САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет
по домашней работе №3
«Кэш-память»

Выполнил(а): Василенко Михаил Глебович

Номер ИСУ: 227231

студ. гр. М3134

Санкт-Петербург 2021

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

Теоретическая часть.

Кэш-память/кэш представляет собой быстродействующую память ограниченного объема, которая располагается между регистрами процессора и относительно медленной основной памятью и хранит наиболее часто используемую информацию.

При обращении процессора за командой или данными сначала проверяется их наличие в кэше. Если необходимая информация находится в кэше, она быстро извлекается (кэш-попадание). Если необходимая информация отсутствует (кэш-промах), то она выбирается из основной памяти, передается в процессор и одновременно заносится в кэш.

Для того чтобы уже следующие обращения к кэшу приводили как можно чаще к кэш-попаданиям, передача из оперативной памяти в кэш-память происходит не просто нужными байтами или словами, а кэш линиями (обычно это 64Б). Для того что бы кэш линии не пересекались, адреса берут кратные длине кэш линии.

Помимо самих данных, мы храним и служебную информацию кэш линии: тег и бит актуальности

Рисунок 1 – структура записи в кэше

Рисунок 2 – адрес кэш линии

Пусть кэш линия имеет размер 64Б, тогда все адреса кратны 64, а значит младшие 6 бит – нули, можем про них пока забыть, индекс – номер кэш линии в кэше, если их 512, то индекс занимает 9 бит, тогда тег занимает все остальные биты адреса.

Как определить есть нужные нам данные в кэше или нет? Первый способ: перебрать все кэш линии за линейное время, но это слишком долго, такой подход обесценивает идею кэша. Второй способ: прямая адресация, просто взять и обратиться к данным по индексу, но тут может возникнуть следующая проблема: у 2ух разных адресов могут совпадать индексы, а отличаться только тэги, тогда 2 такие кэш линии будут постоянно вытиснять одна-другую. Компромиссным решением является ассоциативность – разобьем кэш линии на блоки, индекс будет хранить индекс блока. Если у нас 512 кэш линий, а ассоциативность 4, то у нас получится 128 блоков по 4 кэш линии, в каждом из которых мы будем применять линейный поиск.

Кэш прямого отображения – кэш с ассоциативностью 1.

Кэш делится на уровни, обычно на 3, существует и 4 уровень, но он нужен для интегрированной видеокарты.

L1 – кэш первого уровня. Содержит то, что активно используется, но не уместилось в регистры. Обычно делится на кэш для данных L1d и кэш для команд L1i. Обычный объём порядка 32КБ. Скорость доступа примерно 4 такта.

L2 – кэш второго уровня. Обычно общий для данных и для инструкций. Содержит часто используемые данные. Обычный объём порядка 256КБ. Скорость доступа примерно 16 тактов.

L3 – кэш третьего уровня. По скорости доступа ненамного быстрее оперативной памяти, но у него есть свои преимущества. Обычный объём порядка 8МБ. Скорость доступа примерно 60 тактов. Обычно общий для всех ядер. До L3 кэша существенно больше каналов, а значит скорость передачи лучше чем в оперативной памяти, L3 позволяет обмениваться данными между разными ядрами, быстрее чем через оперативную память.

Условие первой задачи

Имеются две системы с кэшами прямого отображения – S1 и S2.

S1 имеет только кэш первого уровня L1, для которого коэффициент попадания составляет 95%, время отклика 4 нс и штраф за промах 100 нс.

S2 имеет двухуровневый кэш. Характеристики L1 аналогичны L1 из S1. L2 имеет время отклика 20 нс, коэффициент промахов 50% и штраф за промах 100 нс.

Нужно определить среднее время обращения к памяти (AMAT) в нс для S1 и S2.

Решение первой задачи

1) S1:

В S1 время отклика почти всегда 4 нс, и очень редко 100 нс, значит среднее время должно быть между 4 нс и 100 нс, ближе к 4 нс. Найдем в скольких процентах случаев у нас будет кэш промах: 100% - 95% = 5%.

Сначала мы посылаем запрос в кэш, за 4 нс получаем ответ, если данные есть в нем, то в память мы не посылаем запрос, а если у нас кэш промах, то мы посыпаем запрос в память, время отклика до которой 100 нс. Мы в любом случае потратим 4 нс, а дополнительные 100 нс только в 5% случаев.

4 + 0.05 * 100 = 9 нс. Как и ожидалось $4 < 9 < 100$, и среднее время отклика ближе к 4.

2) S2:

Если у нас происходит промах в L1, то есть шанс найти данные в L2, скорость отклика которого быстрее, чем у основной памяти, а значит средняя скорость отклика у S2 должна стать еще быстрее чем 9 нс.

Посылаем запрос в L1, тратим на это 4 нс, если кэш промах, то посылаем запрос в L2, а это происходит в 5% случаев, тратим 20 нс на запрос в L2, если кэш промах, то тратим еще 100, а это происходит в 50% случаев.

4 + 0.05 * (20 + 0.5 * 100) = 7.5

Ответ на первую задачу: 9 нс, 7.5 нс.

Условие второй задачи

Имеется кэш с ассоциативностью 4 и размером 8 КБ. Размер кэш-линии составляет 128 байт. Размер основной памяти 1 МБ.

Необходимо определить размер тега адреса.

Решение второй задачи

Объем кэша равен 8КБ = 8192 байт, а одна кэш линия состоит из 128 байт, значит всего кэш линий 8192 / 128 = 64. Ассоциативность кэша равна 4, значит у нас есть 64 / 4 = 16 блоков по 4 кэш линии в каждом.

Тк адрес кэш линии кратен размеру кэш линии, то все адреса кратны 128, а значит младшие 7 разрядов – нули, 4 разряда левее хранят индекс блока, в котором лежит кэш линия, тк групп всего 16, для кодирования 16 номеров достаточно 4 бита. Мы выяснили, что 4 + 7 битов заняты под индекс блока и нули, а значит все остальное занято тегом адреса. Для того что бы найти размер тега адреса найдем размер адресов в нашей системе.

Размер основной памяти 1Мб = 2^20 байт, а для кодирования 2^20 различным адресов нам хватит 20 битов. 20 = размерТега + 4 + 7 => размерТега = 20 – 11 = 9.

Ответ на вторую задачу: 9