**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 № 8**

«ВЛИЯНИЕ КЭШ-ПАМЯТИ НА ВРЕМЯ ОБРАБОТКИ МАССИВА»

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. Исследование зависимости времени доступа к данным в памяти от их  
объема.

2. Исследование зависимости времени доступа к данным в памяти от порядка их обхода.

**Задание**

1. Написать программу, многократно выполняющую обход массива  
заданного размера тремя способами.  
2. Для каждого размера массива и способа обхода измерить среднее время  
доступа к одному элементу (в тактах процессора). Построить графики  
зависимости среднего времени доступа от размера массива.  
3. На основе анализа полученных графиков:  
- определить размеры кэш-памяти различных уровней, обосновать  
ответ, сопоставить результат с известными реальными значениями;  
- определить размеры массива, при которых время доступа к элементу  
массива при случайном обходе больше, чем при прямом или  
обратном; объяснить причины этой разницы во временах.

4. Составить отчет по лабораторной работе.

**Описание работы**

* Написал на языке программирования Си++ программу, выполняющую обход массива в прямом, обратном и случайном порядке, варьируя размер обрабатываемого массива. Обеспечил предварительным обходом прогрев кэш-памяти. Данные о граничных значениях размера массива определил на основании документации процессора.
* Измерил с помощью функции \_\_rdtsc() время обхода массива в тактах процессора, определил средние значения.
* Построил графики зависимости времени обработки массива в зависимости от размера массива.
* Составил отчёт о выполненной работе.

**Заключение**

В соответствии с эмпирическими принципами работы кэш-контроллера опытным путем выяснили наиболее предпочтительные порядки обхода массива с точки зрения оптимизации по времени работы программы.

**Приложение 1**

/\* Copyright 2022 Olimpiev Y. \*/

#include <ctime>

#include <iostream>

#include <stdlib.h>

#include <vector>

#include <x86intrin.h>

#include <cstring>

volatile union ticks{

unsigned long long t64;

struct s32 { unsigned long th, tl; } t32;

} start, end;

static size\_t ArrayAccess(size\_t size, int\*(\*order)(size\_t)) {

int\* arr = order(size);

unsigned long long time = 0;

start.t64 =\_\_rdtsc();

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

k = arr[k];

}

end.t64 = \_\_rdtsc();

time = (end.t64 - start.t64);

free(arr);

return time / size;

}

static size\_t ArrayTour(size\_t size, int\*(\*order)(size\_t), size\_t iterAmount) {

size\_t time = 0;

// Warm up cache.

ArrayAccess(size, order);

for (size\_t i = 1; i < iterAmount; i++) {

time += ArrayAccess(size, order);

}

return time / iterAmount;

}

static int\* InitCircleLinkedFwdArray(size\_t size) {

int\* ret = (int\*)calloc(size, sizeof(ret[0]));

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

ret[i] = (i + 1) % size;

}

return ret;

}

static int\* InitCircleLinkedRvsArray(size\_t size) {

int\* ret = (int\*)calloc(size, sizeof(ret[0]));

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

ret[i] = size - 1 - i - 1;

}

return ret;

}

static int\* InitCircleLinkedRndArray(size\_t size) {

std::srand(std::time(0));

int\* ret = (int\*)calloc(size, sizeof(ret[0]));

std::vector<int> indexes;

indexes.reserve(size);

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

indexes.push\_back(i);

}

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

size\_t index = std::rand() % indexes.size();

ret[i] = indexes[index];

indexes.erase(indexes.begin() + index);

}

return ret;

}

int main(int argc, char\* argv[]) {

size\_t minSize = 128;

size\_t maxSize = 16000000;

size\_t currSize = minSize;

// Default order = fwd.

int\*(\*order)(size\_t) = InitCircleLinkedFwdArray;

if (argc == 2) {

if (strcmp(argv[1], "fwd") == 0) {

std::cout << "forward order on" << std::endl;

order = InitCircleLinkedFwdArray;

}

if (strcmp(argv[1], "rvs") == 0) {

std::cout << "reverse order on" << std::endl;

order = InitCircleLinkedRvsArray;

}

if (strcmp(argv[1], "rnd") == 0) {

std::cout << "random order on" << std::endl;

order = InitCircleLinkedRndArray;

}

}

while (currSize < maxSize) {

size\_t iterationsAmount = 3;

size\_t time = 0;

time = ArrayTour(currSize, order, iterationsAmount);

printf("Array size:%lu bytes.\tAverage access to an element:%lu ticks.\n", currSize \* sizeof(int), time);

currSize \*= 2;

}

return EXIT\_SUCCESS;

}

**Build on Linux Ubuntu:**

g++ -O0 main.cpp -o lab8

**Приложение 2**

График зависимости времени обхода программы (в тактах процессора) в зависимости от рамзера и порядка обхода.