УДК 004 DOI: 10.17587/it.25.490-501

А. В. Вишнеков, д-р техн. наук, проф., avishnekov@hse.ru,
Е. М. Иванова, канд. техн. наук, доц., emivanova@hse.ru,
К. Э. Басова, магистрант, kezubakina@edu.hse.ru,
Е. О. Ветелина, магистрант, eovetelina@edu.hse.ru,
Национальный исследовательский университет "Высшая школа экономики", Москва

Интерактивный учебно-исследовательский комплекс для моделирования процессов в вычислительных системах

Рассматриваются вопросы моделирования работы функциональных блоков вычислительной системы, требования к моделям с точки зрения решения образовательных и исследовательских задач. Предлагается структура моделирующего комплекса, способы визуализации компонентов архитектуры и принципов функционирования блоков вычислительной системы. Приводятся описания алгоритмов работы моделирующих программ.

Ключевые слова: *моделирование, архитектура вычислительной системы, визуализации компонентов архитектуры, алгоритмы функционирования блоков*

Введение

В соответствии с Программой "Цифровая экономика Российской Федерации" ключевой задачей является подготовка кадров для цифровой экономики, развития прорывных информационных технологий [1]. Решение этой задачи во многом связано с повышением качества подготовки специалистов инженерных специальностей по направлению "Информатика и вычислительная техника". Важнейшим аспектом подготовки специалистов по информационным технологиям является изучение архитектуры современных компьютеров, принципов и алгоритмов функционирования блоков вычислительных систем. Сложность данной задачи заключается в следующем.

Во-первых, предметом изучения в данном случае является такой сложный технический объект, как вычислительная система. В настоящее время существует множество теоретических работ и учебных пособий, раскрывающих принципы построения архитектуры вычислительных систем (ВС) и алгоритмов функционирования их отдельных модулей. Однако учащимся сложно воспринимать большие объемы текстовой информации и на ее основе выстраивать представление об архитектурных особенностях той или иной части ВС и процессе обработки данных в системе. В данном случае компьютерная модель устройства/алгоритма работы будет незаменима. Визуализация сделает любые концепции организации блоков ВС более наглядными, понятными, упростит понимание их работы. Это позволит более эффективно использовать аудиторные занятия преподавателем, существенно сократить время на дополнительную самостоятельную проработку материала учащимися, а также даст возможность обучения в режиме on-line по технологиям e-lealning [2] или blended learning [3, 4], сократив требуемое время и средства для обсуждения и закрепления изучаемого материала.

Во-вторых, для комплексного изучения архитектуры ВС недостаточно теоретических сведений, необходимо использовать наглядные средства, демонстрирующие различные архитектурные принципы и технологии, применяемые при проектировании ВС. Невозможно проследить этапы обработки данных и особенности обработки данных в динамике на реальном компьютере. Многие особенности построения блоков вычислительной системы лежат на микроархитектурном уровне. В этом случае помочь в изучении архитектуры ВС могут компьютерные модели.

В-третьих, в отличие от аналитики на основе таблиц, рисунков или графиков модели-

рование дает понимание динамики процесса, позволяет наблюдать поведение реальной вычислительной системы во времени с необходимым уровнем детализации.

В четвертых, при компьютерном моделировании создается безрисковая среда, позволяющая безопасно и многократно применять и анализировать возможные сценарии декомпозиции компьютера и настройки его составных частей на конкретные режимы работы, которые выполнить в реальности невозможно.

Архитектура ВС, включая функциональную и структурную организацию, — это целый комплекс понятий, принципов взаимодействия программного обеспечения и аппаратной части ВС. Моделировать работу отдельных частей, входящих в понятие "архитектура ВС", аппаратных средств ВС, отдельных алгоритмов их функционирования можно инструментами САПР на языках поведенческого описания устройств (например, SystemVerilog, SystemC) с высокой точностью и степенью детализации модели. Однако, во-первых, подобные средства и модели, как правило, не являются свободно распространяемыми, и, во-вторых, эти средства предназначены для инженеров-проектировщиков ВС, а не для учебных целей, где такая степень детализации не требуется.

Предлагается создать моделирующий комплекс, в структуру которого входят блоки компьютерного моделирования наиболее сложных для понимания устройств/алгоритмов/протоколов функционирования ВС:

- конвейерной обработки потока машинных инструкций;
- способов уменьшения простоев конвейера;
- пошаговой работы ядра суперскалярного процессора;
- протоколов кэш-согласования в многоядерных системах;
- алгоритмов быстрого преобразования адресов с помощью Translation Lookaside Buffer (TLB);
- различных технологий многопоточной обработки данных;
- многоуровневой кэш-памяти;
- алгоритма отображения строк оперативной памяти на строки кэш-памяти;
- алгоритмов замещения строк кэш-памяти и др.

Постановка задачи

Задача заключается в создании моделирующего комплекса, позволяющего исследовать и изучать архитектурные особенности функциональных модулей вычислительной системы. Построить общую компьютерную модель вычислительной системы не представляется возможным в связи с высокой сложностью объекта моделирования. Поэтому целесообразно создать отдельные имитационные модели частей ВС с учетом технологических решений и архитектурных принципов, положенных в основу структуры того или иного блока ВС.

Исходя из основных архитектурных составляющих ядра ВС (процессор и память) следует рассмотреть модели именно этих частей ВС и их особенности. Основные требования, предъявляемые к моделям блоков ВС для решения учебных и исследовательских задач, следующие:

- степень детализации должна позволять решать учебные задачи и проводить исследования модели в целях оценки ресурсно-временных характеристик устройств;
- требуется простая и наглядная демонстрация алгоритмов работы исследуемого блока ВС;
- необходим дружественный интерфейс с контекстной помощью и/или подробное описание шагов моделирования;
- интерактивность модели должна давать возможность пошагового отслеживания поведения блока ВС или изменений состояния модели в динамике;
- должна быть возможность организации режима самостоятельного/удаленного обучения по технологиям e-lealning или blended learning;
- необходима вариативность параметров модели для исследования поведения блоков ВС при различных начальных условиях, а также для обеспечения большого числа различающихся заданий для выполнения каждым студентом собственного варианта;
- время моделирования должно позволить изучить специфику процессов в отведенное по учебному графику время (как правило, два академических часа);
- необходима сопроводительная документация, включающая детальное описание изучаемого блока/алгоритма/принципа.

Описание метода решения задачи

Исходя из основных архитектурных составляющих ядра ВС (процессор и память) следует рассмотреть модели именно этих частей ВС. Рассмотрим особенности современных процессоров. Процессор — это сложное устройство, выполняющее вычисления по заданной пользователем программе и управля-

ющее ходом этих вычислений. Основными частями каждого процессора являются: ядро/ядра, внутренняя кэш-память, блоки управления обменом информацией между процессором и другими устройствами ВС. Поэтому представляется правильным создать модели работы процессорного ядра, модели работы кэш-памяти, модели управления обменом данными процессора и ряда других блоков, таких как оперативная память (ОП). Необходимо разработать модель центрального процессора (CPU) как универсального процессора наиболее общего типа, встречающегося в каждом современном компьютере.

Ядро современного CPU — это достаточно сложная система, и для его моделирования требуется создать комплекс моделей.

Модель 1. Основной механизм, который в той или иной мере реализуется в каждом процессорном ядре для ускорения исполнения команд программы — это конвейерная обработка [5]. Поэтому первая модель, с исследования которой следует начать изучение процессора, будет "Модель конвейерной обработки потока машинных инструкций ядром центрального процессора". Модель "Конвейер ЦП" должна продемонстрировать:

- последовательность действий для исполнения машинной инструкции;
- разбиение машинной инструкции на несколько фаз (для их возможного совмещения для разных инструкций);
- разбиение CPU на несколько ступеней конвейера, исполняющих свою фазу инструкции;
- параллельное во времени исполнение разных фаз разных инструкций;
- потактовую загрузку ступеней конвейера CPU.

Модель 2. Поскольку конвейер команд является достаточно сложным функциональным блоком, то его модель также может получиться весьма сложной для понимания. Однако можно создать для исследования несколько взаимодополняющих моделей конвейера. Таким образом, модель "Конвейера ЦП" (или модификация модели 1) могла бы позволить проследить влияние параметров блоков процессора (ступеней конвейера) на скорость вычислений и загрузку исполнительных блоков и должна продемонстрировать:

- особенности программного кода, вызывающие задержки в вычислениях и простои ступеней конвейера [6, 7];
- способы влияния на ускорение вычислений путем изменения параметров конвейера инструкций;
- возможность накопления статистических данных для сравнения различных технологий и их влияния на ускорение вычислений.

Модель 3. Дальнейшее углубление в исследование принципов работы ядра CPU позволит понять и оценить сложность, необходимость и механизмы реализации принципов суперскалярности в современных процессорах [8—10], сочетающих особенности CISC- и RISC-архитектур и преимущества обоих подходов. Здесь существуют огромные перспективы для исследований. Предлагается остановиться на моделировании базовых механизмов [11] "Пошаговой работы ядра суперскалярного процессора", а именно:

- многофункциональной обработки данных с дублированием и переименованием переменных (регистров и ячеек памяти) [12];
- внеочередного исполнения операций [7, 8];
- предсказания переходов [13, 14] и спекулятивного исполнения программного потока [15].

Модель "Суперскаляр" должна продемонстрировать:

- все перечисленные выше базовые механизмы;
- принцип разбиения процессорного ядра на "front-end"- и "back-end"-кластеры [16—19];
- принцип преобразования потока сложных CISC-инструкций в простые RISC-подобные микрооперации [8, 10, 11, 15];
- ускоренную выборку инструкций из кэш-памяти уровня L1i [20, 21];
- исключение повторной загрузки данных из кэш-памяти уровня L1d [8];
- временную диаграмму загрузки блоков ядра процессора "front-end"- и "back-end"-кластеров;
- конвейерное исполнение инструкций и микроопераций;
- выполнение программы по верному и неверному предсказанию ветви с очисткой конвейера в последнем случае [9];
- возможность отслеживания влияния параметров исполнительного кластера ядра CPU и параметров программного кода на производительность.

Модель 4. Основная технология, позволяющая распараллелить вычисления — технология многопоточной обработки программных потоков ядрами CPU [22, 23]. Поэтому следующей моделью для исследования механизма вычислений, выполняемых процессором, будет "Моделирование различных технологий многопоточной обработки данных". Модель "Виды многопоточности" должна продемонстрировать:

- все способы многопоточной обработки программ;
- пошаговую работу процессора;
- процессорное время, затраченное при каждом способе;
- возможность сравнения всех способов.

Модель 5. Современная вычислительная система имеет сложную иерархически организованную подсистему памяти [5]: самая быстрая сверхоперативная память — внутренняя память процессора, состоящая из регистров/буферов процессорных ядер, кэш-памяти нескольких уровней (L0-L1-L2-L3-L4) [11, 24, 25], оперативной памяти, внешней памяти, реализованной на нескольких разнородных запоминающих устройствах (ЗУ), облачных хранилищах. Модель "Иерархия памяти" должна продемонстрировать:
- место каждого ЗУ в иерархии памяти ВС;
- назначение каждого типа памяти;
- объемы передаваемых данных за однократное обращение к ЗУ;
- латентность и скоростные характеристики всех типов ЗУ;
- способы доступа к данным для каждого типа памяти.

Модель 6. Иерархическая организация подсистемы памяти ВС привела к наличию нескольких типов адресов и адресных пространств. В результате при каждом обращении в память (за инструкциями программы или за данными) требуется задействовать блоки преобразования адресов и служебные таблицы. Таким образом, процесс вычисления конечного физического адреса оперативной памяти может быть достаточно сложной процедурой. Для ускорения переадресации в процессоре существуют блоки TLB [26]. Механизм их работы отражен в следующей модели "Алгоритмов быстрого преобразования адресов с помощью Translation Lookaside Buffer (TLB)". Модель "TLB" должна продемонстрировать:
- назначение и структуру TLB;
- разные случаи текущего состояния памяти (TLB, таблицы страниц);
- пошаговое вычисление физического адреса в разных случаях;
- преимущество использования TLB.

Модель 7. Многоядерные многопоточные процессоры могут выполнять программы, работающие с одними и теми же разделяемыми данными. Одновременное выполнение операций может привести к нарушению когерентности памяти [27]. Существуют протоколы, реализуемые аппаратно кэш-контроллером, которые позволяют избежать неверного изменения общих данных. Реализуются они на последнем уровне иерархии кэш-памяти и демонстрируют различные технологии организации этого уровня. Поэтому следующая модель для исследования механизма вычислений, выполняемых процессором, — это "Моделирование протоколов кэш-согласования в многопроцессорных (многоядерных) системах". Модель "Кэш-протоколы" должна продемонстрировать:
- пошаговую работу каждого из исследуемых протоколов;
- изменения, проводимые в памяти (кэш и ОП);
- последовательность выполняемых действий при считывании, записи и удалении данных из кэш-памяти/ОП;
- возможность сравнения всех протоколов.

Модель 8. Кэш-память процессора строится на ассоциативных запоминающих устройствах, которые слишком сложны и дороги, поэтому существуют разные способы организации размещения (отображения) данных в кэш-памяти, различающихся степенью ассоциативности [11]. Кроме того, при поиске данных в кэш-памяти существует вероятность, что искомые данные там могут быть обнаружены (кэш-попадание) или не обнаружены (кэш-промах). Модель "Отображение строк в кэш-памяти" должна отражать:
- принцип организации кэш-памяти с разной степенью ассоциативности;
- последовательность действий при чтении/записи данных при кэш-попадании и кэш-промахе;
- возможность исследования влияния типа и размера памяти на число кэш-промахов.

Модель 9. Для наращивания емкости используется многоуровневая кэш-память (L1, L2, ...). Существуют различные стратегии организации управления обменом данными между уровнями кэш-памяти. Две наиболее различающиеся: инклюзивная и эксклюзивная. Модель "Многоуровневая кэш-память" должна позволить исследовать:
- влияние параметров кэш-памяти (емкости и латентности каждого уровня) на эффективность кэш-памяти в целом;
- сравнить каждый тип памяти.

Результаты работы моделирующего комплекса

Компьютерная модель 1 "Конвейер ЦП" отображает принцип конвейерной обработки инструкций программного кода для упро-

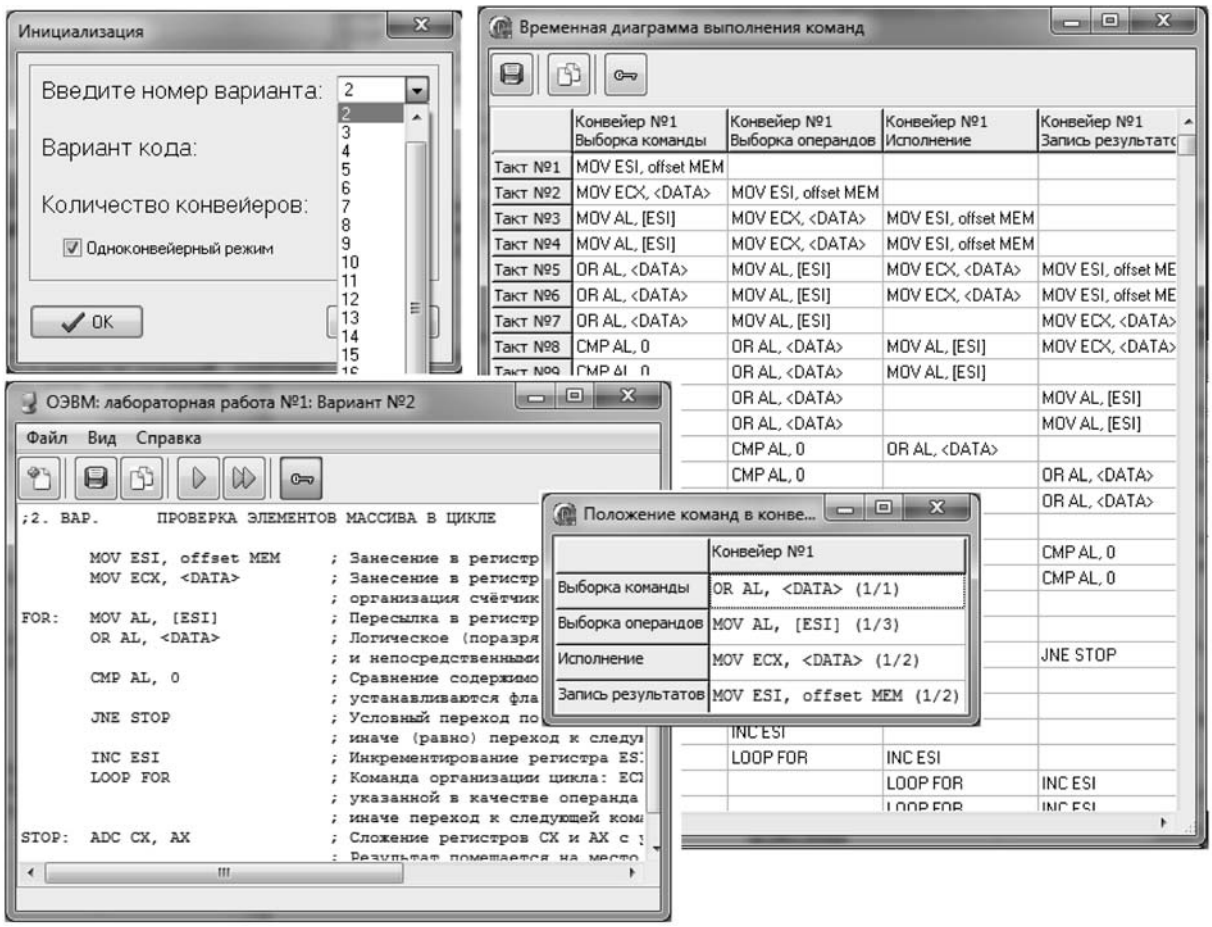

Рис. 1. Интерфейс компьютерной модели конвейера инструкций

щенного варианта процессора с упрощенным конвейером команд, состоящего из четырех ступеней. Моделирующая программа имеет несколько окон (рис. 1):

- главное меню — доступны функции начальных установок, запуска моделирования (по шагам и до конца), сохранения результатов, управления положением окон;
- инициализация — позволяет выбрать вариант модели и число конвейеров в процессоре (один — несколько);
- положение команд в конвейере — показывает при пошаговом моделировании положение команд на ступенях конвейера (в блоках процессора) на каждом такте моделирования выполнения фрагмента программного кода;
- временная диаграмма выполнения команд — отражает полную загрузку ступеней конвейера.

Выполняя моделирование, студенты могут проследить последовательность выполнения инструкций в процессоре, увидеть параллельное выполнение разных фаз разных инструкций блоками CPU, исследовать влияние типов исполняемых инструкций и числа конвейеров на скорость вычислений. Моделирование помогает понять сложность механизма совмещения фаз и увидеть простои блоков CPU, вызванные разной длительностью фаз, зависимостью инструкций по данным или передачей управления. Для облегчения усвоения материала по теме разработано учебное пособие [28].

Модифицированная усложненная компьютерная модель 2 "Конвейера ЦП" (рис. 2) на вкладке "временная диаграмма" отражает аналогичную загрузку блоков процессора (ступеней конвейера), но позволяет исследовать влияние изменений исходных условий моделирования (или сложности организации процессора) на производительность, на загруженность блоков процессора и длительность простоев.

Следующие настройки допустимы:

- отмены повторного чтения операндов;
- уравнивания длительности всех фаз инструкции;
- изменение длины конвейера (числа ступеней);
- дублирования сильно загруженных ступеней конвейера;
- добавление блока предсказания переходов.

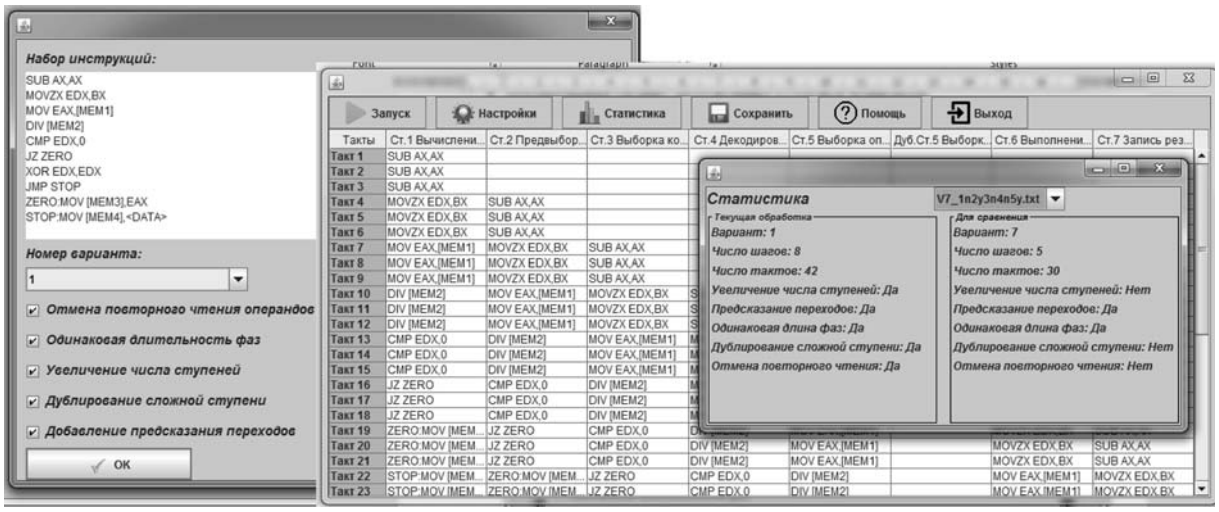

Рис. 2. Интерфейс усложненной компьютерной модели конвейера инструкций

Кроме того, можно проанализировать статистику — сравнительные данные по нескольким проходам программы с разными настройками.

Более сложная модель суперскалярного ядра CPU позволяет изучать и исследовать особенности микроархитектуры ядра современных процессоров Intel, такие как Skylake [8, 29], Zen [8, 30]. Она позволяет выполнить три исследовательские работы и имеет многооконный интерфейс (рис. 3).

В окне "Задача" можно задать параметры процессора и программного кода и проследить влияние изменения этих параметров на загрузку блоков процессора и скорость вычислений. Окно "МОПы" демонстрирует инструкции из моделируемого фрагмента кода и результат их трансляции в микрооперации (мопы) с уче-

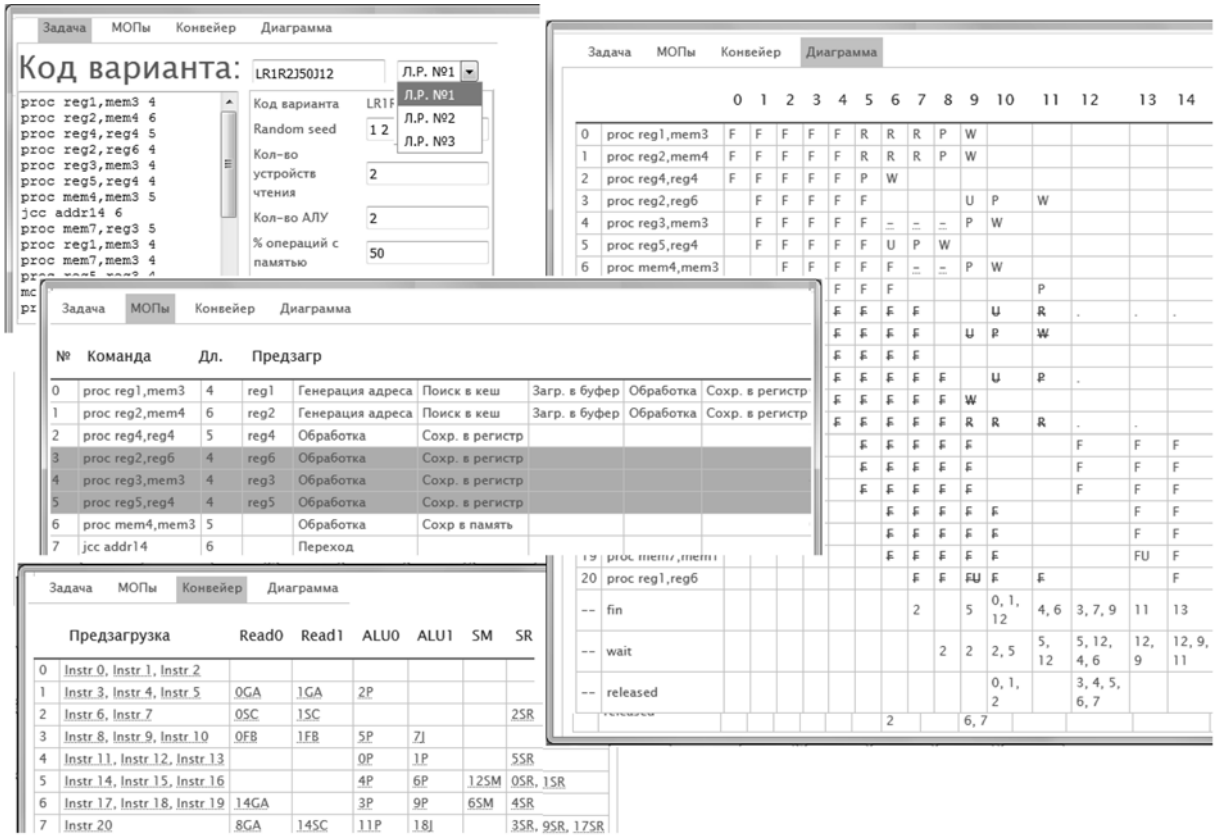

Рис. 3. Окна моделирующей программы "Суперскаляр"

том буферизации данных из ОП/КЭШ. Окно "Конвейер" позволяет увидеть выборку и декодирование набора инструкций во front-end-кластере ядра процессора и проследить динамику загрузки блоков back-end-кластера. Окно "Диаграмма" отражает последовательность выполнения инструкций по тактам и наглядно демонстрирует принцип внеочередного (Out-of-Order) исполнения операций. Также можно увидеть спекулятивное исполнение ветки программы с использованием механизма предсказания переходов (Brunch Predict) с моделированием ситуации неверного предсказания адреса перехода и очисткой конвейера. Микроархитектурные особенности ядра процессора и принципы его функционирования отражены в учебном пособии [31], где также поясняется работа моделирующей программы.

Модель "Виды многопоточности" наглядно демонстрирует принцип исполнения нескольких программных потоков в процессоре одним из выбранных способов [22, 23]: CMT (course-gained multithreading), FMT (fine-gained multithreading), SMT (simultaneous multithreading) и дает возможность их сравнения (рис. 4).

Для заданного варианта кодов можно проследить переключения между потоками и сравнить процессорное время, затраченное при каждом способе.

Модель "TLB" (рис. 5) демонстрирует алгоритм трансляции виртуального адреса в физический с применением буфера ассоциативной

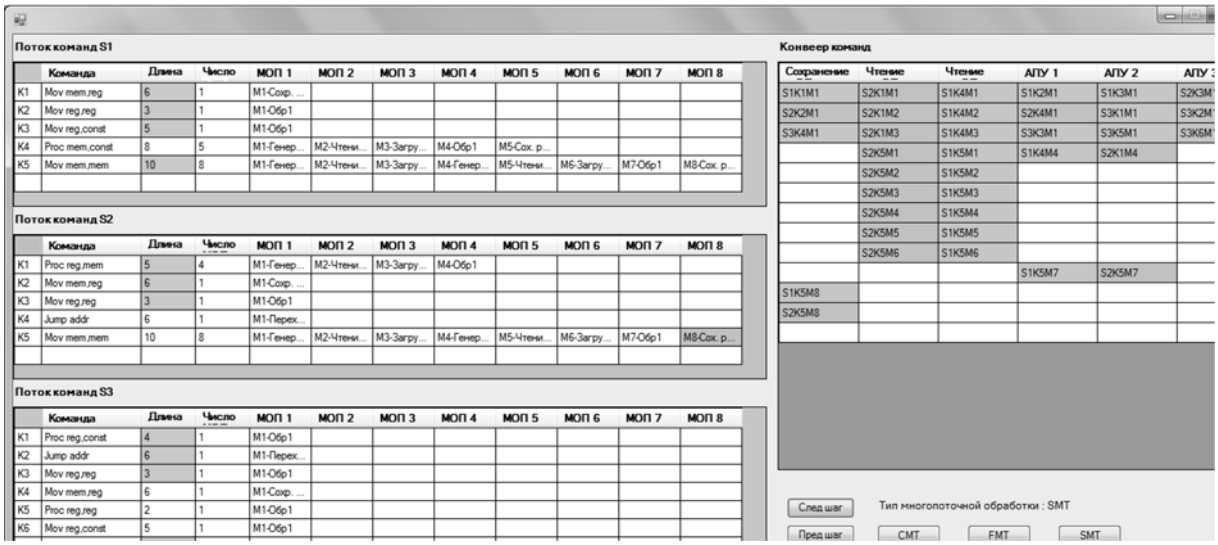

Рис. 4. Окна моделирующей программы "Суперскаляр"

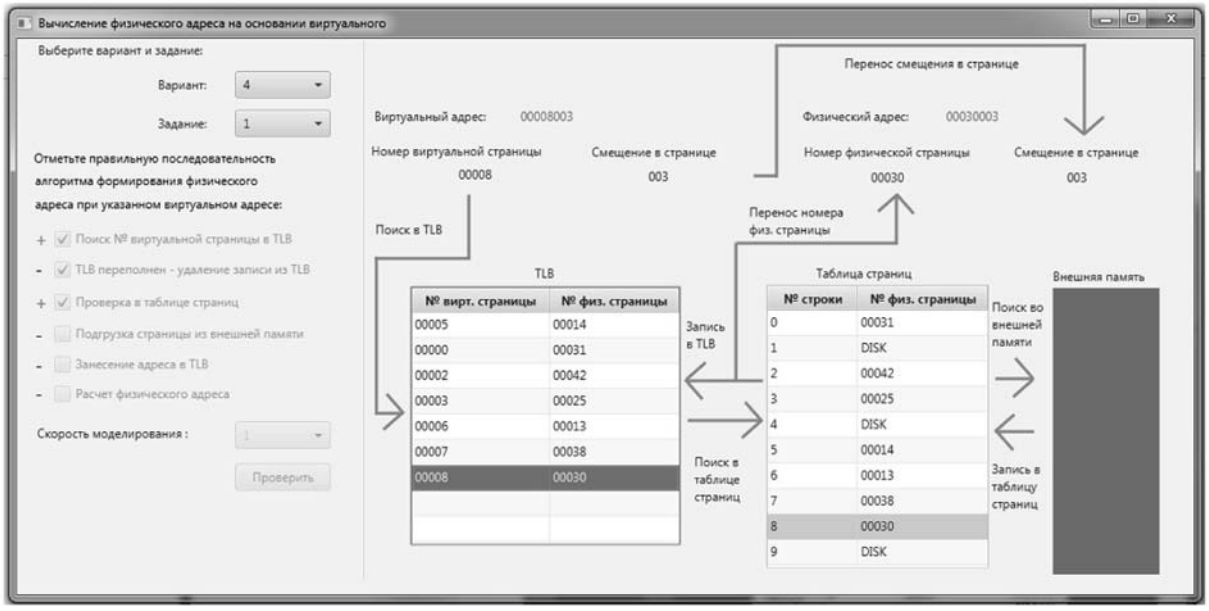

Рис. 5. Окно моделирующей программы после моделирования

трансляции и таблицы страниц. Модель позволяет проанализировать последовательность действий при различных начальных условиях:

- запись присутствует в TLB;
- запись отсутствует в TLB, но есть в таблице страниц;
- запись отсутствует в TLB и в таблице страниц.

Особенности механизма трансляции адресов и работы моделирующей программы отражены в учебном пособии [32].

Модель "Протоколы кэш-согласования" позволяет изучить способ обратной записи (write back) и сравнить несколько протоколов поддержания когерентности кэш-памяти последнего уровня иерархии LLC: MSI, MOSI, MESI, MOESI/MESIF. Интерфейс программы включает несколько окон для отражения строк оперативной памяти и локальной кэш-памяти каждого из четырех процессоров/ядер, а также статусов каждой копии разделяемых данных в кэше (рис. 6). На каждом шаге моделирования в журнале выводится информация о выполненных действиях при реализации того или иного протокола. В результате моделирования можно исследовать скорость работы того или иного протокола (число выполняемых операций). Подробная информация о механизмах поддержания когерентности памяти и руководство по работе с моделирующей программой содержится в работе [33].

Модели алгоритмов замещения строк кэш-памяти (аналогично и для страниц ОП) позволяют проанализировать механизм работы запоминающего устройства, понять причины кэш-промахов, проследить проверку важности строк кэш-памяти, изменения данных и тегов (ссылок на записи) для небольшого числа последовательности обращений в память (рис. 7).

Рассмотрены все основные алгоритмы замещения: RND, LFU, LRU, MRU, FIFO и гибридный алгоритм ARC [34]. Модель демонстрирует для каждого алгоритма особенности его реализации: содержимое строк ОП, состояние кэш (памяти тегов и памяти данных), состояние стека протоколов (временные метки или счетчик обращений).

Модель "Отображение строк в кэш-памяти" позволяет исследовать различные механизмы отображения строк ОП на строки кэш-памяти. Рассмотрены три варианта отображения: прямое, полностью ассоциативное, наборно-ассоциативное (рис. 8). На каждом шаге моделирования можно проанализировать порядок работы блоков кэш-контроллера, последовательность поиска данных с указанным адресом, результаты поиска: кэш-попадание или кэш-промах. В качестве исходных параметров задается число наборов/строк в кэш-памяти, изменяя которые можно исследовать их влияние на эффективность кэш-памяти.

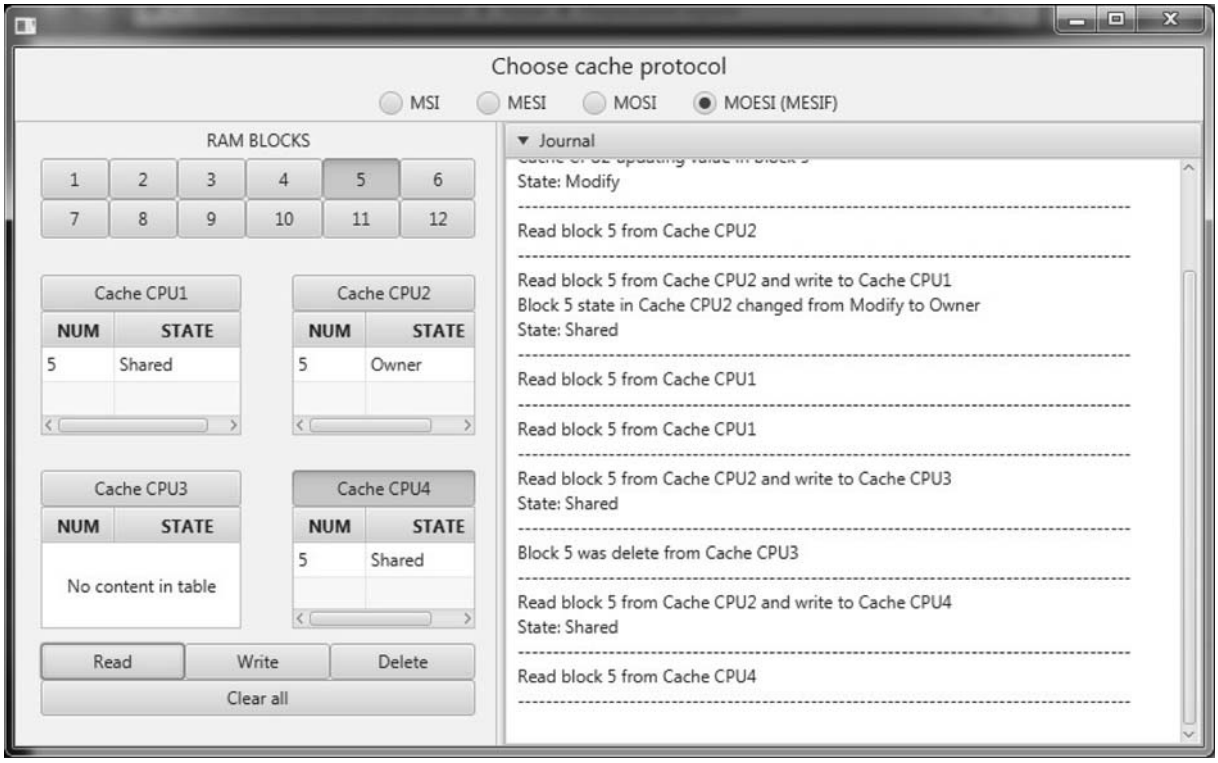

Рис. 6. Интерфейс модели "Протоколы кэш-согласования"

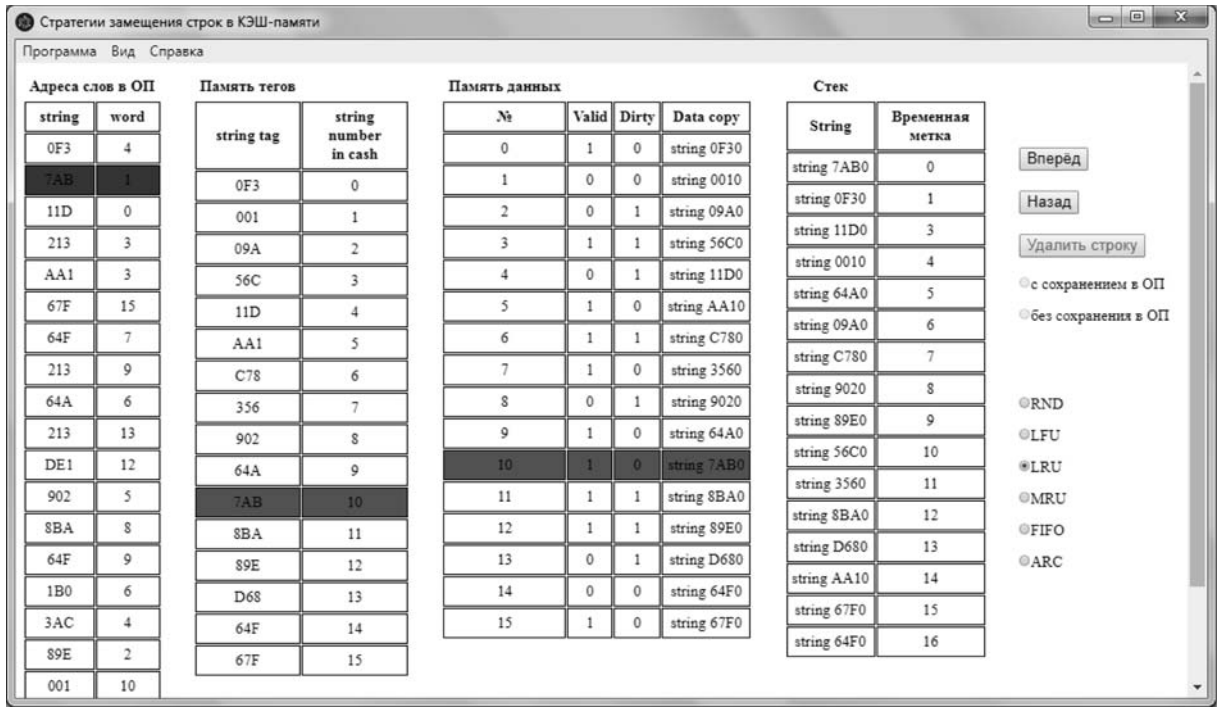

Рис. 7. Интерфейс модели "Алгоритмы замещения строк кэш-памяти"

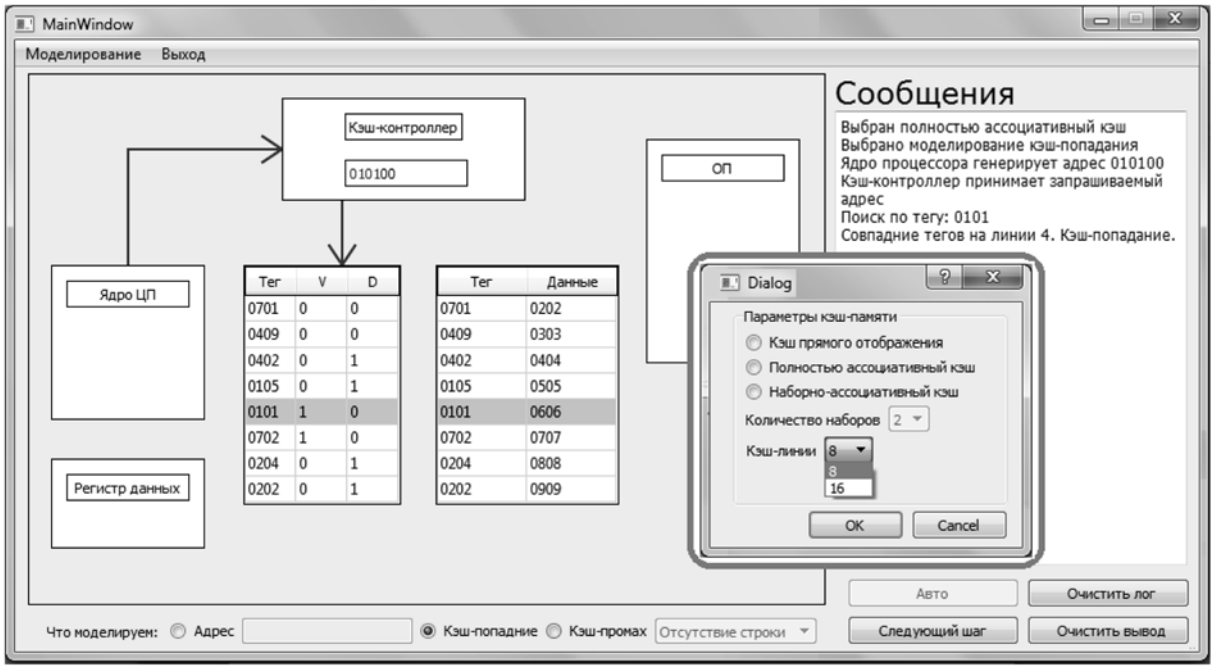

Рис. 8. Интерфейс модели "Отображение строк в кэш-памяти"

Модель "Многоуровневая кэш-память" позволяет сравнить два типа организации многоуровневой кэш-памяти: инклюзивный и эксклюзивный [35]. Меняя параметры моделирования, можно проследить скорость заполнения и процент использования каждого уровня кэш-памяти в обоих случаях, влияние типа кэш-памяти, его размера и числа строк на эффективность кэш-памяти: латентность операций чтения и на число кэш-промахов (рис. 9).

Заключение

Моделирующий комплекс включает необходимое число моделей для изучения архитектуры

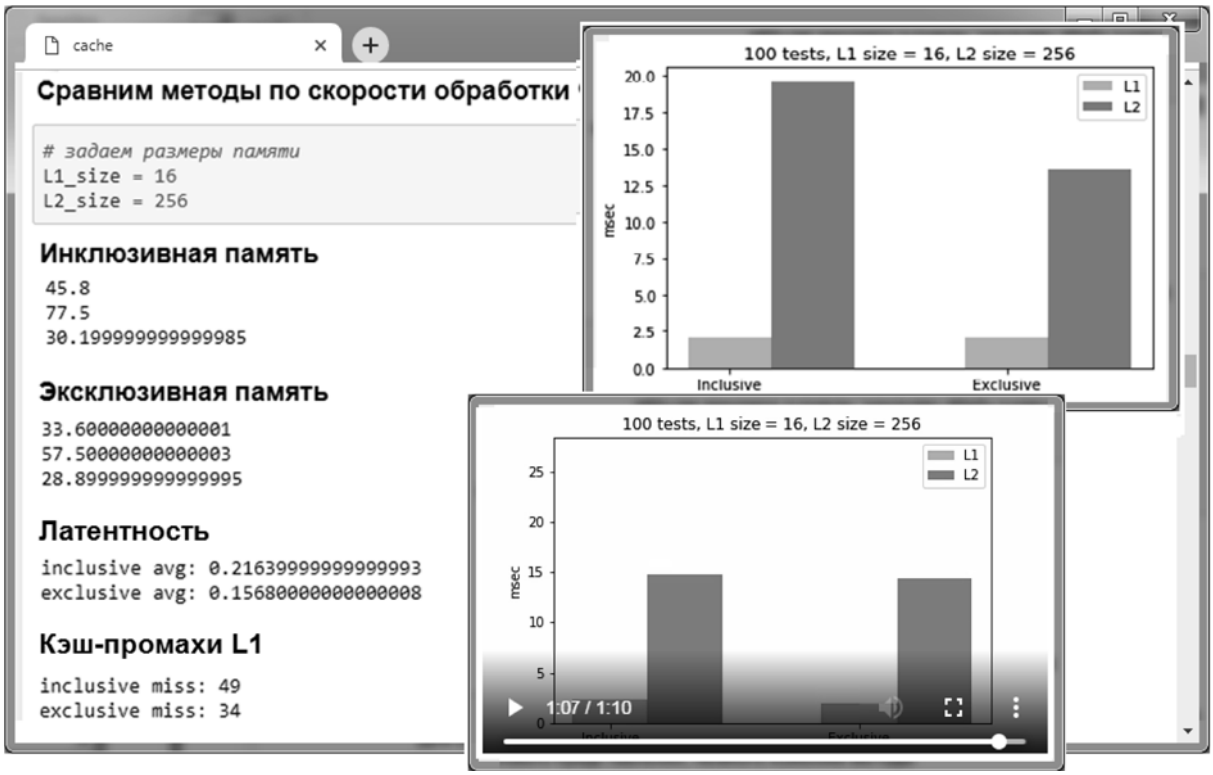

Рис. 9. Интерфейс модели "Многоуровневая кэш-память"

вычислительной системы, позволяет исследовать особенности алгоритмов функционирования отдельных блоков, узлов ВС. Разработанные имитационные модели блоков ВС обладают необходимой степенью абстракции и отвечают критериям наглядности, удобства использования, интерактивности, вариативности параметров. Предлагаемый моделирующий комплекс позволяет педагогам выстроить процесс обучения, эффективно сочетающий в себе элементы очного и on-line образования.

Важным достоинством предлагаемого моделирующего комплекса является возможность вариации параметров моделей отдельных блоков ВС, что дает возможность исследования ресурсно-временных характеристик процессов обработки данных блоками вычислительной системы.

Список литературы

1. **Национальная** программа "Цифровая экономика Российской Федерации". URL: http://government.ru/rugovclassifier/614/events/ (дата обращения 29.03.19).

2. **Association** of e-Learning specialists. URL: http://www.elearningnc.gov (дата обращения 15.03.19).

3. **Garrison D. R., Vaughan N. D.** Blended learning in higher education: Framework, principles, and guidelines. San Francisco, CA: Jossey-Bass, 2008.

4. **Waha B., Davis K.** University students' perspective on blended learning // Journal of Higher Education Policy and Management. 2014. Vol. 36, N. 2. P. 172—182. DOI: 10.1080/1360080X.2014.884677.

5. **Таненбаум Э.** Архитектура компьютера. СПб.: Питер, 2014. 811 с.

6. **Kanter D.** Intel's Sandy Bridge Microarchitecture. Website Real World Tech. 2010. URL: https://www.realworldtech.com/sandy-bridge/10/ (дата обращения 25.02.19).

7. **Intel** Sandy Bridge microarchitecture overview. Out-of-Order Cluster. URL: https://hw-lab.com/intel-sandy-bridge-cpu-microarchitecture.html/4 (дата обращения 13.01.19).

8. **Intel® 64** and IA-32 Architectures Optimization Reference Manual. 2018. URL: https://docslide.net/download/link/intel-64-and-ia-32-architectures-optimization-reference-64-and-ia-32-architectures (дата обращения 10.01.19).

9. **Корогод С.** Новое в архитектуре процессоров Silvermont Intel Atom Z3xxx. 2013. URL: http://www.ixbt.com/portopc/atom-clover5.shtml (дата обращения 28.03.19).

10. **Процессорная** микроархитектура AMD Bulldozer. URL: http://compress.ru/article.aspx?id=21786&iid=995 (дата обращения 15.03.19).

11. **Электронная** энциклопедия процессорных терминов. URL: http://www.ixbt.com/cpu/cpu-pedia.shtml (дата обращения 15.03.19).

12. **Maleen A., Suvinay S., Mark J. C., Joel E., Daniel S.** SAM: Optimizing Multithreaded Cores for Speculative Parallelism // 26th International Conference "Parallel Architectures and Compilation Techniques (PACT)". 2017 P. 64—78.

13. **Coursera.** Branch Prediction Introduction. URL: https://www.coursera.org/lecture/comparch/branch-prediction-introduction-BF58C (дата обращения 29.03.19).

14. **Branch** Prediction Review. URL: https://courses.cs.washington.edu/courses/csep548/06au/lectures/branchPred.pdf (дата обращения 10.02.19).

15. **Intel** Analysis of Speculative Execution Side Channels. White Paper. 2018. URL: https://newsroom.intel.com/wp-content/uploads/sites/11/2018/01/Intel-Analysis-of-Speculative-Execution-Side-Channels.pdf (дата обращения 29.03.19).

16. **Hinton G., Sager D., Upton M.** etc. The Microarchitecture of the Pentium4 Processor. Intel Technology Journal Q1, 2001. URL: http://www.ecs.umass.edu/ece/koren/ece568/papers/Pentium4.pdf_(дата обращения 29.03.19).

17. **Swinburne R.** Feature-Intel Core i7-Nehalem Architecture Dive. bit-tech. 2008. URL: http://www.bit-tech.net/reviews/tech/cpus/intel-core-i7-nehalem-architecture-dive/5/ (дата обращения 28.03.19).

18. **Озеров С.** Архитектура CPU. Computerra. 2005, 37(609). URL: http://www.kinnet.ru/cterra/609/233266.html (дата обращения 28.03.19).

19. **Нечай О.** Подборка статей автора. URL: http://www.libma.ru/ kompyutery_i_internet/cifrovoi_zhurnal_kompyuterra_74/p2.php (дата обращения 28.03.19).

20. **Полувялов А.** В ожидании Willamette — история архитектуры IA-32 и как работают процессоры семейства P6. 2000. URL: https://www.ixbt.com/cpu/pentium4-1.html (дата обращения 28.03.19).

21. **Гарматюк С.** "Новая старая" архитектура Core i7: чего больше — сходства или различий? 2008. URL: https://www.ixbt.com/cpu/intel-ci7-theory.shtml_(дата обращения 28.03.19).

22. **Архитектурные** аспекты параллелизма. URL: http://www.intuit.ru/studies/courses/4447/983/lecture/14921 (дата обращения 15.03.19).

23. **Многонитевая** архитектура микропроцессоров. Открытые системы. URL: https://www.osp.ru/os/2002/01/180943/ (дата обращения 15.03.19).

24. **Intel's** Embedded DRAM: New Era of Cache Memory. URL: https://www.eetimes.com/author.asp?section_id=36&doc_id=1323410 (дата обращения 28.03.19).

25. **Процессоры** Intel Sandy Bridge — все секреты. URL: https://www.ixbt.com/cpu/sandy-bridge.shtml (дата обращения 28.03.19).

26. **Intel 64** and IA-32 Architectures Software Developer's Manual. URL: http://www.intel.ie/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf (дата обращения 28.03.19).

27. **Neupane,** Mahesh. Cache Coherence. URL: https://web.archive.org/web/20100620091706/http://cse.csusb.edu/schubert/tutorials/csci610/w04/MN_Cache_Coherence.pdf (дата обращения 01.03.19).

28. **Иванова Е. М.** Моделирование работы многоконвейерного процессора. М.: МИЭМ, 2011.

29. **Skylake** (client) — Microarchitectures — Intel. URL: https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client) (дата обращения 18.03.19).

30. **Zen** — Microarchitectures — AMD. URL: https://en.wikichip.org/wiki/amd/microarchitectures/zen (дата обращения 21.03.19).

31. **Иванова Е. М.** Модель работы ядра суперскалярного ЦП. М.: НИУ ВШЭ, 2018.

32. **Иванова Е. М.** Быстрое преобразование адресов с помощью Translation Lookaside Buffer (TLB). М.: НИУ ВШЭ, 2018.

33. **Иванова Е. М.** Модель протоколов КЭШ-согласования в многопроцессорных (многоядерных) системах. М.: НИУ ВШЭ, 2018.

34. **Петраков И. Е.** Модели и алгоритмы гибридизации стратегий кэширования // Электронный научный журнал "Программные продукты, системы и алгоритмы". 2016. № 2. DOI: 10.15827/2311-6749.19.187.

35. **Гарматюк С.** Современные десктопные процессоры архитектуры x86: общие принципы работы. URL: https://www.ixbt.com/cpu/x86-cpu-faq-2006.shtml (дата обращения 29.03.19).

A. V. Vishnekov, Ph. D., Professor, e-mail: avishnekov@hse.ru,
E. M. Ivanova, Ph. D., Associate Professor, e-mail: emivanova@hse.ru,
K. E. Basova, master's student, "Computer systems and networks", e-mail: kezubakina@edu.hse.ru,
E. O. Vetelina, master's student, "Computer systems and networks", e-mail: eovetelina@edu.hse.ru,
National Research University Higher School of Economics, Moscow, 123458, Russian Federation

Interactive Educational-Research Complex for Processes Simulation in Computer Systems

This article aims at the improvement of the engineer training quality in the field of "computer Science and engineering" using on-line and blended learning technology. We consider studying of modern computers architecture as an example. The complexity of the modeling object, the multi-faceted process of studying the computer architecture, the limited period — are the factors that determine the difficulty of the problem. A computer simulator allows you to explore the stages and features of the calculation process in dynamics with the necessary detail. It can't be done any other way. Also, computer modeling creates a risk-free environment that allows you to safely and repeatedly apply and analyze possible scenarios, decomposition of computer nodes and configuring its components for different modes of operation. This is either impossible to perform on real devices or may result in computer failure under students' inept actions. The paper discusses the problems of the computer units and devices simulating, the requirements for models in terms of solving educational and research problems. We propose a structure of the simulating complex, visualization methods of computer architecture components and principles of computer system device's operation, algorithms descriptions of the modeling programs. Since the computer architecture implies a whole complex of concepts, including functional and structural organization, software and hardware interaction algorithms, the considered simulating complex consists of several simulators: step-by-step operation of the superscalar processor core; cache coherence protocols in multiprocessor (multi-core) systems; algorithms for fast address translation with the help of Translation Lookaside Buffer (TLB); various technologies for multithreaded data processing; pipeline processing of the machine instructions stream; memory paging algorithms; cache mapping algorithms; cache replacement algorithms; etc. The main functions and features of each model are considered. An important advantage of the proposed simulating complex is the ability to vary the parameters of models of specific computer devices, which makes it possible to explore the resource/time characteristics of data processing by computer system devices.

Keywords: *computer architecture, simulating, visualization of the computer architecture components, algorithms of the computer devices operating*

DOI: 10.17587/it.25.490-501

References

1. **National** program "Digital economy of the Russian Federation", available at: http://government.ru/rugovclassifier/614/events/ (retrieved 29.03.19) (in Russian).

2. **Association** of e-Learning specialists, available at: http://www.elearningnc.gov (retrieved 15.03.19).

3. **Garrison D. R., Vaughan N. D.** Blended learning in higher educa-tion: Framework, principles, and guidelines, San Francisco, CA, Jossey-Bass, 2008.

4. **Waha B., Davis K.** University students' perspective on blended learning. *Journal of Higher Education Policy and Management*, vol. 36, 2014, no. 2, pp. 172—182. DOI: 10.1080/1360080X.2014.884677.

5. **Tanenbaum A. S., Austin T.** Structured Computer Organization, Prentice Hall, 2012.

6. **Kanter D.** Intel's Sandy Bridge Microarchitecture, available at: https://www.realworldtech.com/sandy-bridge/10/ (retrieved 25.02.19).

7. **Intel** Sandy Bridge microarchitecture overview. Out-of-Order Cluster, available at: https://hw-lab.com/intel-sandy-bridge-cpu-microarchitecture.html/4 (retrieved 13.01.19).

8. **Intel®** 64 and IA-32 Architectures Optimization Reference Manual (2018), available at: https://docslide.net/download/link/intel-64-and-ia-32-architectures-optimization-reference-64-and-ia-32-architectures (retrieved 10.01.19).

9. **Korogod S.** New in the Silvermont architecture of Intel Atom Z3xxx Processors, available at: http://www.ixbt.com/portopc/atom-clover5.shtml (retrieved 28.03.19) (in Russian).

10. **Pakhomov S.** AMD Bulldozer processor microarchitecture, available at: http://compress.ru/article.aspx?id=21786&iid=995 (retrieved 15.03.19) (in Russian).

11. **Digital** encyclopedia of processor terms, available at: http://www.ixbt.com/cpu/cpu-pedia.shtml (retrieved 15.03.19) (in Russian).

12. **Maleen A., Suvinay S., Mark J. C., Joel E., Daniel S.** SAM: Optimizing Multithreaded Cores for Speculative Parallelism, *PACT 2009 — 26th International Conference on Parallel Architectures and Compilation Techniques, Raleigh, North Carolina, USA,* pp. 64—78.

13. **Coursera.** Branch Prediction Introduction, available at: https://www.coursera.org/lecture/comparch/branch-prediction-introduction-BF58C (retrieved 29.03.19).

14. **Control** Hazards. Branch Prediction Review, available at: https://courses.cs.washington.edu/courses/csep548/06au/lectures/branchPred.pdf (retrieved 10.02.19).

15. **Intel** Analysis of Speculative Execution Side Channels. White Paper. 2018, available at: https://newsroom.intel.com/wp-content/uploads/sites/11/2018/01/Intel-Analysis-of-Speculative-Execution-Side-Channels.pdf (retrieved 29.03.19).

16. **Hinton G., Sager D., Upton M., Boggs D., Carmean D., Kyker A., Roussel P.** The Microarchitecture of the Pentium4 Processor. *Intel Technology Journal Q1*, available at: http://www.ecs.umass.edu/ece/koren/ece568/papers/Pentium4.pdf (retrieved 29.03.19).

17. **Swinburne R.** Feature-Intel Core i7-Nehalem Architecture Dive. bit-tech, available at: http://www.bit-tech.net/reviews/tech/cpus/intel-core-i7-nehalem-architecture-dive/5/ (retrieved 29.03.19)

18. **Ozerov S.** CPU Architecture. Computerra. 2005, 37(609), available at: http://www.kinnet.ru/cterra/609/233266.html (retrieved 28.03.19) (in Russian).

19. **Nechaj O.** Author's articles compilation, available at: http://www.libma.ru/kompyutery_i_internet/cifrovoi_zhurnal_kompyuterra_74/p2.php (retrieved 28.03.19) (in Russian).

20. **Poluvyalov A.** Waiting for the Willamette — the history of the IA-32 architecture and how do the P6 family processors work, available at: https://www.ixbt.com/cpu/pentium4-1.html (retrieved 28.03.19) (in Russian).

21. **Garmatyuk S.** "New old" core i7 architecture: what's more — similarities or differences?, available at: https://www.ixbt.com/cpu/intel-ci7-theory.shtml (retrieved 28.03.19) (in Russian).

22. **Architectural** aspects of concurrency, available at: http://www.intuit.ru/studies/courses/4447/983/lecture/14921 (retrieved 15.03.19) (in Russian).

23. **Multithreaded** architecture of microprocessors. *Open systems*, available at: https://www.osp.ru/os/2002/01/180943/ (retrieved 15.03.19) (in Russian).

24. **Intel's** Embedded DRAM: New Era of Cache Memory, available at: https://www.eetimes.com/author.asp?section_id=36&doc_id=1323410 (retrieved 28.03.19).

25. **Intel** Sandy Bridge processors — all secrets, available at: https://www.ixbt.com/cpu/sandy-bridge.shtml (retrieved 28.03.19) (in Russian).

26. **Intel** 64 and IA-32 Architectures Software Developer's Manual, available at: http://www.intel.ie/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf (retrieved 28.03.19).

27. **Neupane,** Mahesh. Cache Coherence, available at: https://web.archive.org/web/20100620091706/http://cse.csusb.edu/schubert/tutorials/csci610/w04/MN_Cache_Coherence.pdf (retrieved 01.03.19).

28. **Ivanova E. M.** Simulation of multi-pipeline processor operation, Moscow, MIEM, 2011 (in Russian).

29. **Skylake** (client) — Microarchitectures — Intel, available at: https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client) (retrieved 18.03.19).

30. **Zen** — Microarchitectures — AMD, available at: https://en.wikichip.org/wiki/amd/microarchitectures/zen (retrieved 21.03.19).

31. **Ivanova E. M.** Model of superscalar CPU core operation, Moscow, NRU HSE, 2018 (in Russian).

32. **Ivanova E. M.** Fast address translation using Translation Lookaside Buffer (TLB), Moscow, NRU HSE, 2018 (in Russian).

33. **Ivanova E. M.** Model of cache coherence protocols in a multi-core systems, Moscow, NRU HSE, 2018 (in Russian).

34. **Petrakov I. E.** Models and algorithms of caching strategies hybridization. *Digital scientific journal "Software products, systems and algorithms",* 2016, no. 2, DOI: 10.15827/2311-6749.19.187 (in Russian).

35. **Garmatyuk S.** Modern desktop processors of ×86 architecture: General operation principles, available at: https://www.ixbt.com/cpu/x86-cpu-faq-2006.shtml (retrieved 29.03.19) (in Russian).