**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, 21203 группы

Неретина Степана Ивановича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кандидат технических наук, доцент кафедры параллельных вычислений

А.Ю. Власенко

Новосибирск 2020

**СОДЕРЖАНИЕ**

ЦЕЛЬ…………………………………………………………………...……3

ЗАДАНИЕ……………………………………………………………………4

ОПИСАНИЕ РАБОТЫ……………...………..…………………………..…5

ЗАКЛЮЧЕНИЕ……………...…………………..………………………..…6

Приложение 1. Код программы...…………...………………………….…..8

Приложение 2. Генерация графика с помощью Python………………….10

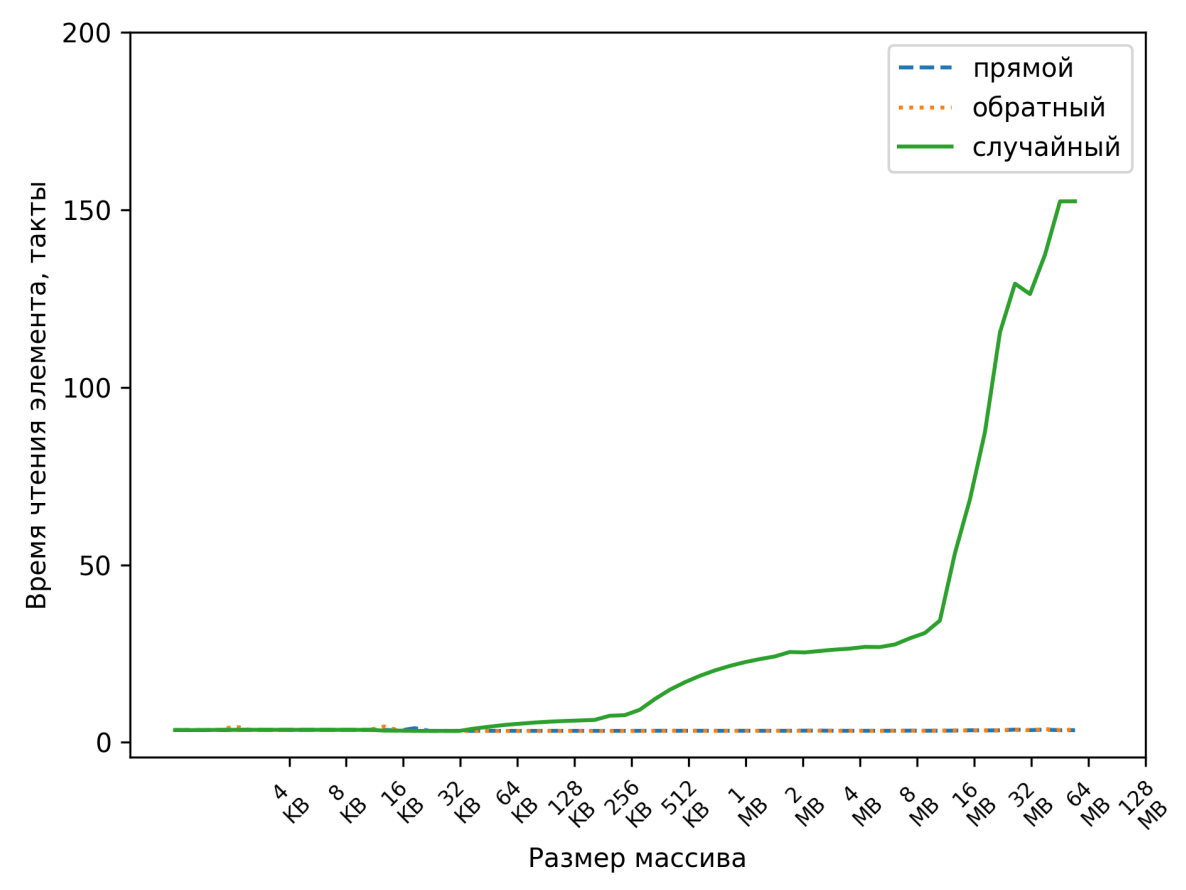
**ЦЕЛЬ**

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.

**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.

**График роста тактов обращения**

****

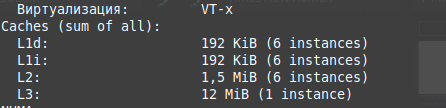
**Оценка уровня кэша**

По графику можно заметить, что прирост тактов начинается с 192kB, что явно говорит о том, что размер L1 кэша равен 192kB.

Следующее возрастание начинается с 1,5MB, что соответствует размеру L2 кэша – 1,5MB.

В следующий раз более сильный прирост начинается после 12MB, что очень похоже на размер L3 кэша.

Данные, полученные **c помощью команды lscpu** подтверждают данные предположения.



**ЗАКЛЮЧЕНИЕ**

В ходе данной работы были установлены размеры кэшей процессора i7-10750H и выяснено, насколько существенен прирост времени обращения при "переходе" с одного кэша на другой.

**Приложение 1. Код программы**

#include <cstdio>

#include <cstdlib>

#include <string>

constexpr inline int MIN\_N = 1024 / sizeof(int);

constexpr inline int MAX\_N = 32 \* 1024 \* 1024LL / sizeof(int);

void gen\_forward(int\* data, const int iterations\_count) {

for (int i = 0; i < iterations\_count - 1; i++) {

data[i] = i + 1;

}

data[iterations\_count] = 0;

}

void gen\_backward(int\* data, const int N) {

data[0] = N - 1;

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

data[i] = i - 1;

}

}

void gen\_random(int\* data, const int N) {

int\* perm = (int\*)malloc(N \* sizeof(int));

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

perm[i] = i;

}

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

int j = rand() % (i + 1);

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

}

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

data[perm[i]] = perm[(i + 1) % N];

}

free(perm);

}

int test(int\* data, const int N, const size\_t iterations\_count, void (\*gen)(int\* data, const int N)) {

gen(data, N);

int k;

long long i;

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

k = data[k];

}

unsigned long long start = \_\_builtin\_ia32\_rdtsc();

for (k = 0, i = 0; i < N\*iterations\_count; i++) {

k = data[k];

}

unsigned long long end = \_\_builtin\_ia32\_rdtsc();

printf(" %d, %zu, %llu\n", N\*4, iterations\_count, (end-start) / N / iterations\_count);

return k;

}

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

if (argc < 2) return 1;

auto K = (size\_t) (atoi(argv[1]));

int\* data = (int\*)malloc(MAX\_N \* sizeof(int));

for (int N = MIN\_N; N <= MAX\_N; N \*= 1.2) {

printf("forward,");

test(data, N, K, gen\_forward);

printf("backward,");

test(data, N, K, gen\_backward);

printf("random,");

test(data, N, K, gen\_random);

}

free(data);

return 0;

}

**Приложение 2 . Генерация графика с помощью Python.**

from matplotlib import pyplot as plt

import subprocess

from math import log2

plt.figure(figsize=(8, 6))

\_suffixes = ['B', 'KB', 'MB', 'GB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB']

def file\_size(size):

# determine binary order in steps of size 10

# (coerce to int, // still returns a float)

order = int(log2(size) / 10) if size else 0

# format file size

# (.4g results in rounded numbers for exact matches and max 3 decimals,

# should never resort to exponent values)

return '{:.4g}\n{}'.format(size / (1 << (order \* 10)), \_suffixes[order])

x = {

"forward": [],

"backward": [],

"random": []

}

y = {

"forward": [],

"backward": [],

"random": []

}

n = 10

out = subprocess.Popen(['./cmake-build-debug/lab5', str(n)], stdout=subprocess.PIPE)

data = out.stdout.readlines()

for item in data:

line = item.strip().decode("utf-8").split(",")

try:

x[line[0]].append(int(line[1]))

y[line[0]].append(int(line[3]))

except Exception as e:

print(e)

plt.plot(x["forward"], y["forward"], ls="dashed", label="прямой")

plt.plot(x["backward"], y["backward"], ls="dotted", label="обратный")

plt.plot(x["random"], y["random"], ls="solid", label="случайный")

xticks\_a= [0, 8\*1048576, 16\*1048576, 24\*1048576, 32\*1048576]

yticks\_a = [0, 50, 100, 150, 200]

plt.xticks(xticks\_a, ['0 MB', '8 MB', '16 MB', '24 MB', '32 MB'], rotation = 45 )

plt.yticks(yticks\_a)

plt.xlabel("Размер массива")

plt.ylabel("Время чтения элемента, такты")

plt.legend()

plt.savefig('res.png', dpi=300)