САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 3

«КЭШ-ПАМЯТЬ»

Выполнил: Султанов Мирзомансурхон Махсудович

Номер ИСУ: 311629

студ. гр. М3134

Санкт-Петербург

2021

Цель работы: решение задач по теме «кэш-память».

Теоретическая часть

Каждый раз, когда процессор нуждается в данных из памяти, он обращается к ближайшему к нему кэшу (для 4-х уровневого кэша это, например, L4) и ожидает, что нужные данные окажутся там. Если данные действительно оказались там, то происходит попадание (hit), и мы можем сразу забрать данные, и дальше никуда идти не надо. Но вот если данных там не оказалось, то происходит промах (miss) и придётся идти к следующему кэшу, ожидая, что там окажутся данные, что отнимет у нас прилично времени, что и называют “штраф за промах”. Дальше картина повторяется. Проверяется, есть ли у следующего кэша данные. Либо мы промахиваемся, либо попадаем. При промахе идём дальше и т.д., пока не дойдём до оперативки в самом худшем случае. Почему же это самый худший случай? Потому что это отнимет больше всего времени, так как придётся пройти самый больший путь. Таким образом, нам хочется, чтобы попадали мы чаще, чем промахивались, чтобы минимизировать среднее время доступа к памяти.

А как мы можем найти среднее время доступа к памяти (AMAT)? Для подобных целей существует математическое ожидание, что и определяет среднее значение случайной величины. Здесь случайной величиной является время доступа к памяти. Перед этим ещё разберёмся с двумя важными понятиями. Коэффициент попадания (hit rate или hit ratio) и коэффициент промахов (miss rate или miss ratio) это вероятности того, что мы попадём в кэше или промахнёмся. Находятся данные величины следующим образом: общее число попаданий для первой величины или промахов для второй величины, делённое на количество запросов к кэшу. Будем считать, что время попадания (Hit time) — это время, требующееся,

чтобы взять данные из кэша при попадании. А время промаха (Miss time) — это время, требующееся, чтобы получить данные из ближайшего кэша либо же оперативки с этими данными при промахе. Тогда AMAT можно найти следующим образом:

AMAT = Hit ratio * Hit time + Miss ratio * Miss time

Также замечу, что Hit ratio + Miss ratio = 1, так как никаких других исходов не существует. Мы либо попали, либо промахнулись. Тогда эту формулу можно записать как:

AMAT = (1 – Miss ratio) * Hit time + Miss ratio * Miss time

Условие первой задачи

Имеются две системы, различающихся лишь кэшем.
На первой установлен общий кэш для команд и данных размеров 32 КБ. На второй используется раздельный кэш: 16 КБ для команд и 16 КБ для данных.

Штраф за промах 50 тактов. Коэффициенты промахов для кэшей представлены в таблице 1. Время отклика 1 такт для раздельных кэшей и доступа к командам в общем кэше, 2 такта для доступа к данным в общем кэше. 75% обращений от общего числа обращений к кэшу проводится по командам и 25% – по данным.

Таблица 1 – Коэффициент промахов для кэшей

Размер	Кэш команд	Кэш данных	Общий кэш
16 КБ	0.64%	6.47%	2.87%
32 КБ	0.15%	4.82%	1.99%

Нужно определить среднее время обращения к памяти (AMAT) для этих кэшей.

Решение первой задачи

Для обоих случаев всего есть 4 варианта развития. У нас есть обращение по команде, и мы можем либо попасть, либо промахнуться. Или же у нас есть обращение по данным, и опять же мы можем либо попасть, либо промахнуться. Если мы промахнулись, то нам придётся потратить 50 тактов + время доступа в том или ином случае. Тогда AMAT для раздельного кэша можно найти следующим образом:

$$AMAT = 0.75 * ((1 – 0.0064) * 1T + 0.0064 * 51\ T) + 0.25 * ((1 – 0.0647) * 1T + 0.0647 * 51\ T) = 2.04875T$$

А для общего кэша выйдет:

$$AMAT = 0.75 * ((1 – 0.0199) * 1T + 0.0199 * 51T) + 0.25 * ((1 – 0.0199) * 2T + 0.0199 * 52T) = 2.245T$$

Ответ: 2.04875T для раздельного кэша и 2.245T для общего кэша.

Условие второй задачи

Имеется кэш с прямым отображением, состоящий из 8 кэш-линий размером 64 байта. Запросы к байтам в памяти идут в следующем порядке:

204, 320, 181, 520, 8, 4081, 634, 1078, 1316, 1136, 1606, 1164, 1981, 1543, 128, 4058, 330, 5299, 1139, 1568.

Какие кэш-линии по окончанию не будут в кэше?

Решение второй задачи

Память разбита на куски по 64 последовательных байта, которые мы копируем в кэш. При этом нумерация байтов логичным образом идёт с 0.

Тогда можем узнать, каким кускам принадлежат байты, просто поделив нацело на 64. Получим следующее:

3, 5, 2, 8, 0, 63, 9, 16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82, 17, 24.

Условимся, что наш кэш использует алгоритм кэширования LRU (last recently used), то есть при необходимости вытеснения кэш-линии будет вытесняться та кэш-линия, последний запрос к которой был совершён раньше, то есть та кэш-линия, которой мы давно не использовали.

Первые 8 запросов заполнят весь кэш, независимо от того, что было в кэше до этого. Если, например, строчка уже изначально была в кэше, то она так и останется в кэше, но её приоритет на вымещение будет минимальным, так как к ней только совершили запрос, а значит, что до 8 запроса никто его не вытеснит. Если же строчки не было, то мы вытесним какую-то из строчек, которые были изначально, а не какую-либо из новых, как расписано чуть выше. Теперь у нас в кэше строчки 3, 5, 2, 8, 0, 63, 9, 16.

До предпоследнего запроса нет такого запроса, который бы совпадал с каким-нибудь запросом, которые были не больше 8 запросов раньше, а это означает, что мы каждый раз будем записывать новую строчку и выкидывать давно не использованную. Замечу, что у нас есть по два запроса к 2 строчке, к 17 строчке и к 63 строчке, но для них разница между номером запроса больше 8, а это означает, что мы успеем вытеснить эту строчку, пока дождёмся второго запроса к ней. Значит, к этому моменту у нас будут храниться следующие строчки: 18, 30, 24, 2, 63, 5, 82, 17.

Последний запрос происходит к 24 строчке, но 24 строчка на сей раз есть в кэше, а значит, что мы сразу получим к нему доступ и мы никого не вытесним из кэша. Значит, в конце в кэше будут линии 18, 30, 24, 2, 63, 5, 82, 17.

Замечу, что нам неизвестен размер оперативной памяти, поэтому и не понятно, сколько всего возможных кэш-линий, но так как есть запрос к 82 линий, то их как минимум 83. Пусть m – максимальный номер кэш-линии в таком случае. Тогда в кэше отсутствуют линии 1, 3, [6, 16], [19, 23], [25, 29], [31, 62], [64, 81], [83, m]. При этом если m = 82, то последний отрезок [83, m] будем считать пустым.

Ответ: в кэше отсутствуют линии 1, 3, [6, 16], [19, 23], [25, 29], [31, 62], [64, 81], [83, m], где m - максимальный номер кэш-линии.