САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 3

«Кэш-память»

Выполнил: Рассадников Григорий

Номер ИСУ: 334838

студ. гр. М3138

Санкт-Петербург

2021

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

Теоретическая часть

Вспомним основные сведения известные нам о кэше и зачем он нужен: при попытке чтения или записи в место в основной памяти процессор проверяет, есть ли данные из этого расположения в кэше. Если это так, процессор будет считывать или записывать в кэш вместо гораздо более медленной основной памяти. Данные передаются между памятью и кэшем в блоках фиксированного размера, называемых строками кэша или блоками кэша. При копировании строки кэша из памяти в кэш создается запись кэша. Запись кэша будет включать скопированные данные, а также запрошенное место в памяти (называемое тегом). Когда процессору необходимо прочитать или записать местоположение в памяти, он сначала проверяет наличие соответствующей записи в кэше. Кэш проверяет содержимое запрошенного места памяти во всех строках кэша, которые могут содержать этот адрес. Если процессор обнаруживает, что место в памяти находится в кэше, происходит попадание в кэш (hit rate - HR). Однако если процессор не находит место в памяти в кэше, происходит промахи кэша (miss rate - MR). В случае попадания в кэш-память процессор немедленно считывает или записывает данные в строку кэша. При промахе кэша кэш выделяет новую запись и копирует данные из основной памяти, после чего запрос выполняется из содержимого кэша.

Анализ производительности: чтобы посчитать долю промахов кэша, нужно число промахов разделить на общее число доступов памяти. Тогда чтобы посчитать долю попаданий, нужно из 1 вычесть долю промахов.

Давайте посчитаем среднее время доступа памяти (average memory access time - AMAT): Если известны задержка доступа, частота промахов и штраф за промахи, среднее время доступа к памяти может быть рассчитано с помощью: $t_{cash} + MR_{cash} \cdot MP_{mm}$где t_{cash} — задержка доступа кэша первого уровня, MR_{cash} — это частота пропущенных кэшей первого уровня и MP_{mm} является ли дополнительным циклом, которые требуется пропустить на более высоком уровне для обслуживания по сравнению с ударом на более высоком уровне, и рассчитывается с помощью: $MP_{mm} = t_{mm} + MR_{mm} \cdot MP_{mm2}$ эта формула может быть расширена и использована рекурсивно для всех последующих уровней иерархии памяти, чтобы получить AMAT.

Кэш память с прямым отображением: в этой организации кэша каждое расположение в основной памяти может входить только в одну запись кэша. Поэтому кэш с прямым сопоставлением также можно назвать кэшем «одностороннего набора ассоциативных». Он не имеет политики размещения как таковой, поскольку нет выбора, какое содержимое записи кэша следует выселить. Это означает, что, если два местоположения

сопоставляются с одной и той же записью, они могут постоянно выбивать друг друга. Несмотря на простоту, кэш с прямым сопоставлением должен быть намного больше, чем ассоциативный, чтобы дать сопоставимую производительность, и он более непредсказуем. Пусть x — номер блока в кэше, y — номер блока памяти, а n — количество блоков в кэше, то сопоставление выполняется с помощью уравнения x = y mod n. Пример: Рисунок 1.

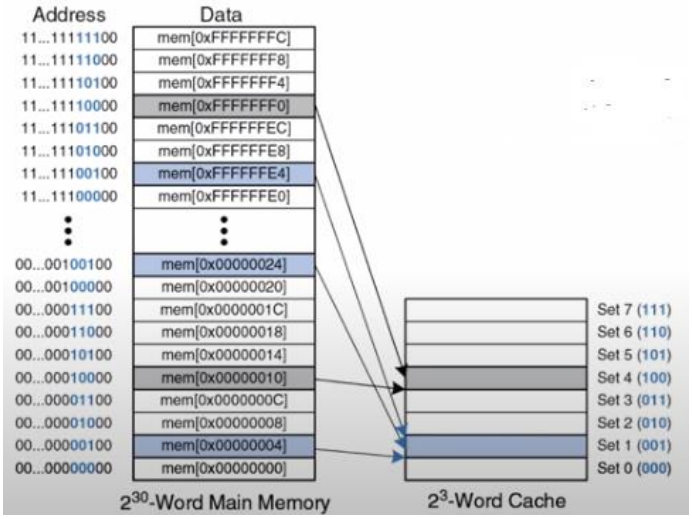

Рисунок 1 — кэш с прямым отображением

Множественно – ассоциативный кэш: кэш, в котором кэш–линии делятся на несколько отдельных блоков. Это позволяет запоминать данные, которые были положены в ячейку кэша не только на последнем hit, но и на более ранних в зависимости от ассоциативности кэша. При увеличении ассоциативности линейно увеличивается глубина нашего запоминания hit. Но при этом возрастает количество ячеек памяти, претендующих на одну кэш-линию. Пример кэша с ассоциативность 2: Рисунок 2.

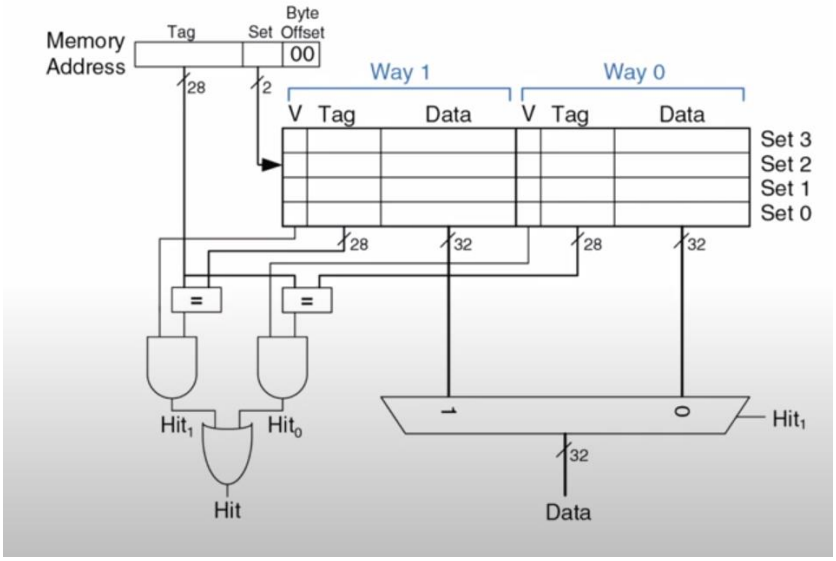

Рисунок 2 — кэш с ассоциативностью 2

Практическая часть

1. Решение задачи (№1).

Условие: имеются две системы, различающихся лишь кэшем.
На первой системе, с временем такта 1.25 нс, установлен кэш с ассоциативностью 2, для которого коэффициент промахов равен 1%.
На второй системе время такта составляет 1.4 нс, а кэш имеет ассоциативность 4.
Определите, какой коэффициент промахов должна иметь вторая система, чтобы иметь меньшее среднее время обращения к памяти (AMAT), чем первая, если время отклика кэша 1 такт, а штраф за промах 75 нс.

Решение: Ассоциативность кэша не влияет на формулу т. к. время отклика систем кэша в задаче одинаковое и равно 1 такт.
Про первый кэш нам известно:

- t_{cash1} = 1.25 нс * число тактов = 1.25 нс * 1 = 1.25 нс
- MR_{cash1} = 1% = 0.01
- MP_{mm} = 75 нс * число тактов = 75 нс
- AMAT1 = $t_{cash1} + MR_{cash1} \cdot MP_{mm}$ = 1.25 + 0.01* 75 = 2 нс

Про второй кэш нам известно:

- t_{cash2} = 1.4 нс * число тактов = 14 нс * 1 = 1.4 нс
- Пусть MR_{cash} = x% = x/100
- MP_{mm} = 75 нс * число тактов = 75 нс
- AMAT2 = $t_{cash2} + MR_{cash2} \cdot MP_{mm}$ = 1.4 + x*75/100 нс

По условию AMAT2 <AMAT1, значит 1.4 + x*75/100 <2, получаем x <0.8.
Ответ: меньше 0.8%.

2. Решение задачи (№9)

Условие: 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: Одна кэш-линия вмещает 64 байта, значит одна кэш линия будет хранить 64 последовательных байта памяти после последнего обращения к кэш-линии. Всего кэш линий 8, значит кэш будет хранить максимально 512 байт памяти. I- я кэш-линия будет хранить всего набор из последовательных 64 байт памяти в диапазоне [((x%512)//64) *64, ((x%512)//64) *64 +63] Заведен таблицу (Таблица 1), в которой будем моделировать работу нашего кэша (какой диапазон памяти хранит каждая кэш-линия). При обработке нового байта, если новый байт входит в диапазон, которых хранит соответствующая кэш-линия, значит сработал hit rate, иначе miss rate.

Кэш-линии Запросы	0	1	2	3	4	5	6	7
204				[192, 255]				
320				[192, 255]		[320, 383]		
181			[128, 191]	[192, 255]		[320, 383]		
520	[512, 575]		[128, 191]	[192, 255]		[320, 383]		
8	[0, 63]		[128, 191]	[192, 255]		[320, 383]		
4081	[0, 63]		[128, 191]	[192, 255]		[320, 383]		[4032, 4095]
634	[0, 63]	[576, 639]	[128, 191]	[192, 255]		[320, 383]		[4032, 4095]
1078	[1024, 1087]	[576, 639]	[128, 191]	[192, 255]		[320, 383]		[4032, 4095]
1316	[1024, 1087]	[576, 639]	[128, 191]	[192, 255]	[1280, 1343]	[320, 383]		[4032, 4095]
1606	[1024, 1087]	[1600, 1663]	[128, 191]	[192, 255]	[1280, 1343]	[320, 383]		[4032, 4095]
1164	[1024, 1087]	[1600, 1663]	[1152, 1215]	[192, 255]	[1280, 1343]	[320, 383]		[4032, 4095]
1981	[1024, 1087]	[1600, 1663]	[1152, 1215]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
1543	[1536, 1599]	[1600, 1663]	[1152, 1215]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
128	[1536, 1599]	[1600, 1663]	[128, 191]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
4058	[1536, 1599]	[1600, 1663]	[128, 191]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
320	[1536, 1599]	[1600, 1663]	[128, 191]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
5299	[1536, 1599]	[1600, 1663]	[5248, 5311]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
1139	[1536, 1599]	[1088, 1151]	[5248, 5311]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]
1568	[1536, 1599]	[1088, 1151]	[5248, 5311]	[192, 255]	[1280, 1343]	[320, 383]	[1920, 1983]	[4032, 4095]

Таблица 1 — запросы подаваемые в кэш

Пометим все hit rate зеленым цветом, miss rate красным. Теперь посмотрим на наше множество байтов из условия задачи, те, которые не входят в

диапазон байтов, содержащийся в соответствующей кэш-линии не будут в кэше. Проверив данные байты получаем: множество {128, 520, 8, 1164, 181, 1078, 1606, 634}. Ответ: 128, 520, 8, 1164, 181, 1078, 1606, 634.