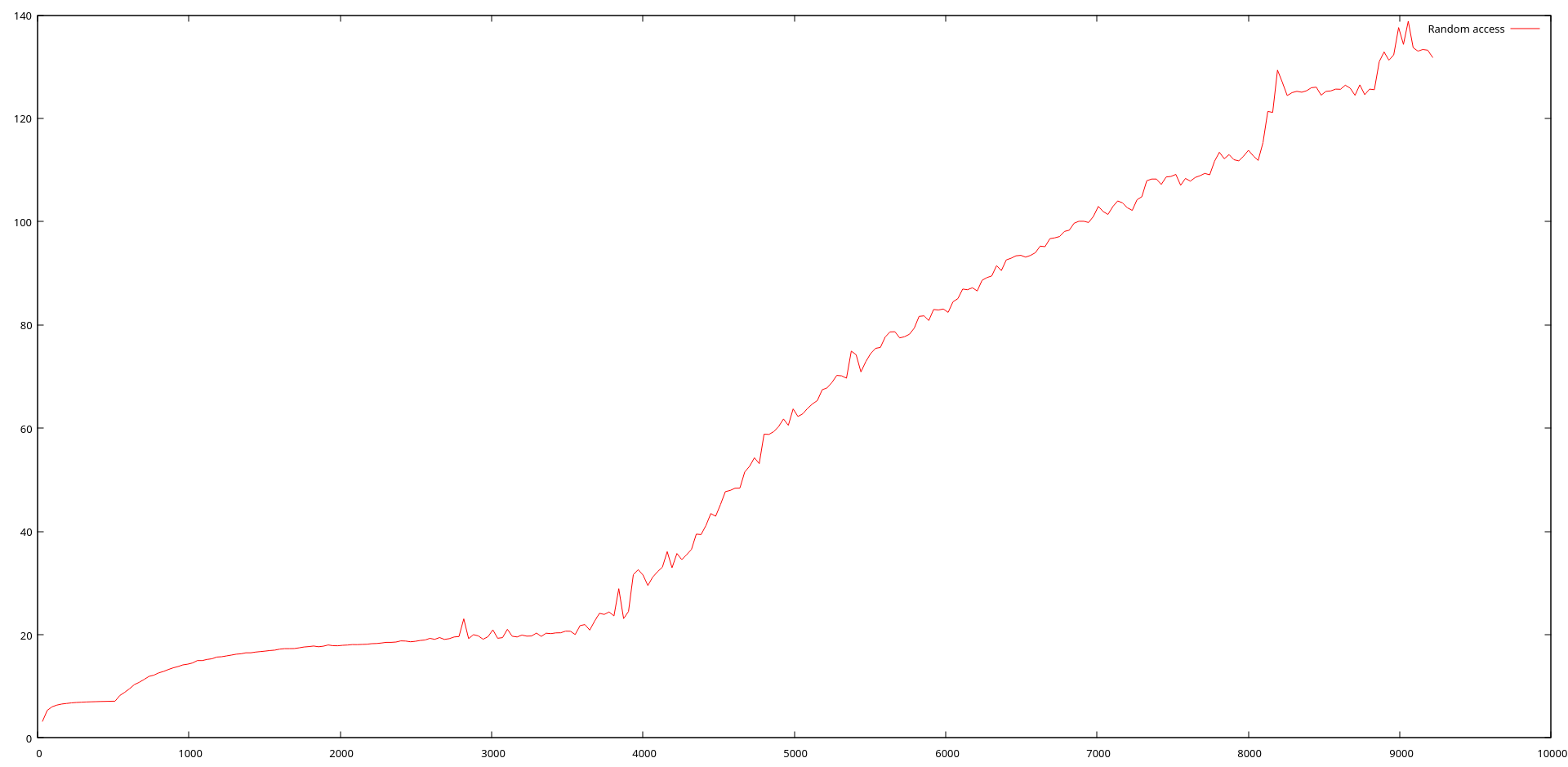
* Прямой;
* Обратный;
* Случайный.

**Прямой обход** подразумевает, что значением ячейки массива с индексом i будет i+1, т.е. индекс следующей соседней ячейки массива, в случае последнего элемента следующим элементом будет самый первый. Таким образом мы получаем следующий индекс и так обходим массив несколько раз.

**Обратный обход** – прямой обход, но с условием, что теперь обход идёт с конца массива линейным порядком.

**Случайный обход** подразумевает, что значением ячейки массива с индексом i будет некоторый индекс, который невозможно предугадать, но гарантируется, что индекс не выйдет за пределы массива. Тем самым, ходя по разным индексам, мы полностью обходим массив.

1. **График роста тактов обращения**



1. **Листинг программы**

#include <chrono>

#include <random>

#include <climits>

#include <x86intrin.h>

constexpr int minSize = 4 \* 1024 / sizeof(int);

constexpr int maxSize = 2 \* 1024 \* 1024 / sizeof(int);

constexpr int loops = 16;

void traverseAndPrint(const int\* data, const int size) {

volatile int k;

unsigned long long i, minTicks = ULLONG\_MAX;

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

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

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

k = data[k];

}

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

minTicks = std::min(minTicks, end - start);

}

printf("%llu %0.2lf\n",

size \* sizeof(int) / 1024, static\_cast<double>(minTicks) / size);

}

void shuffleAndTest(int\* data, const int size) {

std::random\_device rd;

std::mt19937 gen(rd());

// фишер-йетс

auto randomPermutation = new int[size];

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

randomPermutation[i] = i;

}

for (int i = size - 1; i > 0; --i) {

std::uniform\_int\_distribution dis(0, i);

int j = dis(gen);

std::swap(randomPermutation[i], randomPermutation[j]);

}

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

data[randomPermutation[i]] = randomPermutation[(i + 1) % size];

}

traverseAndPrint(data, size);

delete[] randomPermutation;

}

int main() {

FILE\* file = freopen("output.txt", "w", stdout);

std::vector cacheWarming(maxSize, 0.1);

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

for (size\_t j = 0; j < minSize; ++j) {

cacheWarming[i] = std::sqrt(cacheWarming[i] \* cacheWarming[i] + cacheWarming[j] \* cacheWarming[j]);

}

}

auto data = new int[maxSize];

for (int size = minSize; size <= maxSize; size += minSize) {

shuffleAndTest(data, size);

}

delete[] data;

fclose(file);

return 0;

}

1. **Оценка размера кэша**

По графику можно заметить, что прирост тактов начинается с 32kB, что явно говорит о том, что размер L1 кэша равен 32kB.

Следующее возрастание в середине между 0 и 1000Kb, что соответствует размеру L2 кэша - 512Kb.

Ситуация с L3 кэшем более интересная — процессор AMD Ryzen 5 PRO 4650U архитектуры Zen2 семейства Renoir имеет два CCX (блока ядер), каждый из которых имеет свой L3 кэш размером 4Mb. Сначала процессор пользуется кэшем своего CCX, далее со следующим приростом после 4Mb пользуется кэшем чужого CCX, что негативно влияет на производительность. Когда заканчивается кэш L3 обоих CCX (после 8Mb), производительность ухудшается ещё заметнее.

Вследствие предвыборки данных, прямой и обратный обходы не росли с увеличением размера массива. Зато видна особенность обхода массива в случайном порядке: в этом случае количество тактов процессора, затраченное на получение элемента массива, заметно растет с увеличением размера массива потому что что кэш-контроллер не может корректно выполнить предвыборку данных для обхода массива в случайном порядке.

1. **Заключение**

В ходе данной работы были установлены размеры кэшей процессора Ryzen 5 PRO 4650U и выяснено, насколько существенен прирост времени обращения при "переходе" с одного кэша на другой.