УДК 004.318, 004.254

Оптимизация механизма предварительного считывания в кэш-памяти второго уровня

С.И. Аряшев, К.С. Бычков

ФГУ ФНЦ НИИСИ РАН, bychkov@cs.niisi.ras.ru

***Аннотация* — В статье рассмотрен метод повышения производительности подсистемы памяти суперскалярного RISC-микропроцессора путем модернизации буфера предварительного считывания в составе кэш-памяти второго уровня. Приведены результаты моделирования поведенческой модели микропроцессора, показавшие эффективность предложенных усовершенствований: повышение скорости копирования данных без дополнительных задержек в тракте чтения.**

***Ключевые слова* — микропроцессорная система, подсистема памяти, кэш-память, предвыборка данных, буферы предварительного считывания.**

I. Введение

Подсистема памяти является одним из ключевых элементов, определяющих производительность современных микропроцессорных систем. Зачастую именно подсистема памяти представляет собой «узкое место» системы в плане производительности, что вызвано превышением производительности процессорного ядра процессора над производительностью памяти, а также тем, что число ядер может в несколько раз превышать число каналов памяти [1].

В современных вычислительных системах используется многоуровневая система памяти, построенная по иерархическому принципу. На каждом следующем уровне иерархии увеличивается объем памяти и падает ее быстродействие по сравнению с предыдущим (более высоким) уровнем иерархии.

В стандартных пользовательских программах доля команд загрузки и сохранения, выполняющих обращения к памяти, достигает 30%. Например, основными операциями ОС Linux являются копирование и сохранение-восстановление контекста. Программы, выполняемые арифметическими сопроцессорами, требуют загрузки и выгрузки более значительных объемов данных, поскольку разрядность входных данных для векторных и комплексных сопроцессоров составляет 128 и более бит, кроме того, многие команды, работающие с комплексными и векторными величинами, имеют более двух операндов. Это приводит к ситуации, в которой для обеспечения бесперебойной работы высокопроизводительной системы с суперскалярным процессорным ядром, выполняющим две и более команды за один такт, необходимо обеспечить обращение к подсистеме памяти на каждом такте работы ядра. Это, в свою очередь, приводит к дополнительному увеличению количества запросов к памяти.

Важную роль в обеспечении высокого уровня производительности многоуровневой подсистемы памяти играет организация информационного обмена между различными уровнями иерархии и, в частности, буферизация запросов чтения и записи. Буферизоваться могут запросы как между уровнями иерархии, так и внутри одного уровня. Одним из самых распространенных методов буферизации является предварительная выборка данных, основанная на предсказании данных, которые потребуются памяти более высокого уровня иерархии в ближайшем будущем.

Существуют два подхода к реализации предварительной выборки: программный и аппаратный [1-3]. В первом случае программист или компилятор анализирует запросы к памяти и использует в коде программы специальные команды предварительной выборки данных. В архитектуре MIPS такой командой является команда PREF. Команда проверяет наличие данных по указанному адресу в кэш-памяти и, в случае промаха, считывает данные из памяти. Теоретически программный подход может осуществлять предварительную выборку для любых шаблонов обращений к памяти. На практике ограничение на использование данного метода накладывается необходимостью определения шаблона на стадии компиляции программы. Вследствие этого снижается точность предварительной выборки из-за недостатка информации, доступной только во время выполнения программы.

Аппаратная выборка осуществляется специальными буферами при выполнении программы. Дополнительные команды при этом не используются. В данной статье рассматриваются аппаратные методы предвыборки и пути их усовершенствования, примененные в разрабатываемых в НИИСИ РАН высокопроизводительных RISC-микропроцессорах. Проведено моделирование процессора (использовалась модель на уровне регистровых передач) и измерения скорости копирования массивов данных, показавшие эффективность предлагаемых методов.

II. Подсистема памяти и ее производительность. Кэш-память

Подсистема памяти подавляющего большинства современных высокопроизводительных систем состоит из регистров процессора, многоуровневой кэш-памяти, оперативной (основной) памяти и внешних накопителей [4]. Количество уровней кэш-памяти обычно не превышает трех.

Кэш-память первого уровня (L1-кэш) обычно выполняют раздельной для кодов команд и данных, что обусловлено высокой частотой обращений к командам и данным двух независимых блоков: буфера выборки/выполнения команд и блока загрузки/сохранения регистров.

Кэш-память второго уровня чаще выполняют совмещенной (в ней хранятся и команды, и данные), поскольку частота обращений к ней значительно ниже, а отсутствие разделения обеспечивает более эффективное заполнение кэш-памяти информацией благодаря динамическому изменению соотношения хранимых команд и данных по мере выполнения программы в зависимости от специфики конкретной задачи.

Существуют два подхода к выполнению команд записи применительно к кэш-памяти: сквозная запись (write-through) и обратная запись (write-back). При сквозной записи записываемые данные автоматически передаются на следующий уровень иерархии подсистемы памяти. Применение сквозной записи оправдано для кэш-памятей небольшого объема. С увеличением объема кэш-памяти целесообразно перейти к политике обратной записи, при которой блок информации в который осуществляется запись, помечается как перезаписанный, запись на нижние уровни иерархии передается не автоматически, а только при замещении этого блока информации другим. Это позволяет сократить количество обращений к следующему уровню иерархии.

Важную роль играет ассоциативность кэш-памяти. В кэш-памяти прямого отображения строка с заданным адресом может быть помещена только в одну строку кэш-памяти – это наиболее простое решение, обеспечивающее минимальное время доступа к данным, но интенсивность обмена с внешним уровнем иерархии памяти при этом будет максимальна. В полностью ассоциативной кэш-памяти строка с любым адресом может быть помещена в любую строку кэш-памяти. Это наиболее сложный алгоритм, время доступа максимально, но интенсивность обмена с внешней памятью минимальна. Промежуточным решением является секционировано-ассоциативная организация кэш-памяти, при которой массив кэш-памяти разделен на N секций, и строка с данным адресом может быть помещена в определенную строку любой секции. Число секций N в кэш-памяти современных процессоров различно, но типичные значения N для кэш-памяти первого уровня редко превышают 8, а второго уровня – 16.

В разрабатываемом в НИИСИ РАН перспективном микропроцессоре 1890ВМ8 реализованы секционировано-ассоциативные раздельные кэш-памяти первого уровня команд (объем 32 Кбайт, 8 секций) и данных (объем 16 Кбайт, 4 секции), а также совмещенная кэш-память второго уровня (объем 512 Кбайт, 4 секции). Длина строки всех кэш-памятей составляет 4 двойных слова (256 бит). Быстродействие кэш-памяти первого уровня составляет 1 такт на чтение и 2 такта на запись (для L1-кэша данных), кэш-памяти второго уровня – 4 такта на чтение и 6 тактов на запись.

III. Типы операций в L2-кэше и возможности их ускорения

Для повышения эффективности обмена данными между кэш-памятью первого и второго уровней и оперативной памятью используется механизм буферизации данных.

В кэш-памяти второго уровня разрабатываемого в НИИСИ РАН микропроцессора реализовано четыре типа буферов для различных операций [5]:

- Буфер для записываемых в память L2-кэша данных на 4 строки с возможностью «склейки» (БЗП). Буфер используется при запросах на запись из L1-кэша (что позволяет сразу же начать обработку следующего запроса на загрузку/сохранение), а также при заполнении строки данными из оперативной памяти. В буфере предусмотрена возможность «склейки» запросов по соседним адресам ячеек памяти в пределах одной строки: данные будут размещены в одной ячейке буфера и будут записаны в память за один такт.

- Буфер для данных, передаваемых в оперативную память при обратной записи, на 4 строки (БОЗ).

- Буфер предварительного считывания данных для L2-кэша (БПС) – для ускорения считывания кэшируемых областей памяти применяется механизм предварительной выборки данных. Предварительная выборка заключается в предсказании будущих запросов на чтение и выдаче данных для этих запросов более высоким уровням иерархии памяти до момента поступления запроса. Данный подход значительно уменьшает задержки при обращении к памяти.

- Буфер для сохраняемых данных при некэшируемых обращениях и обращениях, промахнувшихся мимо L2-кэша на 4 двойных слова (БНО).

Схема подключения этих буферов в подсистеме памяти микропроцессора 1890ВМ8 приведена на рис. 1.

Буфер предварительного считывания перед выставлением запроса на чтение по определенному адресу контроллеру оперативной памяти проверяет, не содержатся ли данные по этому адресу в буфере обратной записи (он выполняет функции так называемого «жертвенного» буфера – victim-буфера).

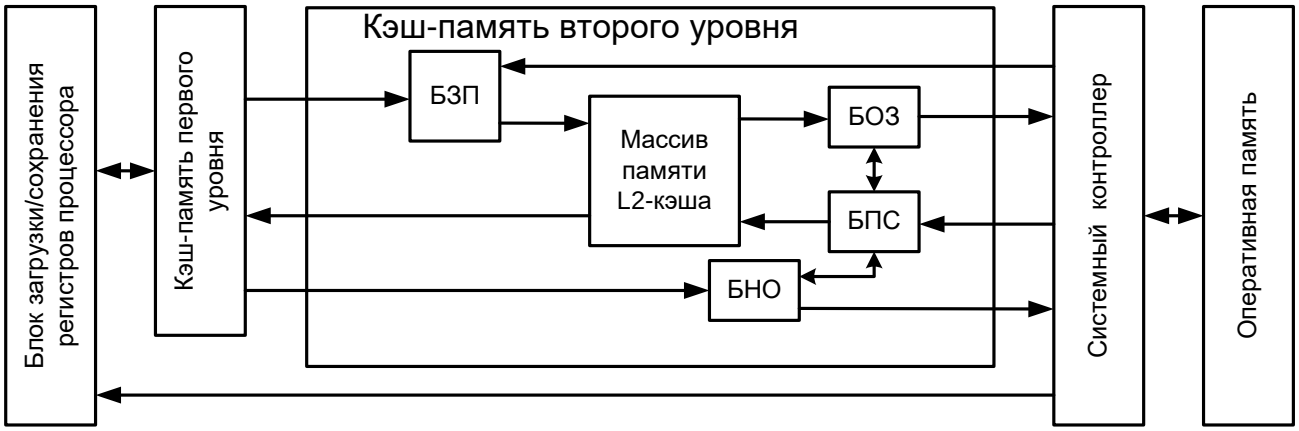

Рис. 1. Функциональная схема подключения буферов в подсистеме памяти микропроцессора 1890ВМ8

Аналогичным образом для поддержания информационной целостности проверяется и содержимое буфера некэшируемых обращений, который может также содержать записи, порожденные командами сохранения с политикой кэширования в обход кэш-памяти второго уровня.

Рассмотрим подробнее буферы предварительного считывания.

IV. Буферы предварительного считывания данных

Буфер предварительного считывания данных в L2-кэш (БПС, см. рис. 1) используется для предварительного вычитывания блоков данных размером в строку кэш-памяти (256 бит) из оперативной памяти. Функциональная схема простейшего БПС приведена на рис. 2.

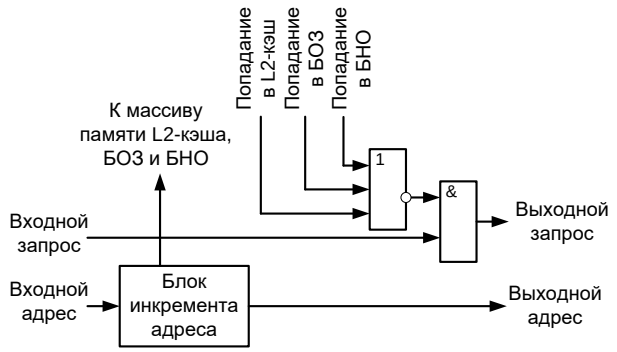

Рис. 2. Функциональная схема буфера предварительного считывания с инкрементацией адресов

При промахе в кэш-память первого уровня инициируется запрос в L2-кэш, и одновременно формируется входной запрос буферу предварительного считывания. Запрошенный L1-кэшем адрес инкрементируется на длину строки, затем проверяется наличие инкрементированного адреса в массиве памяти L2-кэша или в прочих буферах, и при отрицательном результате (отсутствие строки) формируется запрос к системному контроллеру. Данный запрос обладает наименьшим приоритетом по сравнению с запросами, вызванными промахами в кэш-памяти первого уровня, как команд, так и данных.

Функциональная схема предложенного модернизированного БПС приведена на рис. 3.

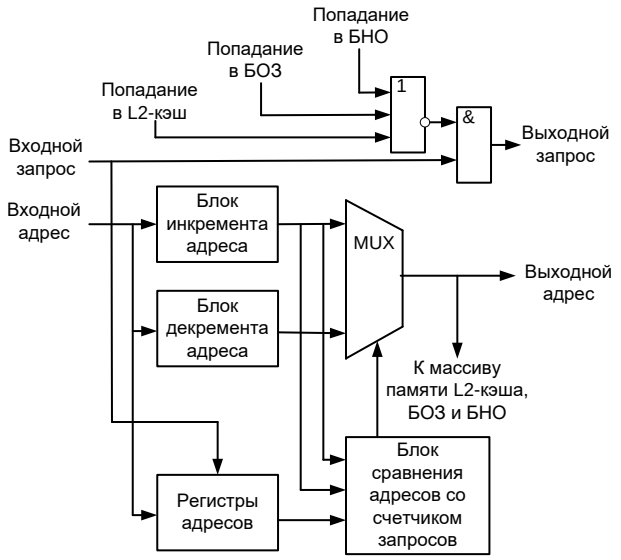

Рис. 3. Функциональная схема модернизированного буфера предварительного считывания с анализом истории запросов

В состав БПС добавляются два регистра, хранящие адреса двух последних запросов к буферу по принципу FIFO. При поступлении входного запроса, формируемого вышеописанным образом, происходит сравнение входного адреса с адресами предыдущих запросов. Если адреса запросов возрастают (т.е. адрес предыдущего запроса совпадает с декрементированным на длину строки входным адресом), то инкрементируется соответствующий «положительный» счетчик запросов. Если же адреса запросов убывают (адрес предыдущего запроса

совпадает с инкрементированным входным адресом), то инкрементируется «отрицательный» счетчик запросов.

Изначально предварительное считывание осуществляется в порядке возрастания адресов, а при значении «отрицательного» счетчика равном двум происходит переключение мультиплексора, и на выход БПС передается декрементированный входной адрес. Соответственно, при предварительном считывании в порядке убывания адресов переключение на инкрементированный адрес осуществляется при достижении значения два в «положительном» счетчике.

Описанная схема не требует значительных аппаратурных затрат и не приводит к появлению дополнительных задержек в тракте выдачи запросов системному контроллеру, что было основным ограничением, наложенным на модернизированный буфер при проектировании.

V. Моделирование и результаты измерений

Эффективность различных методов оптимизации механизма доступа к памяти имеет смысл оценивать, начиная с этапа создания модели будущей микросхемы на уровне регистровых передач (RTL-модели). Необходимость этого обусловлена тем, что на первых этапах разработки механизм может быть реализован не оптимально, и его внедрение не приведёт к положительному эффекту. Кроме того, на этапе создания RTL-модели затраты на модернизацию и моделирование минимальны.

При оценке производительности RTL-модели подсистемы памяти возникают затруднения с использованием существующих программ [6]. Это связано с тем, что предназначенные для запуска на готовых микросхемах тесты могут идти на RTL-модели неприемлемо длительное время. Кроме того, неизбежно задействуются и другие блоки микропроцессора, работа которых может повлиять на результат измерения.

Корректность работы RTL-модели микропроцессора с введенными усовершенствованиями была проверена с помощью запуска баз регрессионных тестов и случайных тестов, а также загрузкой ОС Linux и запуском тестов и приложений пользователя.

Для измерения эффективности буферов предварительного считывания был составлен специальный набор тестов (см. табл. 1).

Тесты 1 и 2 представляют собой части теста LMBENCH, замеряющего скорость копирования данных с использованием библиотеки языка Си GLIBC. Данная библиотека используется для динамически скомпонованных программ, она обеспечивает системные вызовы и основные функции (open, malloc, printfb и т.п.).

Тесты 3-6 представляют собой считывание двойными словами блоков данных по 512 байт с возрастанием и убыванием адресов, а также с вытеснением строк из L2-кэша и без оного.

Тест 7 представляет собой чередующиеся серии считывания по восемь строк из памяти, внутри нечетных серий адреса возрастают, внутри четных – убывают. Тест 8 аналогичен тесту 7, но длина серии составляет три строки.

Тесты 9-12 представляют собой считывание простыми словами блоков данных по 512 байт с возрастанием и убыванием адресов, а также с вытеснением строк из L2-кэша и без вытеснения.

Таблица 1

Соотношения скоростей копирования данных

Тест	V1/V0 для БПС1	V1/V0 для БПС2
1. glibc_copy_direct	1,21	1,21
2. glibc_copy_reverse	1,00	1,22
3. ld_64_direct	1,15	1,15
4. ld_64_direct_no_wb	1,41	1,41
5. ld_64_reverse	1,00	1,18
6. ld_64_reverse_no_wb	1,00	1,40
7. ld_comb_long_series	1,02	1,04
8. lw_128_combine	1,00	1,00
9. lw_128_direct	1,55	1,55
10. lw_128_direct_no_wb	1,59	1,59
11. lw_128_reverse	1,00	1,50
12. lw_128_reverse_no_wb	1,00	1,65

Результаты измерений скорости копирования приведены в табл. 1 и в наглядном виде на рис. 4. В таблице и на рисунке под V0 понимается скорость копирования без буфера предварительного считывания, под V1 – с применением буфера, БПС1 – буфер предварительного считывания с инкрементацией адресов, БПС2 – буфер предварительного считывания с анализом истории запросов.

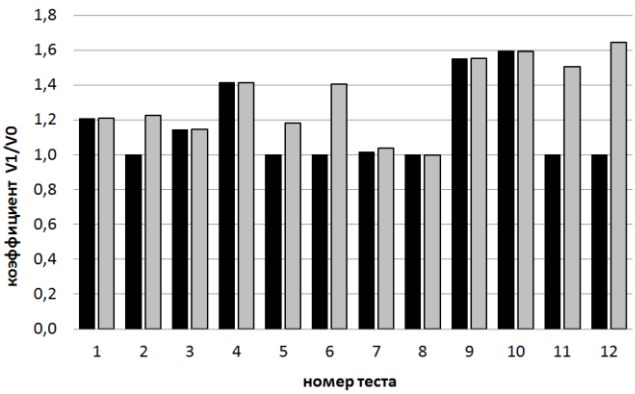

Рис. 4. Относительный выигрыш в скорости копирования при введении в кэш-память второго уровня буферов предварительного считывания

Черные столбцы на рис. 4 относятся к БПС1, а серые – к БПС2.

Выигрыш в скорости копирования при применении предложенного буфера по сравнению с буфером с инкрементацией адресов составляет (при наличии в тесте операций копирования с убыванием адресов) от 20 до 65 процентов. Результат теста 7 подтверждает корректность работы механизма сравнения адресов и счетчиков.

VI. Перспективы развития

В качестве развития буфера предварительного считывания предполагается разделение механизмов предварительной выборки для потоков операций загрузки и сохранения, а также введение механизма для отдельной выборки кодов команд (запросов от кэш-памяти команд первого уровня). Это предположительно приведет к дополнительному росту производительности подсистемы памяти микропроцессора за счет ускорения операций сохранения в кэш-память второго уровня и считывания кодов команд из оперативной памяти.

VII. Выводы

Предложены усовершенствования буфера предварительной загрузки данных в кэш-память второго уровня 1890ВМ8. Проведено моделирование, подтвердившее корректность работы модели процессора и показавшее выигрыш в скорости копирования данных из оперативной памяти в кэш-память от 20 до 65 процентов по сравнению с применением инкрементного буфера. Внедрение модернизированного механизма не привело к дополнительным задержкам в тракте считывания данных.

Литература

[1] Steven P. Vanderwiel, David J. Lilja. Data prefetch mechanisms // Journal ACM Computing Surveys (CSUR). 2000. V. 32. Issue 2. P. 174–199.

[2] Kyle J. Nesbit, James E. Smith. Data Cache Prefetching Using A Global History Buffer. 2005. URL: http://www.eecg.toronto.edu/~steffan/carg/readings/ghb.pdf (дата обращения: 31.03.2016).

[3] S. Palacharla, R.E. Kessler. Evaluating Stream Buffers as a Secondary Cache Replacement. URL: https://www.ece.cmu.edu/~ece548/localcpy/isca94.pdf (дата обращения: 31.03.2016).

[4] Alan J. Smith. Cache Memories // Journal ACM Computing Surveys (CSUR). 1982. V. 14. Issue 3. P. 473–530.

[5] Аряшев С.И., Бобков С.Г., Саяпин П.В. Методика оптимизации и оценки эффективности кэш-памяти второго уровня / Сб. трудов VI Всероссийской научно-технической конференции «Проблемы разработки перспективных микро- и наноэлектронных систем». Ч. 4. М.: Горячая линия – ИППМ РАН. 2014. С. 31–18.

[6] Аряшев С.И., Корниленко А.В., Зубковская Н.В., Саяпин П.В. Повышение производительности подсистемы памяти методом буферизации данных. // Информационные технологии. 2013. № 6. С. 11–17.

Optimizing the Prefetch Mechanism in the Secondary Cache Memory

S.I. Aryashev, K.S. Bychkov

SRISA RAS, bychkov@cs.niisi.ras.ru

***Keywords* — microprocessor system, memory subsystem, cache memory, buffer memory, prefetching, stream buffer.**

Abstract

The memory subsystem is one of the key elements that determine the efficiency of modern microprocessor systems. It is often that the memory subsystem is a “bottleneck” of the system in terms of performance [1-4].

A share of the commands to load and to store data is up to 30% in the standard user programs. Programs, which are performed by arithmetic coprocessors, require loading and storing larger volumes of data. This leads to a situation when, to ensure the smooth operation of the system with high-performance superscalar processor core performing two or more instructions in a single clock cycle, you must provide a reference to the memory subsystem on each pipeline stage.

The organization of information exchange between the different levels of the hierarchy plays the important role for ensuring a high level of memory performance. The important part of such organization of information exchange is the buffering memory requests.

The hardware prefetch techniques used in the high-performance RISC-microprocessor are considered in this article.

Four types of buffers for different operations are implemented in the L2 cache in the developed NIISI RAS microprocessor [5]. One of them is the prefetch data buffer to L2 cache. Prefetch data buffer is used for pre-subtraction data block (cache line = 256 bits) of RAM.

Two registries, keeping the addresses of the last two requests to the buffer according to the FIFO principle, are added to the buffer design.

When an incoming request is received, the input address and the address of the previous request are compared, and the appropriate counter (“up” or “down”) is incremented. Initially, prefetching is performed in ascending order of addresses. If the “down”-counter value is equal to two, then multiplexer switching occurs, and the decrement input address is transferred to the buffer output. Accordingly, switching to the increment address is carried out when the "up"-counter value equal to two.

While the performance of RTL-model memory subsystem is estimated, there are problems with the usage of the existing programs [6]. This happens due to the fact that the tests are intended to run on chips can be executed unacceptably long time at RTL-model. The correctness of the RTL-operation model of the microprocessor with the new buffer was checked by running the database regression testing and random testing, as well as loading Linux operating system and running user application tests.

To measure the effectiveness of the prefetch buffer, a special set of tests built on the basis of LMBENCH test was prepared. Such test set measures the speed of copying data using the C language library GLIBC.

The speed of the operation using the proposed buffer compared to the buffer of increment address (if decreasing addresses copying operations are present in the test) is grown from 20 to 65 percent.

This scheme does not require significant hardware expenses and does not lead to additional delays in the path issuing requests to read from RAM, which were the main constraints imposed on the design of the upgraded buffer.

REFERENCES

[1] Steven P. Vanderwiel, David J. Lilja. Data prefetch mechanisms // Journal ACM Computing Surveys (CSUR). 2000. V. 32. Issue 2. P. 174–199.

[2] Kyle J. Nesbit, James E. Smith. Data Cache Prefetching Using A Global History Buffer. 2005. URL: http://www.eecg.toronto.edu/~steffan/carg/readings/ghb.pdf (дата обращения: 31.03.2016).

[3] S. Palacharla, R.E. Kessler. Evaluating Stream Buffers as a Secondary Cache Replacement. URL: https://www.ece.cmu.edu/~ece548/localcpy/isca94.pdf (дата обращения: 31.03.2016).

[4] Alan J. Smith. Cache Memories // Journal ACM Computing Surveys (CSUR). 1982. V. 14. Issue 3. P. 473–530.

[5] Aryashev S.I., Bobkov S.G., Sayapin P.V. Methodology of the optimization and efficiency evaluation of for the Secondary Cache / Sb. trudov VI Vserossijskoj nauchno-tehnicheskoj konferencii «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem». 2014. V 4. pp. 31–18 (in Russian).

[6] Aryashev S.I., Kornilenko A.V., Zubkovskaya N.V., Sayapin P.V. Improving memory performance by buffering data. // Informacionnye tehnologii. 2013. № 6. pp. 11–17 (in Russian).

.