САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 3

«КЕШ-ПАМЯТЬ»

Выполнил: Султанов Мирзомансурхон Махсудович

студ. гр. М313Д

Санкт-Петербург

2020

Цель работы: закрепление материала по теме “кэш-память” путем решения задач по данной теме

Условие задачи

Имеется следующее определение глобальных переменных и функций

Таблица 1 – Глобальные переменные и функции

Глобальные переменные	Функции

unsigned int size = 1024 * 1024;
double x[size];
double y[size];
double z[size];
double xx[size];
double yy[size];
double zz[size];
	void f(double w) {
	for (unsinged int i=0; i<size; ++i) {
	x[i] = xx[i] * w + x[i];
	y[i] = yy[i] * w + y[i];
	z[i] = zz[i] * w + z[i];
	}
	}

Рассмотрим систему с L1 кэшем данных с ассоциативностью 4-way размером 32 КБ и размером строки 64 байта. Кэш L2 представляет собой 8-way ассоциативный кэш размером 1 МБ и размером строки 64 байта. Алгоритм вытеснения: LRU. Массивы последовательно хранятся в памяти, и первый из них начинается с адреса, кратного 1024.

Определите процент попаданий (число попаданий к общему числу обращений) для кэшей L1 и L2 для выполнения предложенной функции.

В ответе нужно представить два числа, равных % попаданий для L1 и L2 кэшей.

Практическая часть

Заметим, что так как для хранения double необходимо 8 байт, а размер кэш-линии составляет 64 байта, то в одной кэш-линии может быть 8 элементов. При этом элементы x[k…k+7], y[k…k+7], z[k…k+7], xx[k…k+7], yy[k…k+7], zz[k…k+7], где k кратно 8 ссылаются на один и тот же блок из кэш-линий в L1 и L2.

Рассмотрим, что происходит в нулевой итерации:

В самом начале оба кэша пустые. Выполняя команду “x[0] = xx[0] * w + x[0];”, мы обращаемся к L1 3 раза. Сначала мы пытаемся получить кэш-линии со значениями x[0…7] и xx[0…7]. Но, получив кэш-промах в L1, мы обращаемся к L2 за этими значениями, но нетрудно догадаться, что и там мы получаем кэш-промах. В конце концов мы получаем кэш-линии x[0…7] и xx[0…7] из памяти в L2, а затем из L2 в L1, а из L1 на процессор. Затем необходимо записать новое значение x[0] в L1, что у нас получается сделать без промаха.

Выполняя команду “y[0] = yy[0] * w + y[0];”, получаем аналогичным образом 1 попадание из 3 обращений у L1 и 0 попаданий из 2 обращений к L2.

Выполняя же команду “z[0] = zz[0] * w + z[0];”, пытаясь получить значения z[0] и zz[0] мы опять наткнёмся на кэш-промах в L1, а затем и в L2. Затем получаем кэш-линии z[0…7] и zz[0…7] из памяти в L2, а вот группа (множество) кэш-линий в L1 уже будет полным (так как в L1 ассоциативность 4-way), поэтому чтобы переместить кэш-линии z[0…7] и zz[0…7] из L2 в L1, необходимо сначала согласно алгоритму вытеснения LRU(“least recently used – меньше всего использовался в последнее время”) вытеснить строки x[0…7] и xx[0…7]. Затем записываем новое значение z[0] в L1 без промаха.

Таким образом, за нулевую итерацию к L1 мы обратились 9 раз и попали 3 раза (каждый раз когда записывали новые значения), а к L2 мы обратились 6 раз и ни разу не попали.

Теперь разберёмся, что происходит с первой по седьмую итерацию:

Что происходит на L1? На первой команде первой итерации аналогично нулевой итерации нам потребуются элементы x[1] и xx[1], однако в группе (множестве) кэш-линий сейчас находятся, z[0…7], zz[0…7], y[0…7], yy[0…7]. По алгоритму LRU необходимо вытеснить y[0…7] и yy[0…7], а затем нужно записать новое значение x[1]. Заметим, что аналогично на следующих командах мы всегда будем получать кэш-промахи на L1, когда будем читать элементы, и попадать, когда нужно будет записать новое

получившееся значение. Таким образом, на каждой итерации с первой по седьмую мы обращаемся к L1 9 раз и попадаем 3 раза.

Что же происходит в то время на L2? Каждый раз, когда мы вытесняем значения двух строк из L1, нам нужно получить нужные кэш-линии из L2, а после нулевой итерации, они как раз здесь и хранятся. Таким образом, на каждой итерации с первой по седьмую мы обращаемся к L2 6 раз и попадаем все 6 раз.

Для удобства соберём всю получившуюся информацию во второй таблице.

Таблица 2 – Обращения и попадания на 0…7 итерации.

Итерации --->	0	1	2	3	4	5	6	7
Обращения к L1	9	9	9	9	9	9	9	9
Попадания в L1	3	3	3	3	3	3	3	3
Обращения к L2	6	6	6	6	6	6	6	6
Попадания в L2	0	6	6	6	6	6	6	6

Теперь заметим то, что данные значения будут циклично повторяться. Действительно, на итерации, кратной 8, мы будем работать с новыми группами (множествами) кэш-линий и все действия будут повторяться. А так как количество кратно 8, то и общий процент попадания будет равен проценту попадания на итерациях 0…7. Таким образом, процент попадания на L1 = 27/72 = 1/3 ≈ 33,4%; процент попадания на L2 = 6*7 / 6*8 = 7/8 = 87,5%.