

Основными способами архитектурной организации кэш-памяти является кэш-память с прямым отображением и полностью ассоциативная кэш-память [7]. В кэш-памяти с прямым отображением каждому блоку микрокоманд сопоставляется строго определённая строка кэш-памяти. В полностью ассоциативной кэш-памяти любой блок может быть размещен в любой строке кэша. Ассоциативность предполагает наличие стратегии, по которой будет выбираться замещаемая строка. Стратегия замещения обычно реализуется аппаратно и является важнейшей характеристикой кэш-памяти полностью ассоциативного типа

Анализ стратегии

В качестве стратегии замещения данных предлагается проанализировать стратегию замещения Random, в которой в случае кэш-промаха замещаемый блок выбирается случайным образом [9,10]. При реализации этой стратегии в кэш-контроллере может быть использован генератор случайных чисел, максимально генерируемое значение которого, равно количеству строк в кэш-памяти. Реализации основных стратегий замещения, таких как FIFO, LRU, MRU, LFU, Second Chance и др., предполагает накопление и анализ различного рода статистических данных [11], например о частоте или порядке использования блоков в кэш-памяти. Это требует дополнительных затрат аппаратуры, резко возрастающих с увеличением объёма кэш-памяти, т.к. для накопления и анализа данных необходимо использовать дополнительные блоки регистров (рис. 2).

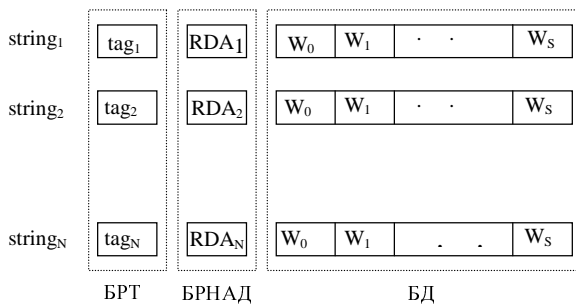


Рисунок 2 - Регистровая структура кэш-памяти полностью ассоциативного типа, реализующая алгоритмы FIFO, LRU, MRU, LFU, Second Chance.

Каждой строке кэш памяти ставится в соответствие отдельный регистр из блока регистров накопления и анализа данных (БРНАД) в котором в зависимости от выбранной стратегии накапливаются и анализируются данные, на основании которых выбирается замещаемая строка

При реализации стратегии Random нет необходимости использовать дополнительные блоки регистров, так как строка кэш памяти замещается при каждом кэш-промахе случайным образом и накопление и анализ каких либо данных не происходит (рис 3).

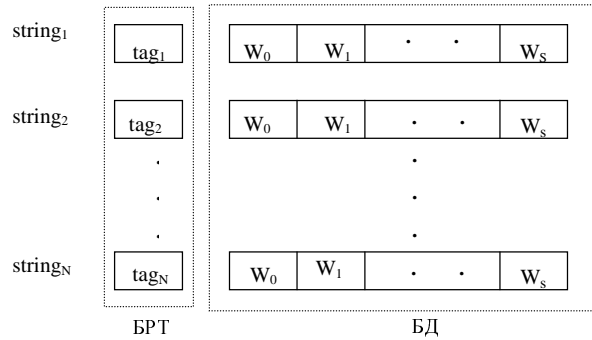


Рисунок 3 - Регистровая структура кэш-памяти полностью ассоциативного типа, реализующая алгоритм Random.

Блок регистров тэгов (БРТ) содержит тэги блоков микрокоманд, находящихся в данный момент в кэш-памяти, а также бит достоверности данных v-bit. Разрядность регистров тэгов на один разряд больше разрядности поля тэга в адресе микрокоманды.

Блок данных (БД) представляет собой модуль статической памяти, состоящий из N строк по S ячеек в каждой строке. Здесь число N не обязательно является целой степенью двойки. Это связано с тем, что в адресе микрокоманды отсутствует поле, определяющее строку кэш-памяти, и эта строка определяется кэш-контроллером. Однако увеличение числа строк кэш-памяти приводит к значительному усложнению схемы кэш-контроллера. Каждая строка блока данных может хранить любой блок микрокоманд из УП.

Применительно к КМУУ последовательность алгоритма замещения Random может быть следующей:

На первом шаге из адреса запрашиваемой микрокоманды выделяется поле тэга.

На втором шаге проверяется, произошло кэш-попадание или кэш-промах. Последовательно для каждой из N строк кэш-памяти проверяется следующее:

- Если содержимое i-й строки достоверно и тэг блока, находящегося в этой строке, совпадает с тэгом в адресе МК, то имеет место кэш-попадание и выполняется «короткий» цикл чтения из i-й строки микрокоманды, соответствующей полю Word в адресе МК.

- В противном случае проверяется следующая строка. Если проверены все строки и совпадение тэгов не обнаружено, то имеет место кэш-промах.

На третьем шаге выполняется последовательная проверка всех строк кэш-памяти на достоверность их содержимого. Для замещения выбирается первая обнаруженная строка с недостоверным содержимым. Этот шаг эффективен лишь до тех пор, пока не будут выполнены первые N кэш-промахов, после чего проверка строк на достоверность будет излишней – все строки будут содержать некоторые блоки микрокоманд из УП. Однако в дальнейшем выполнение этого шага не замедляет работу кэш-контроллера, поскольку фактически шаг 3 выполняется аппаратно одновременно с шагом 4, и имеет лишь логический приоритет в выборе замещаемой строки.

На четвертом шаге кэш-контроллер случайным образом генерирует число от 1 до числа равно количеству строк в кэш памяти. Таким образом, генерируемое число соответствует номеру строки кэш-памяти. Строка с соответствующим номером подлежит замещению.

В общем случае, недостатком случайного выбора замещаемой строки является то, что не учитывается возможная высокая вероятность обращения к замещаемому блоку в ближайшее время. Вероятность кэш-попаданий алгоритма при использовании той или иной стратегии зависит от области использования технологии кэширования.

В таблице 1 показаны результаты имитационного моделирования в среде C++ стратегии Random и LRU (Least Recently Used) в КМУУ с кэш-памятью полностью ассоциативного типа. Исходными данными при определении вероятности кэш попаданий, являются: граф схема реализуемого алгоритма, вероятности переходов по логическим условиям, адресация микрокоманд, число строк и слов в кэш-памяти. Вероятности логических условий считаются известными, потому что эти значения могут зависеть не только от результатов выполнения микрокоманд, но и от внешних факторов (например, от датчиков). Эти значения могут быть получены только опытным путём в процессе функционирования КМУУ, реализующего алгоритм в реальных условиях. Содержимое операционных микрокоманд (ОМК) на этапе определения вероятности кэш-попаданий не имеет значения. Действия, выполняемые в ОМК, либо никак не влияют на процесс выполнения алгоритма, либо влияют на вероятности переходов логических условий, которые уже считаются заданными. Экспериментальный способ определения вероятности кэш попаданий заключается в построении программной модели КМУУ с кэш-памятью с целью сбора статистической информации о количестве кэш-попаданий и кэш-промахов в процессе многократного выполнения алгоритма.

Вероятность кэш попаданий алгоритма определяется как сумма вероятностей кэш-попаданий каждой микрокоманды алгоритма.

$$P_h(G) = \sum_{i=1}^M P_h(a_i)$$

$P_h(a_i)$ - частота возникновения кэш-попаданий каждой микрокоманды алгоритма.

Получены значения вероятности кэш-попаданий для алгоритма размерностью в 20 микрокоманд при разных размерах кэш-памяти. В верхней части ячейки показана вероятность кэш-попаданий алгоритма, используя стратегию Random а в нижней части используя стратегию LRU. При этом N-количество строк в кэш-памяти, S-количество слов в строке кэш-памяти (табл.1). Для каждой размерности кэш-памяти было выполнено 100000 повторений алгоритма. Такое большое количество повторений необходимо для более точного определения количества кэш-попаданий. Это связано с тем, что переходы по ветвям логических условий выбираются каждый раз случайным образом и при каждом проходе алгоритма может выполняться разное количество микрокоманд.

Таблица-1 – Вероятности кэш-попаданий алгоритма при использовании стратегии LRU и Random.

S \ N	1	2	3	4	5	6
1	0,276 0,276	0,310 0,329	0,393 0,393	0,402 0,463	0,426 0,544	0,467 0,631
2	0,276 0,276	0,310 0,329	0,393 0,393	0,402 0,463	0,425 0,544	0,468 0,630
3	0,553 0,553	0,618 0,630	0,671 0,716	0,675 0,821	0,944 0,944	0,944 0,944
4	0,553 0,553	0,618 0,630	0,671 0,716	0,675 0,821	0,944 0,944	0,944 0,944
5	0,674 0,674	0,761 0,809	0,966 0,966	0,966 0,966	0,966 0,966	0,966 0,966
6	0,674 0,674	0,761 0,809	0,966 0,966	0,966 0,966	0,966 0,966	0,966 0,966

Исходя из содержимого таблицы, можно сделать вывод, что значения вероятности кэш-попаданий алгоритма, используя стратегию Random и LRU, зависят от размерности кэш-памяти. При некоторых размерностях эти значения совпадают, а при некоторых отличаются. В таблице выделен диапазон размерностей, при которых значения вероятности кэш попаданий для двух стратегий отличаются. Среднее значения расхождений в пользу LRU внутри этого диапазона составляет 7,25 %.

Заключение

Анализ стратегии случайного замещения данных Random в кэш-памяти КМУУ показывает, что она является экономной с точки зрения затрат оборудования, так как нет необходимости накапливать и анализировать различного рода статистическую информацию, например о частоте или порядке использования блоков в кэш-памяти. Вероятности кэш-попаданий алгоритма при использовании стратегии замещения Random совпадают с вероятностями кэш-попаданий при использовании стратегии LRU при определённых размерностях кэш-памяти. В диапазоне размерностей, где значения кэш-попаданий двух стратегий отличаются, LRU по вероятности кэш-попадания эффективнее, чем Random в среднем на 7,25%. Дальнейшее продолжение работы авторы видят в поиске наиболее оптимальных стратегий замещения, которые бы учитывали всю специфику устройств управления.

Литература

1. Баркалов А.А., Матвиенко А.В. Реализация микропрограммного устройства управления композицией автоматов с жесткой и программируемой логикой // Микропроцессорные средства, разработка и применение. – К.: ИК АН УССР, 1985. – С. 38-42.
2. Шагурин И.И., Бердышев Е.М. Процессоры семейства P6 – Pentium II, Pentium III, Celeron и другие. Архитектура, программирование, интерфейс. – М.: Горячая линия – Телеком. – 2000. – 248 с.
3. Гук М. Процессоры Intel от 8086 до Pentium II. – СПб.: Питер, 1997. – 224с.
4. Hill Mark Donald. Aspects of Cache Memory and Instruction Buffer Performance. – Ph. D. Dissertation, Computer Science Division (EECS), University of California, Berkeley, November 1987.
5. Hennessy J.L., Patterson D.A. Computer Architecture: A Quantitative Approach. – Morgan Kaufmann Publishers, San Mateo, CA, second ed., 1996.
6. Faust B. Designing Alpha-based systems. – Byte Magazine. – June 1995. – P. 239-240.
7. Cheremisinova L. Synthesis of single-level circuits in programmable array logic basis. / In: Computer Aided. – Design of Discrete Devices. VI. – Minsk, 1997. – P. 81-85.
8. Hosseini-Khayat S. On Optimal Replacement of Nonuniform Cache Objects // IEEE Transactions on Computers, Vol. 49, № 8, August 2000. – P. 769-778.
9. Johnson T., Shasha D. A Low Overhead High Performance Buffer Management Replacement Algorithm // VLDB 1994: P. 439-450.
10. Kessler R., Hill M. Page Placement Algorithm for Large Real-Indexed Caches // ACM Trans. Computer Systems. – Nov. 1992, vol. 10, no. 4. – P. 338-359.
11. Tanenbaum, Andrew S. Modern Operating Systems (Second Edition). New Jersey: Prentice-Hall 2001.