ЦЕЛЬ

Экспериментальное определение степени ассоциативности кэш-памяти.

ОПИСАНИЕ РАБОТЫ

Для создания кэш-буксования будем создавать массив, равный размеру смещения между фрагментами, умноженному на количество фрагментов, а потом обходить этот массив, обращаясь сначала к первому элементу в каждом фрагменте, потом ко второму, к третьему, и так пока в каждом фрагменте не закончатся элементами.

Код, который реализует такой массив:

```
#define SIZE (1 << 20)
#define OFFSET (1 << 24)

void create_array(int* array, int N, int fragment_size) {
    for (int i = 0; i < N - 1; i++) {
        for(int j = 0; j < fragment_size; j++) {
            array[OFFSET * i + j] = ((i + 1) * OFFSET) + j;
        }
    }

    for (int j = 0, ind = 1; j < fragment_size - 1; j++, ind++) {
        array[OFFSET * (N - 1) + j] = ind;
    }
    array[OFFSET * (N - 1) + fragment_size - 1] = 0;
}
```

Где Offset это смещение, а Size максимальный размер каждого фрагмента.

Сначала заполним массив двумя фрагментами, затем тремя, и так будем заполнять до 32 фрагментов.

Получили следующий график:

Как можем заметить незначительный скачок происходит на количестве фрагментов, равном 7, более значительный на 14, потом постепенно замедляется до 18 фрагментов, а дальше незначительно меняется в пределах 45-50 тактов.

Каждая примерно плавная, почти горизонтальная линия, означает собой новый уровень ассоциативности (скачок на семи соответствует степени ассоциативности буфера трансляции адресов).

ЗАКЛЮЧЕНИЕ

Я изучила такое понятие, как буксование кэщ-памяти, с его помощью научилась экспериментально определять степени ассоциативности кэш-памяти своего процессора.

Приложение 1. Листинг программы

```
#include <iostream>
#include <malloc.h>
#include <float.h>
#include <cstdint>

using namespace std;

#define SIZE (1 << 20)
#define OFFSET (1 << 24)

void create_array(int* array, int N, int fragment_size) {
    for (int i = 0; i < N - 1; i++) {
        for(int j = 0; j < fragment_size; j++) {
            array[OFFSET * i + j] = ((i + 1) * OFFSET) + j;
        }
    }

    for (int j = 0, ind = 1; j < fragment_size - 1; j++, ind++) {
        array[OFFSET * (N - 1) + j] = ind;
    }
    array[OFFSET * (N - 1) + fragment_size - 1] = 0;
}

int main() {
    for (int N = 2; N < 30; N++) {
        int* array = (int*) malloc(sizeof(int) * OFFSET * N);
        int fragment_size = SIZE / N;
        create_array(array, N, fragment_size);

        int K = 69;

        unsigned int high, low;

        asm volatile ("rdtsc\n":"=a"(low),"=d"(high));
        uint64_t start = ((uint64_t)high << 32) | low;

        for (int k = 0, i = 0; i < SIZE * K; i++) {
            k = array[k];
        }

        asm volatile ("rdtsc\n":"=a"(low),"=d"(high));
        uint64_t end = ((uint64_t)high << 32) | low;

        cout << ((double)(end - start) / (SIZE * K)) << endl;

        free(array);
    }
}
```