САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 3

«Кэш-память»

Выполнил(а): ДЗЕСТЕЛОВ ХЕТАГ АРТУРОВИЧ

студ. гр. М3139

Санкт-Петербург

2020

Цель работы: закрепление материала по теме «кэш-память» путем решения задач по данной теме.

Условие задачи

Вариант 1.2. Имеем следующий фрагмент кода:

```
struct ab_array
{
	int a;
	int b;
};

struct ab_array arr[100];

for (i=0; i<100; i++)
{
	arr[i].a = arr[i].a + 5;
	if (i <= 90)
		arr[i].b = arr[i].a * 3;
	else
		c = c + 1;
}
```

Имеется система с полностью ассоциативным кэшем данных размером 16 КБ. Размер кэш-линии 32 байта. При инициализации кэш пустой.

Определить отношение кол-ва промахов к кол-ву всех обращений к памяти (промахи и обращения считаются только для массивов a и b).

Решение задачи

Имеем полностью ассоциативный кэш (fully associative). Это означает, что фактически имеем один блок, который содержит все кэш линии. Размер кэш-линии 32 байта.

Проанализируем имеющиеся данные: структура, состоящая из двух переменных типа `int` (что соответствует 4-м байтам). На хранение всего массива потребуется $8 * 100 = 800$ байт < 16 КБ – полностью помещается в кэш. Структура хранится в памяти последовательно, а т. к. оба типа эквивалентны, то объём хранения – $2 * 4$ байт $= 8$ последовательных байт на одну ячейку массива `arr`. Отсюда в одну кэш-линию помещается до $32/8 = 4$ ячеек массива `arr`.

Проанализируем исполнение алгоритма: обращение к ячейкам массива происходит в следующих строках:

- `arr[i].a = arr[i].a + 5;`
- `arr[i].b = arr[i].a * 3;`

Заметим, что обращение происходит по два раза в каждой строке. Рассмотрим исполнение на 0-й итерации:

1. Попытка обращения в кэш к адресу ячейки массива `arr[0].a` – промах;
2. Загрузка в первую кэш-линию 4-х последовательных ячеек массива `arr[0] - arr[3];`
3. Две попытки обращения в кэш к адресу ячейки массива `arr[0].a` – два обращения (попадание в кэш);
4. Две попытки обращения в кэш к адресам ячеек массива `arr[0].b` и `arr[0].a` – два обращения (попадание в кэш).

Последующие три итерации промахов в кэш не дадут, элементы лежат в кэше.

Заметим, что независимо от результата условного оператора в теле цикла каждые 4 итерации будет происходить промах при обращении в кэш. Подсчитаем кол-во промахов и кол-во обращений:

1. С 0-й по 99-ю итерацию произойдёт 100 / 4 = 25 промахов;

2. С 0-й по 90-ю итерацию произойдёт 91 * 4 = 364 обращения.
3. С 91-й по 99-ю итерацию произойдёт 9 * 2 = 18 обращений.

Итого имеем 25 промахов и 382 обращения, что в отношении даёт $\mathbf{25/382}$.