

**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 20203

Синюкова Валерия Константиновича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
доцент кафедры параллельных
вычислений
Власенко Андрей Юрьевич

Новосибирск 2021

СОДЕРЖАНИЕ

ЦЕЛЬ.....	3
ЗАДАНИЕ	3
ОПИСАНИЕ РАБОТЫ	4
ЗАКЛЮЧЕНИЕ	6
Приложение 1. <i>Листинг программы, реализующей вычисление среднего времени доступа к элементу массива в зависимости от количества фрагментов</i>	7

ЦЕЛЬ

Экспериментальное определение степени ассоциативности кэш-памяти.

ЗАДАНИЕ

1. Написать программу, выполняющую обход памяти в соответствии с заданием (см. описание лабораторной работы №9).
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 по практической работе. Отчет должен содержать следующее.
 - a. Титульный лист.
 - b. Цель практической работы.
 - c. Параметры теста: размер фрагментов, величина смещения.
 - d. График зависимости среднего времени доступа к элементу массива от числа фрагментов.
 - e. Оценку степени ассоциативности различных уровней кэш-памяти согласно выполненным вычислительным экспериментам.
 - f. Реальные значения степеней ассоциативности различных уровней кэш-памяти процессора, подкрепленные доказательствами (скриншоты из программ типа CPU-Z, файлы операционной системы, куски официальной документации по процессору и т.д.)
 - g. Полный компилируемый листинг реализованной программы и команды для ее компиляции.
 - h. Вывод по результатам практической работы.

ОПИСАНИЕ РАБОТЫ

- 1) Была написана программа на языке C++, реализующая заполнение и обход массива данных типа `int` в соответствии с заданием, а также вычисляющая среднее время доступа к элементу массива в зависимости от количества фрагментов, к которым производится доступ за время обхода (*фрагменты* – места в оперативной памяти, которые при попытке доступа к ним, будут отображены в одни и те же множества кэш-памяти, из-за чего, когда количество фрагментов начнет превышать степень ассоциативности кэша, будет возникать кэш-букование: какая-то кэш-строка из множества будет перезаписана, из-за этого среднее время доступа к элементу массива увеличится).

Поскольку тестирование проводилось на 64-битной системе, то в каждом фрагменте находилось 16 элементов массива. Поскольку размер кэша у процессора Intel Pentium CPU G3220 @ 3.00GHz, на котором проводилось тестирование, равен 3 МБ, то есть 786432 элемента типа `int`, смещение было выбрано, соответственно, равно 783432 (*смещение* – “расстояние” между двумя элементами массива, к которым производится доступ, выбранное таким образом, чтобы эти два элемента были отображены в одно множество кэш-памяти, то есть кратное размеру банка кэш-памяти).

С листингом данной программы можно ознакомиться в [приложении](#).

- 2) По результатам измерений была составлена таблица зависимости среднего времени доступа к элементу массива в зависимости от количества фрагментов.

количество фрагментов	среднее время доступа к элементу массива (такт)
1	1
2	2
3	6
4	6
5	14
6	14
7	14
8	14
9	28
10	28
11	29
12	28
13	29
14	28
15	28
16	29
17	32
18	36

19	36
20	36
21	36
22	36
23	36
24	36
25	36
26	36
27	36
28	36
29	36
30	36
31	42
32	42

По данной таблице был построен график.



Выделим количества фрагментов, после которых на данном графике наблюдаются “скачки” среднего времени доступа к элементам массива, сделаем предположения, почему эти “скачки” произошли:

1. 4: соответствует степени ассоциативности буфера трансляции адресов.
 2. 8: соответствует степеням ассоциативности кэш-памяти первого и второго уровней.
 3. 16: соответствует степени ассоциативности кэш-памяти третьего уровня.
- 3) Сравним значения из второго пункта с реальными степенями ассоциативности кэш-памяти разных уровней процессора Intel Pentium CPU G3220 @ 3.00GHz, на котором проводилось тестирование.

Cache		
L1 Data	2 x 32 KBytes	8-way
L1 Inst.	2 x 32 KBytes	8-way
Level 2	2 x 256 KBytes	8-way
Level 3	3 MBytes	12-way

Предположение об ассоциативности кэш-памяти первого и второго уровней было верно.

Предположение об ассоциативности кэш-памяти третьего уровня оказалось не верно.

ЗАКЛЮЧЕНИЕ

По результатам данной практической работы была написана программа, измеряющая среднее время доступа к элементу массива в зависимости от количества фрагментов, к которым производится доступ за время обхода. По данным измерений были построены таблица и график, на основании которых были сделаны предположения о степенях ассоциативности кэш-памяти разных уровней. Предположенные значения были сравнены с реальными степенями ассоциативности.

**Приложение 1. Листинг программы, реализующей вычисление
среднего времени доступа к элементу массива в зависимости от
количества фрагментов**

```
#define M 1000000
#define OFFSET 786432
#define NUMBER_OF_FRAGMENTS 32
#define NUMBER_OF_ELEMENTS_IN_FRAGMENT 16

#include "include/lab1.h"
#include <fstream>
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <stdint.h>

using namespace std;

void arrayFilling(int*& x, int N, int numberOfFragments)
{
    int i, j;
    x = new int[N];
    for (j = 0; j < NUMBER_OF_ELEMENTS_IN_FRAGMENT; ++j)
        for (i = 0; i < numberOfFragments - 1; ++i)
            x[i * OFFSET + j] = (i + 1) * OFFSET + j;
    for (i = 0; i < NUMBER_OF_ELEMENTS_IN_FRAGMENT - 1; ++i)
        x[(numberOfFragments - 1) * OFFSET + i] = 1 + i;
    x[(numberOfFragments - 1) * OFFSET + NUMBER_OF_ELEMENTS_IN_FRAGMENT -
1] = 0;
}

void bypass(ofstream& out, int* x, int N, int numberOfMeasurements, int
numberOfFragments)
{
    int j, i, m, k;
    uint64_t t, min, tbegin, tend;
    uint32_t t1begin, t1end, t2begin, t2end;
    for (j = 0; j < numberOfMeasurements; ++j)
    {
        k = 0;
        tbegin = __builtin_ia32_rdtsc();
        for (i = 0; i < NUMBER_OF_ELEMENTS_IN_FRAGMENT *
numberOfFragments; ++i)
            k = x[k];
        tend = __builtin_ia32_rdtsc();
        t = (uint64_t)((tend - tbegin) / (NUMBER_OF_ELEMENTS_IN_FRAGMENT
* numberOfFragments));
        if (123 == k)
            cout << "!!!!";
        if ((0 == min) || (t < min))
            min = t;
    }
    out << numberOfFragments << ',' << min << endl;
}
```

```

int main(int argc, char** argv)
{
    srand((unsigned)time(0));
    int* x = NULL, numberOfMeasurments = atoi(argv[1]);
    double Pi = LeibnizFormula(M);
    int N;
    time_t time;
    if (1.1 == Pi)
        cout << "!!!!";
    ofstream out(argv[2]);
    if (!out)
    {
        cout << "couldn't open output file" << endl;
        return 1;
    }
    int i;
    for (i = 0; i < NUMBER_OF_FRAGMENTS; ++i)
    {
        N = OFFSET * (i + 1);
        arrayFilling(x, N, i + 1);
        bypass(out, x, N, numberOfMeasurments, i + 1);
        delete[] x;
    }
    return 0;
}

```