МОДЕЛИРОВАНИЕ КЭШ ПАМЯТИ

Петренко А.В.
Кафедра ЭВМ ДонГТУ

Abstract

Petrenko A.V. Simulation of cache memory. *The presented simulation program allows to create and test various models of such important unit of computer, as the cache memory. Some experiments and results are described.*

Одной из основных тенденций в развитии современных ЭВМ является усложнение архитектуры оперативных запоминающих устройств за счет введения различных вариантов кэш-памяти. Сложность получаемых при этом структур и трудности с определением эффективности конкретного варианта кэш-памяти аналитическим путем требуют создания соответствующих моделирующих программ, позволяющих выполнить всесторонние исследования эффективности конкретных технических решений путем проведения серии модельных экспериментов. При этом желательно иметь возможность управлять такими параметрами кэша, как архитектура, размер, алгоритмы работы и пр. с целью выявления наиболее целесообразных вариантов реализации конкретной подсистемы памяти.

Разработанная для данных целей моделирующая программа позволяет как использовать заложенные в программу модели кэшей наиболее распространенных серийно выпускаемых микропроцессоров, так и создавать новые модели кэш памяти.

При создании новой модели пользователь имеет возможность варьировать следующими характеристиками:

- архитектура кэш памяти (с прямым отображением, полностью ассоциативная, наборно-ассоциативная, секторно ассоциативная);
- алгоритм замены строки (случайно, LRU, FIFO, по счетчику);
- принцип определения истинности строки (прямая запись, бит достоверности);
- режим записи (прямая запись, обратная запись);
- размерные характеристики (длина строки, количество строк, количество наборов, размер набора в строках).

Наиболее значимыми факторами, определяющими быстродействие, являются архитектура кэша и алгоритм замены строки [1].

Каждый раз, когда микропроцессору требуется информация, отсутствующая в кэше, он вынужден обращаться через системную шину к основной оперативной памяти. После этого обычно решается, должна ли происходить замена строки в кэш-памяти и какая конкретно строка кэша будет заменена. Далее кратко рассмотрим все заложенные в программу алгоритмы замены строки.

Известны три алгоритма замены строки: LRU, FIFO, случайный. Кроме того рассматривается новый алгоритм с применением счетчика.

Самым простым с точки зрения реализации и аппаратурных затрат является **алгоритм случайной замены строки**, однако эффективность кэша при этом в большинстве случаев ниже, чем при использовании других алгоритмов и существенно зависит от архитектуры и размера кэш-памяти.

Наиболее распространенным в настоящее время [1] является **алгоритм LRU** (Last Recently Used), который обновляет именно ту строку кэша, которая используется менее интенсивно. Новая информация замещает ранее имевшуюся. При этом удаляется в первую очередь информация, отмеченная как недостоверная, а если таковой нет, то чаще всего используется правило удаления из кэш-памяти информации, которая дольше всего оставалась невостребованной. Такой алгоритм обновления называется "замещение наименее используемой информации". В этом случае блок управления кэш-памятью осуществляет контроль использования информации, обеспечивая реализацию этого алгоритма. Отдельные области памяти могут быть определены программным обеспечением или внешней аппаратуры как не подлежащие загрузке во внутренний кэш. В цикле записи при реализации кэш-попадания производится запись по заданному адресу как в соответствующую строку кэш-памяти, так и в адресуемую ячейку основной памяти (сквозная запись). При кэш-промахе запись производится только в основную память и обновление строки кэш-памяти не производится.

Кроме алгоритма LRU может использоваться также **алгоритм замены строки, действующий по принципу FIFO** (First In First Out). Название этого алгоритма говорит само за себя. Для его реализации каждая строка нумеруется уникальным индексом и в случае замены строки заменяется строка с наибольшим индексом. Индекс этой строки обнуляется, а всех остальных строк увеличивается на единицу. Аппаратурные затраты алгоритма FIFO больше, чем при использовании алгоритма LRU, что, однако, компенсируется рядом преимуществ. Алгоритм FIFO, в частности, может применяться при полностью ассоциативной архитектуре построения кэш памяти, в то время как алгоритм LRU может использоваться исключительно для наборно-ассоциативной архитектуры.

Имеется также возможность использования предложенного автором **алгоритма со счетчиком,** являющегося своего рода модификацией алгоритма FIFO, позволяющей уменьшить аппаратные затраты и повысить быстродействие при работе с алгоритмами с большим количеством циклов, т.е. обращений к ячейке памяти с одним адресом. Для построения этого алгоритма используются специальные поля для каждой строки. В такое поле записывается количество обращений к соответствующей строке. Размер этого поля может быть произвольным, но одинаковым для всех строк. При переполнении счетчика значение поля сбрасывается в ноль. Преимущество этого алгоритма над другими заключается в том, что наиболее долго в кэш-памяти остается наиболее интенсивно используемая строка. Однако при этом надо иметь ввиду, что рассмотренный алгоритм будет неэффективен в ряде случаев. Рассмотрим один из них. Предположим, что в начале выполнения программы одна из строк использовалась наиболее интенсивно. Это привело к увеличению значения счетчика для данной строки до некоторого значения m. Далее программа равномерно использует достаточно большое адресное пространство так, что значение счетчика ни в одной из строк не достигает значения m. Как мы видим, строка, использованная вначале, не используется или почти не используется, но и замене она не подлежит, благодаря большому значению счетчика, что равноценно фактическому уменьшению размера кэш-памяти на одну строку. Так постепенно реально используемый размер кэш-памяти может существенно уменьшиться, что приведет к явному уменьшению эффективности кэша.

Программа моделирования кэш памяти

Разработанная программа позволяет создавать до 10 моделей кэш памяти одновременно и проводить их сравнительное исследование, в т.ч. при изменении какого-либо параметра одновременно для всех моделей. При этом могут использоваться как встроенный в программу набор тестов, так и произвольные тестовые программы, создаваемые пользователем. Имеется возможность использования архитектур кэшей наиболее распространенных современных процессоров в качестве эталонов для сравнения (меню "Модель кэша/Стандартные").

В процессе создания новой модели (меню "Модель кэша/Создать новую") пользователь имеет возможность управлять следующими параметрами:

1. название создаваемой модели;
2. архитектура кэш-памяти (с прямым отображением, полностью ассоциативная, наборно-ассоциативная и секторно ассоциативная);
3. тип записи (прямая запись, обратная запись);
4. алгоритм замены строки (LRU, FIFO, счетчик, случайно);
5. определение достоверности строки;
6. количество строк (для архитектуры с прямым отображением и полностью ассоциативной архитектуры);
7. количество наборов (для наборно-ассоциативной и секторно ассоциативной архитектур);
8. размер строки кэш-памяти;
9. размер набора (для наборно-ассоциативной и секторно ассоциативной архитектур).

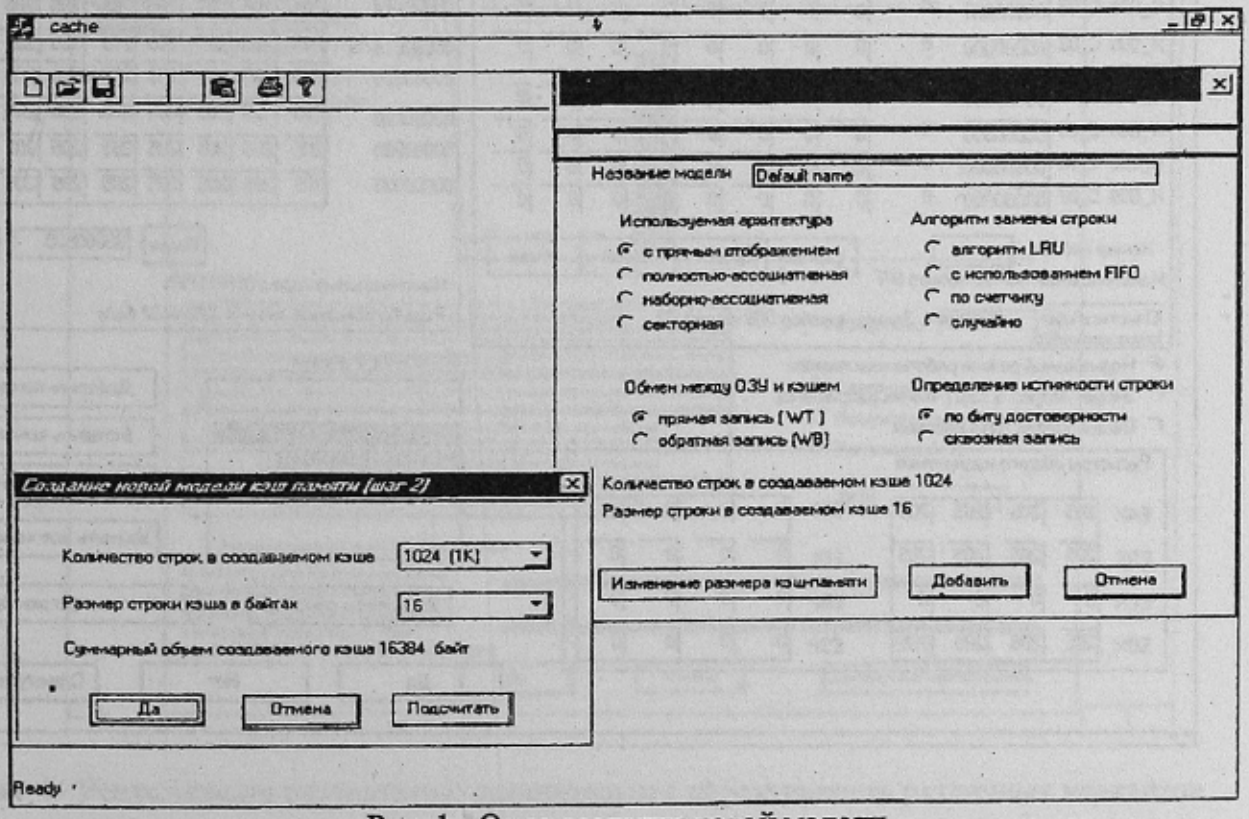

Рис. 1 - Окно создания новой модели

Разработанная программа моделирования была использована для проведения серии экспериментальных исследований. В качестве примеров полученных при этом результатов на рис. 3, 4, 5 и 6 представлены сравнительные характеристики четырех моделей кэш-памяти. Все модели имеют размер поля данных 32KB, реализуется прямая запись с определение истинности строки по биту достоверности и алгоритмом замены строки в кэше по счетчику. Единственное отличие моделей - это архитектура, по которой они простроены:

модель 1 - кэш с прямым отображением;

модель 2 - кэш с полностью ассоциативной архитектурой;

модель 3 - кэш с наборно-ассоциативной архитектурой;

модель 4 - кэш с секторно ассоциативной архитектурой.

Первая диаграмма (рис. 3) характеризует общие аппаратурные затраты при построении данной модели (размер поля данных для всех моделей 32 KB).

Вторая диаграмма (рис. 4) характеризует соотношение эффективности построенных моделей при проведении случайного тестирования.

Графики на рис. 5 и 6 показывают изменение производительности в зависимости от размера строки и количества строк (наборов).

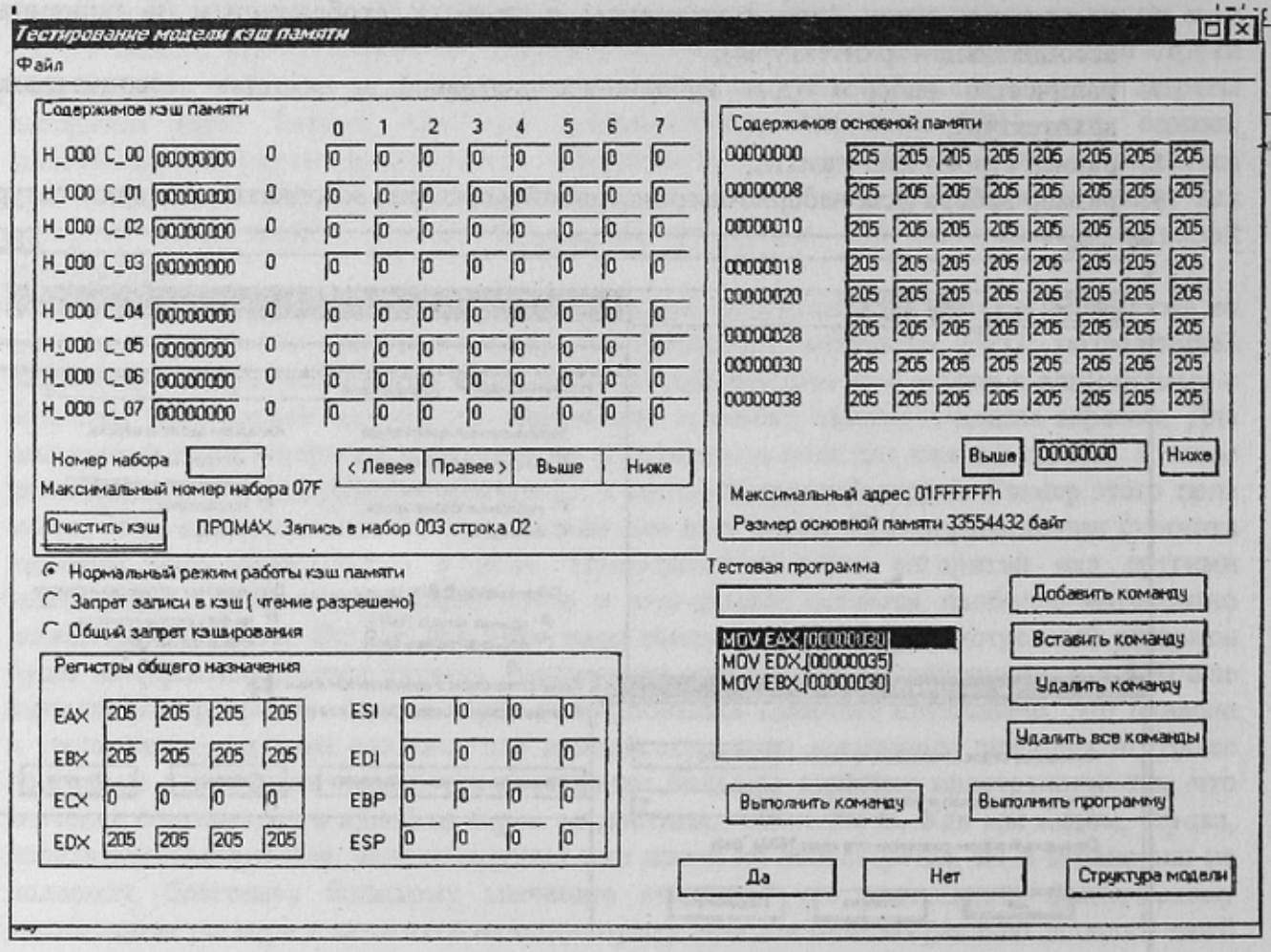

Рис. 2 - Окно тестирования модели

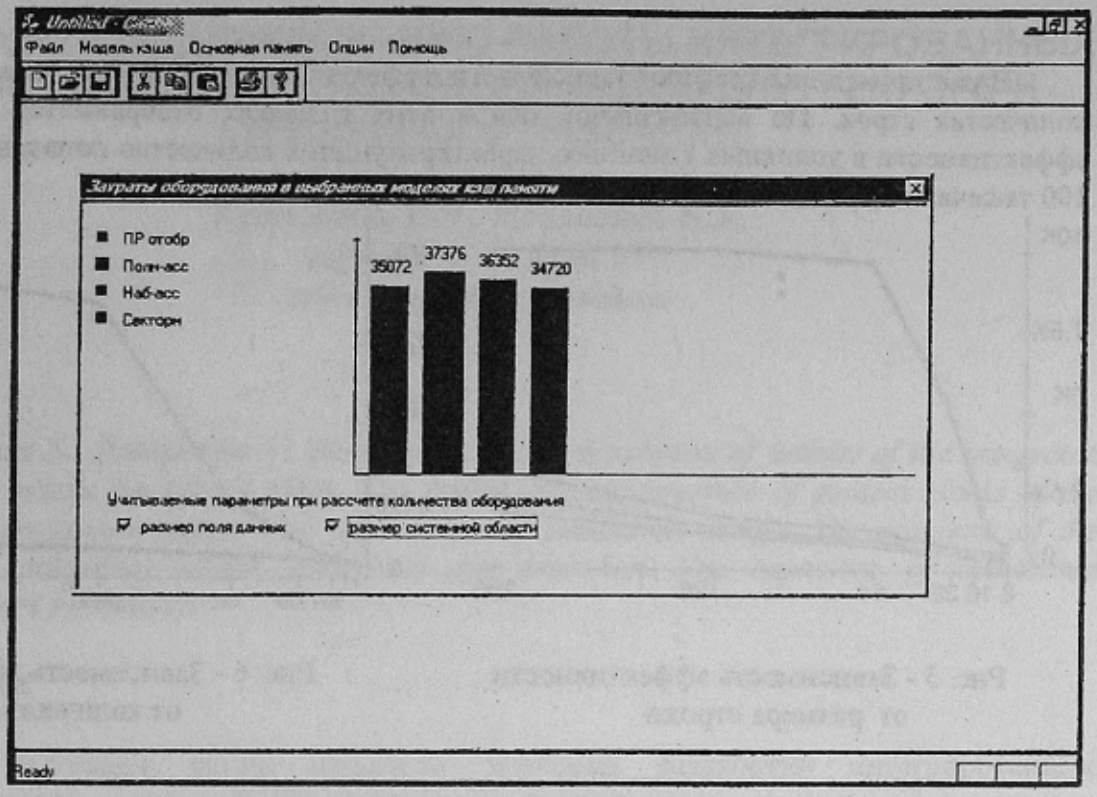

Рис. 3 – Визуализация результатов: общие аппаратурные затраты

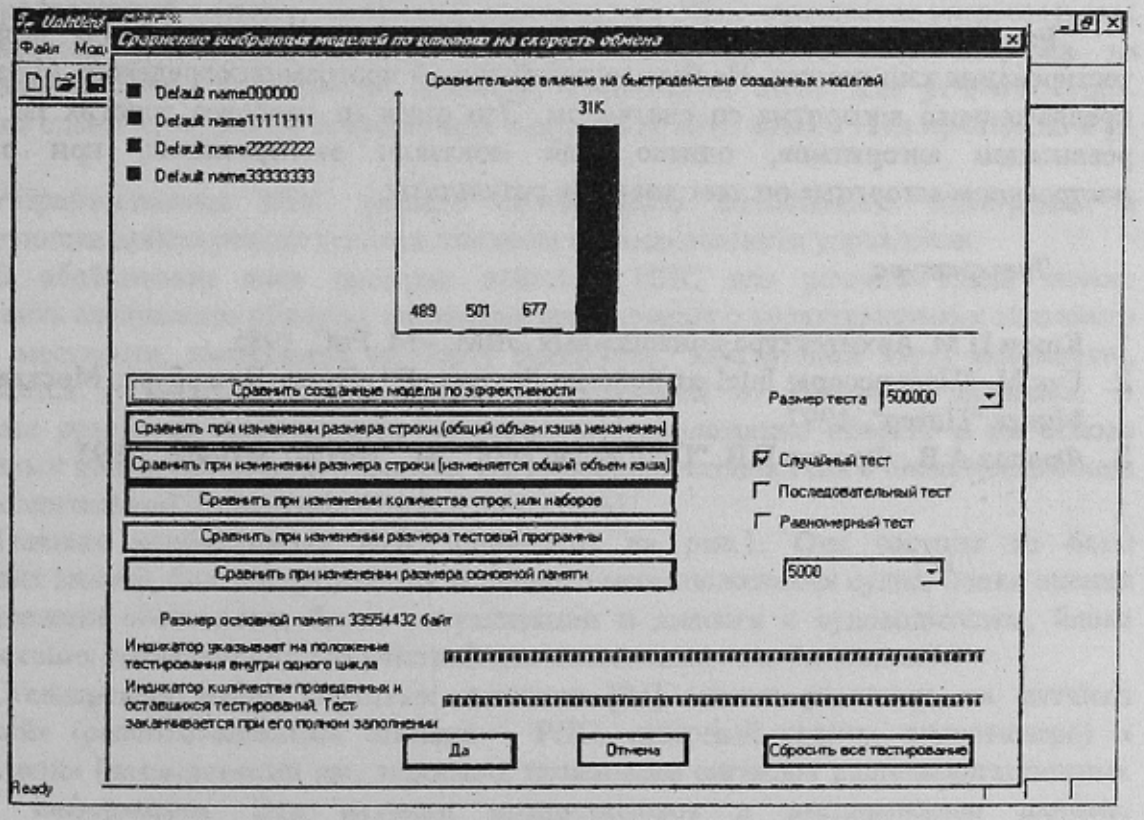

Рис. 4 - Визуализация результатов: сравнительная эффективность различных вариантов кэш-памяти

Ниже приведены графики зависимости эффективности модели от размера строки и количества строк. По вертикальной оси в этих графиках отображается коэффициент эффективности в условных единицах, характеризующих количество попаданий в кэш при 100 тысячах обращений к памяти.

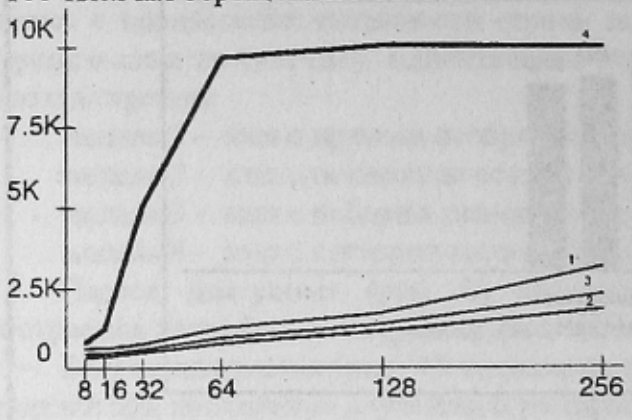

Рис. 5 - Зависимость эффективности от размера строки

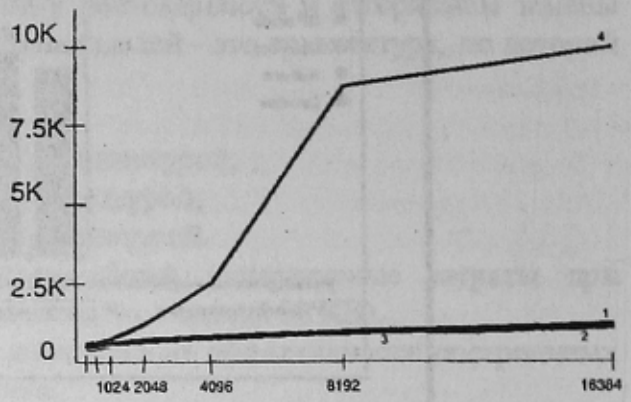

Рис. 6 - Зависимость эффективности от количества строк

Заключение

Разработанная программа является эффективным средством для разработки и тестирования кэш памяти. На базе моделирующей программы определена эффективность предложенного алгоритма со счетчиком. Это один из наиболее дорогих по стоимости реализации алгоритмов, однако, как показали эксперименты, при равномерно построенном алгоритме он дает хорошие результаты.

Литература

1. Коуги П.М. Архитектура конвейерных ЭВМ. - М. РиС, 1985.
2. Гук М. "Процессоры Intel от 8086 до Pentium II", Санкт-Петербург, Москва, Харьков, Минск "Питер", 1997.
3. Фролов А.В., Фролов Г.В. "Процессор i486", М. "Диалог МИФИ", 1995 .