САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе №3

«Кэш-память»

Выполнил(а): Ступников Александр Сергеевич

студ. гр. М3135

Санкт-Петербург

2020

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

Условие задачи

Вариант 2. Имеется следующее определение глобальных переменных и функций

Вариант	Глобальные переменные	Функции

2	`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

1. Ответ: 33.(3)%, 87.5%.

2. Описание решения.

Для начала нужно сказать, что в одной кэш линии умещается восемь элементов одного из массивов x, y, z, xx, yy, zz, так как на кодирование одного числа в формате double уходит 8 байт, а размер линии кэша 64 байт. 64 / 8 = 8 чисел в одной линии кэша. Также следует уточнить, что запись a[x…y] означает все элементы массива a с индексами i такими, что $x \le i \le y$.

Под блоком кэш линий подразумевается set линий ассоциативного кэша. Теперь перейдём непосредственно к решению.

Для ответа на вопрос задачи рассмотрим работу алгоритма при выполнении цикла for. Первый шаг цикла for, i = 0.

Строчка «x[i] = xx[i] * w + x[i]»: сначала происходит обращение в кэш на чтение xx[0], промах, xx[0…7] записываются в кэш L1 в некоторую кэш линию. Далее происходит обращение в кэш на чтение x[0], промах, x[0…7] записываются в кэш L1 в некоторую кэш линию. В конце происходит обращение в кэш на запись x[0], попадание в кэш L1, значение x[0] обновляется в кэше L1.

Строчка «y[i] = yy[i] * w + y[i]»: происходит всё то же самое, только вместо массива xx – массив yy, вместо массива x – массив y.

Строчка «z[i] = zz[i] * w + z[i]»: здесь происходит нечто интересное. Ранее было сказано, что x[0…7], xx[0…7], y[0…7], yy[0…7] записываются в некоторые кэш линии. Заметим, что эти кэш линии находятся в одном и том же блоке, состоящем из 4 кэш линий (ассоциативность L1 равна 4).

Здесь необходимо остановиться и понять, почему это так.

Чтобы для данного кэша L1 два «набора» данных (две кэш линии) с разными адресами были записаны из памяти в один и тот же блок кэш линий, необходимо, чтобы последние 13 бит адресов этих наборов данных совпадали, то есть остатки при делении этих адресов на 2^{13} должны быть равны.

Рассмотрим элемент x[0], его адрес характеризует адрес набора данных x[0…7] (этот набор составляет кэш линию). Из условия задачи известно, что x[0] имеет в памяти адрес кратный 1024, пусть этот адрес равен k, тогда k % 1024 = 0, пусть p = k % 2^{13}.

Рассмотрим элемент y[0], его адрес характеризует адрес набора данных y[0…7] (этот набор составляет кэш линию). y[0] имеет адрес в памяти k + size(x), где size(x) = 1024 * 1024 * 8 – размер массива x в байтах. В итоге (k + 1024 * 1024 * 8) % 2^{13} = k % 2^{13} + (1024 * 1024 * 8) % 2^{13} = p + 0 = p.

То есть кэш линии, в которых хранятся данные x[0…7] и y[0…7] принадлежат одному блоку. Аналогично к этому же блоку после записи в

кэш будут принадлежать кэш линии, хранящие данные z[0…7], xx[0…7], yy[0…7], zz[0…7].

По тем же причинам кэш линии, сохраняющие x[i…i + 7], xx[i…i + 7], y[i…i + 7], yy[i…i + 7], z[i…i + 7], zz[i…i + 7], где i % 8 = 0, относятся к одному и тому же кэш блоку и не могут одновременно вместе находится в кэше L1.

Теперь снова вернёмся к строчке «z[i] = zz[i] * w + z[i]».

При обращении к zz[0] произойдёт промах и числа zz[0…7] запишутся в некоторую кэш линию, которая принадлежит тому же блоку, в котором находятся кэш линии, в которые записаны значения x[0…7], xx[0…7], y[0…7], yy[0…7]. То есть при записи в кэш данные zz[0…7] должны вытеснить некоторые данные из этого блока. Так как алгоритм вытеснения LRU, zz[0…7] вытеснят из кэша L1 в кэш L2 xx[0…7]. Дальше произойдёт кэш промах при попытке прочитать из кэша z[0], в кэш L1 будут записаны данные z[0…7], они вытеснят собой данные x[0…7]. Наконец произойдёт попадание в кэш L1 при записи z[0].

Теперь рассмотрим второй шаг цикла for, i = 1.

Строчка «x[i] = xx[i] * w + x[i]»: сначала происходит обращение в кэш на чтение xx[0], промах для кэша L1, попадание в кэш L2, xx[0…7] записываются в кэш L1 в некоторую кэш линию, вытесняя собой yy[0…7] из кэша L1 в кэш L2. Далее происходит обращение в кэш на чтение x[0], промах для кэша L1, попадание в кэш L2, x[0…7] записываются в кэш L1, вытесняя собой y[0…7] из кэша L1 в кэш L2. В конце происходит обращение в кэш на запись x[0], попадание в кэш L1, значение x[0] обновляется в кэше L1.

Строчка «y[i] = yy[i] * w + y[i]»: происходит всё то же самое, только вместо массива xx – массив yy, вместо массива x – массив y, yy[0…7] вытесняет zz[0…7], y[0…7] вытесняет z[0…7].

Строчка «z[i] = zz[i] * w + z[i]»: Всё аналогично.

На следующих итерациях будет происходить то же, что на первом шаге цикла, если i % 8 = 0; или то же, что на втором шаге цикла, если i % 8 ≠ 0.

Подведём итоги:

1. i % 8 = 0

   Кэш L1: всего обращений в кэш 9 (6 на чтение, 3 на запись), из них 0 кэш попаданий при чтении, 3 кэш попадания при записи.

   Кэш L2: всего обращений в кэш 6 (6 на чтение, 0 на запись), все они кэш промахи.

2. i % 8 ≠ 0

   Кэш L1: всего обращений в кэш 9 (6 на чтение, 3 на запись), из них 0 кэш попаданий при чтении, 3 кэш попадания при записи.

   Кэш L2: всего обращений в кэш 6 (6 на чтение, 0 на запись), из них все кэш попадания.

За 8 шагов цикла:

9 * 8 обращений в L1, 3 * 8 попаданий в L1. Процент попаданий (3 * 8) / (9 * 8) = 33.(3)%

6 * 8 обращений в L2, 6 * 7 попаданий в L2. Процент попаданий (6 * 7) / (6 * 8) = 87.5%.

Ответ: 33.(3)%, 87.5%.