САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 3

«Кэш-память»

Выполнил: Лымарь Павел Игоревич

студ. гр. М313Д

Санкт-Петербург

2020

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

Условие задачи

1. Имеем следующее определение глобальных переменных и функций:

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];`
	`  }`
	`}`

Рисунок 1 – определение глобальных переменных и функций

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.

Определим процент попаданий (отношение числа попаданий к общему числу обращений) для кэшей L1 и L2 для выполнения предложенной функции.

Практическая часть

2. Каждое число типа double занимает в памяти 8 байт. Т.е. в одну кэш-строку будет попадать ровно $\frac{64}{8} = 8$ чисел типа double.

   Первая итерация:

   Опираясь на код, описанный выше, на первой итерации мы вначале обратимся к значению xx[0], и т.к. не найдем его в кэш-памяти L1 и L2 перепишем все значения xx[i] (i = 0..7) в первую кэш-строку текущего блока кэшей L1 и L2 (в каждом блоке L1 ровно 4 строки, т.к. кэш имеет ассоциативность 4-way). Далее имеем обращение к переменной x[0], которая отсутствует в L1 и L2 (промах). Снова обращаемся к x[0], чтобы записать значение (попадание в кэш).

   После снова имеем схожую конструкцию, для массивов yy и y. Вновь 2 промаха и 1 попадание. В кэши, разумеется, добавим значения yy[i], y[i] (i = 0..7).

   В конце, массивов zz и z снова нет в кэшах L1 и L2 и для того, чтобы поместить значения yy[0..7], y[0..7] мы вытесняем строки со значениями xx[0..7] и x[0..7] из блока L1 (нехватка строк, для записи).

   Вторая итерация:

   На последующей итерации мы не найдем значений xx[1] и x[1] в L1, но найдем в L2. То же самое произойдет с yy[1] и y[1], т.к. по алгоритму LRU мы вытеснили кэш-строки с yy[0..7] и y[0..7] на предыдущем шаге, добавив в кэш L1 строки с xx[1..8] и x[1..8]. Далее мы не найдем zz[1], z[1] в L1, т.к. вытеснили строки с zz[0..7] и z[0..7] ранее, добавив в L1 стоки с yy[1..8] и y[1..8], но найдем их в L2.

   …

   Заметим, что на последующих 6 итерациях будет происходить то же, что и во второй, т.к. массивы в L1 будут вытеснять друг друга, и мы будем искать очередное значение в L2, где оно и находится. Т.е. из 9

обращений, на каждой итерации, к L1 только 3 будут попаданиями. А к L2 будет 6 обращений и каждое из них будет попаданием.

Тогда отношение числа попаданий ко всем обращениям, на итерациях $1..8$ цикла, для L1 будет $\frac{3 \cdot 8}{9 \cdot 8} = \frac{1}{3}$. В свою очередь данное отношение для L2 будет равно $\frac{6 \cdot 7}{6 \cdot 8} = \frac{7}{8}$.

Переменная size кратна 8 и при переходе от итерации $8k$, к итерации $8k + 1 (k \in Z)$, мы так же не найдем xx[$8k$] в L1 и L2, т.е. запустим цикл, идентичный итерациям $1..8$.

Таким образом, окончательный процент попаданий в L1 составит 33,3%, L2 - 87,5%.