Учебное пособие

ПИТЕР

Микропроцессоры и вычислительные комплексы семейства «Эльбрус»

РЕКОМЕНДОВАНО УМО
ПО УНИВЕРСИТЕТСКОМУ ПОЛИТЕХНИЧЕСКОМУ ОБРАЗОВАНИЮ

Авторский коллектив:

ЗАО «МЦСТ» и ОАО «ИНЭУМ им. И. С. Брука»:

В. С. Волин, В. Ю. Волконский, Ф. А. Груздов, А. К. Ким, Ю. В. Морозов, Ю. Н. Парахин, Ю. Х. Сахин, С. В. Семенихин

Московский физико-технический институт:

В. И. Перекатов, В. М. Фельдман

Военно-космическая академия им. А. Ф. Можайского:

С. Г. Ермаков, А. М. Зыков, А. Н. Примаков, Э. М. Халиков

Микропроцессоры и вычислительные комплексы семейства «Эльбрус»

Рекомендовано Учебно-методическим объединением вузов по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 230100 «Информатика и вычислительная техника»

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2013

Ким Александр Киирович, Перекатов Валерий Иванович,
Ермаков Сергей Геннадьевич

Микропроцессоры и вычислительные комплексы семейства «Эльбрус»

Серия «Учебное пособие»

Заведующий редакцией	*А. Кривцов*
Ведущий редактор	*Ю. Сергиенко*
Литературный редактор	*Н. Рощина*
Художник	*Л. Адуевская*
Корректор	*Н. Витько*
Верстка	*Л. Егорова*

ББК 32.973.26я7
УДК 004.382 .7(075)

Ким А. К., Перекатов В. И., Ермаков С. Г.

М59 Микропроцессоры и вычислительные комплексы семейства «Эльбрус». — СПб.: Питер, 2013. — 272 с.: ил.

ISBN 978-5-459-01697-0

В учебном пособии представлены результаты многолетней работы специалистов компании ЗАО «МЦСТ», в последние годы тесно сотрудничающей с ОАО «ИНЭУМ им. И. С. Брука».

Описаны вычислительные средства, спроектированные в составе двух архитектурных линий: SPARC и «Эльбрус». Рассмотрены практически все изделия, которые успешно прошли государственные испытания, рекомендованы к серийному производству и начали выпускаться российской промышленностью.

Рекомендовано Учебно-методическим объединением вузов по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 230100 «Информатика и вычислительная техника».

ISBN 978-5-459-01697-0

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29.

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.

Подписано в печать 02.08.12. Формат 70×100/16. Усл. п. л. 21,930. Тираж 1050. Заказ 0000.

Отпечатано с готовых диапозитивов в ИПК ООО «Ленинградское издательство».
194044, Санкт-Петербург, ул. Менделеевская, 9.

Оглавление

Предисловие

В основе ключевых инновационных направлений, определенных государством, лежат информационные технологии, эффективность которых напрямую зависит от используемой микропроцессорной базы, в первую очередь универсальных высокопроизводительных микропроцессоров. Определяя номенклатуру микропроцессоров для реализации современных технологий, можно сделать ставку на продукцию международных корпораций. Это подход, который применяют во многих странах. В ряде случаев он вполне оправдан и у нас. В то же время есть сферы, где ставка на импорт означает утрату высокого статуса страны в его определяющих критериях.

Прежде всего речь идет о сложных наукоемких стратегических системах, составляющих основу обороноспособности страны. Исключительно важно использование отечественных микропроцессоров также в системах, управляющих объектами базовых отраслей индустрии, таких как железнодорожный транспорт, энергосистемы, разведка и добыча углеводородного сырья.

Сейчас, когда успешно обновляется материальная база нашего образования, можно ставить целью воспитание приобщенных к мировому технологическому прогрессу молодых людей, глубоко осознающих достижения и возможности своей страны, готовых внести свой вклад в ее процветание. И очень важно, чтобы они обучались, получали информацию, пробовали свои способности в технологическом окружении отечественного производства.

В качестве особого проектного направления Комиссия по модернизации и технологическому развитию при Президенте Российской Федерации определила создание суперкомпьютеров для обеспечения основных разработок российской промышленности высокопроизводительными системами моделирования. Возможны и другие направления — очевидные

и гипотетические. Важно, чтобы в нашей стране развитие информационных технологий неизменно опиралось на широкое использование отечественных микропроцессоров и вычислительных средств на их основе. При этом нужно учитывать, что коллективы, способные проектировать современные универсальные микропроцессоры, которые содержат сотни миллионов транзисторов, и базовое программное обеспечение для них, не формируются в одночасье — за рубежом всего несколько широко известных в мире корпораций сейчас ведут такую работу. Приобретя в ней многолетний опыт, специалисты, создающие средства вычислительной техники серии «Эльбрус» в ЗАО «МЦСТ» и ОАО «ИНЭУМ им. И. С. Брука», сочли полезным представить свои результаты в книге, выпуск которой мог бы стать одним из оснований для надежды на успех российских разработок.

В первую очередь книга ориентирована на отечественных специалистов в области компьютеростроения. Сложные проблемы предыдущих десятилетий и повсеместная ставка на импортную продукцию существенно сократили их число. В то же время благодаря серьезным мерам по восстановлению отрасли, принятым государством в последние годы, налицо признаки ее возрождения. Можно рассчитывать на то, что у нас есть категория компетентных читателей, способных воспринять описанные в книге проектные решения, среди которых немало новых. Полагаясь на это, мы детализировали ключевые темы до уровня технического описания, принятого в эльбрусовской проектной практике[1]. Важные фрагменты, которые осложнили бы основной текст книги, вынесены в приложения. Именно таким образом представлены микропроцессоры, на базе которых получила развитие продукция архитектурных линий SPARC и «Эльбрус». Наряду с базовыми микропроцессорами довольно подробно характеризуются созданные на их основе вычислительные комплексы, ставшие представителями этих проектных направлений. При описании дальнейших разработок акцент делается в основном на новых свойствах и параметрах, отличающих данную модель от предшествовавшей.

Материал книги формировался таким образом, чтобы показать в ней практически все изделия, которые успешно прошли государственные испытания, рекомендованы к серийному производству и начали выпускаться российской промышленностью. При этом принималась в расчет категория читателей, для которых большое значение наряду с количественными параметрами вычислительных средств имеют их системные свойства, в частности совместимость

[1] Этим, в частности, объясняется параллельное использование терминов на русском и английском языках.

и масштабируемость. Речь идет о пользователях, рассматривающих описываемую продукцию в контексте развертываемых ими информационно-вычислительных и управляющих центров и объектов.

Твердо надеясь на успех российских микропроцессорных технологий, в развитие которых должно внести свой вклад молодое поколение, авторы книги отнесли к очень значимой категории читателей студентов технических университетов, ориентированных на ее тематику. Она рассматривалась как учебно-методическое пособие, пример реализации основополагающих академических знаний, которые обычно приводятся в учебниках по специальности. Среди нескольких серьезных книг отечественных авторов по микропроцессорам, выпущенных за последние годы, в качестве такого примера можно отметить 3-е издание учебника В. Ф. Мелехина и Е. Г. Павловского «Вычислительные машины, системы и сети» (Academia, 2010). Что касается будущих разработчиков, то возможные проблемы с освоением ими материала нашей книги были в существенной степени сняты при преподавании курсов, которые поставили ведущие специалисты в проектировании техники серии «Эльбрус» на базовой кафедре Московского физико-технического института, действующей в ЗАО «МЦСТ» и ОАО «ИНЭУМ им. И. С. Брука». Большое значение придавалось и обучению будущих пользователей, которым предстоит осваивать эти вычислительные средства в самых ответственных приложениях. В связи с этим для работы над книгой были приглашены ученые Военно-космической академии им. А. Ф. Можайского (Санкт-Петербург), ведущие курсы по вычислительным средствам серии «Эльбрус».

В главе 1 «Общая характеристика семейства “Эльбрус”» описываются история, мотивация и концепции создания отечественных вычислительных средств высшей производительности, над которыми более полувека работают специалисты. Дается сжатая характеристика выпущенной за это время продукции.

В главе 2 «Микропроцессоры с архитектурой SPARC, вычислительные средства на их основе» рассматривается реализация архитектуры SPARC в проекте микропроцессора МЦСТ-R500, который стал базовым элементом описываемой далее серии вычислительных комплексов. Заключительные разделы главы посвящены многоядерным системам на кристалле и процессорным модулям на их основе, характеризующим современное состояние этой архитектурной линии.

В главе 3 «Микропроцессоры с архитектурой “Эльбрус”, вычислительные средства на их основе» аналогичным образом представлены вычислительные

средства, спроектированные на базе оригинальной архитектуры «Эльбрус», созданной ведущими специалистами коллектива. Описанию базового микропроцессора «Эльбрус» предшествует характеристика основных свойств аппаратной составляющей новой архитектуры. Представление систем на кристалле и процессорных модулей завершается гетерогенной микросхемой, интегрирующей универсальные ядра «Эльбрус» и DSP-кластер.

В главе 4 «Общее программное обеспечение вычислительных комплексов семейства “Эльбрус”» приводится состав общего программного обеспечения (ОПО) вычислительных комплексов серии «Эльбрус» и характеризуются его отличительные свойства, связанные с реализацией новой архитектуры.

Состав авторов:

- **ЗАО «МЦСТ» и ОАО «ИНЭУМ им. И. С. Брука»:** В. С. Волин, В. Ю. Волконский, Ф. А. Груздов, А. К. Ким, Ю. В. Морозов, Ю. Н. Парахин, Ю. Х. Сахин, С. В. Семенихин;
- **Московский физико-технический институт:** В. И. Перекатов, В. М. Фельдман;
- **Военно-космическая академия им. А. Ф. Можайского:** С. Г. Ермаков, А. М. Зыков, А. Н. Примаков, Э. М. Халиков.

Авторы выражают благодарность за содействие в написании и оформлении книги Ефремовой О. А., Назарову Л. Н., Тарасенко Л. Г., Шевякову В. С., Шерстневу А. Е., Алейнику В. В., Козлову П. Н., Кругляку Ю. Л., Чистякову С. В.

А. К. Ким, генеральный директор ЗАО «МЦСТ»
и ОАО «ИНЭУМ им. И. С. Брука»,

В. И. Перекатов, заместитель генерального директора ЗАО «МЦСТ»
и ОАО «ИНЭУМ им. И. С. Брука» по науке, заведующий кафедрой
информатики и вычислительной техники МФТИ,

С. Г. Ермаков, начальник кафедры автоматизированных систем управления
Военно-космической академии им. А. Ф. Можайского

Список сокращений

AAU — Array Access Unit — устройство обращения к массивам
ALC — Arithmetic Logic Channel — арифметико-логический канал
ALU — Arithmetic Logic Unit — арифметико-логическое устройство
APB — Array Prefetch Buffer — буфер предварительной подкачки массивов
API — Application Programming Interface — интерфейс программирования приложений
ASI — Address Space Identifier — идентификатор адресного пространства
BOOT PROM — Программируемое постоянное запоминающее устройство (ППЗУ) для хранения программ начального тестирования, инициализации и загрузки операционной системы
CISC — Complex Instruction Set Computer — компьютер с полным набором команд
CPU — Central Processor Unit — центральный процессор
CU — Control Unit — устройство управления
DDR — Double Data Rate — передача данных с двойной скоростью
DMA — Direct Memory Access — прямой доступ к памяти
ECC — Error-correcting code — код коррекции ошибок
ELF — Executable and Linkable Format — формат исполняемого и компонуемого модуля
EPIC — Explicitly parallel instruction computing — вычисления с явным параллелизмом команд
HTML — HyperText Markup Language — язык разметки гипертекста
HTTP — HyperText Transfer Protocol — протокол передачи гипертекста
HTTPS — Hypertext Transfer Protocol Secure — расширение протокола HTTP, поддерживающее шифрование
IB — Instruction Buffer — буфер команд
IDE — Integrated Development Environment — интегрированная среда разработки

IEEE — Institute of Electrical and Electronics Engineers — институт инженеров по электротехнике и электронике

LIFO — Last In First Out — дисциплина обслуживания очереди «последним пришел — первым вышел»

LRU — Least Recently Used — механизм обновления кэш-памяти «наиболее давнее использование»

LVDS — Low-Voltage Differential Signaling — низковольтная дифференциальная передача сигналов

MAU — Memory Access Unit — устройство обмена с оперативной памятью

MBUS — Module BUS — шина соединения высокоскоростных компонентов (ВК) с архитектурой SPARC

MFLOPS — Million Floating Operations Per Second — миллион вещественных операций в секунду

MIPS — Million Instructions Per Second — миллион команд в секунду

MMU — Memory Management Unit — устройство управления памятью

MMX — Multi Media Extension — расширение системы команд для обработки мультимедийных данных

MOESI — modified, owned, exclusive, shared, invalid (модифицированный, собственный, исключительный, разделяемый, незначащий) — протокол поддержки когерентности кэшей микропроцессоров

MPP — Massive Parallel Processing — системы с массовым параллелизмом

NB — North Bridge — системный контроллер «северный мост»

NUMA — Non-Uniform Memory Architecture — архитектура мультипроцессорной системы с несимметричным использованием оперативной памяти (ОП)

PA — Physical Address — физический адрес

PF — Predicate File — предикатный файл

PIC — Position-Independent Code — позиционно независимый код

PLU — Predicate Logic Unit — устройство логических предикатов (синоним PU)

PMC — PCI Mezzanine Card — стандарт PCI-мезонинов IEEE 1386.1

PMU — Pipeline Management Unit — устройство управления конвейером

POSIX — Portable Operating System Interface for UniX — интерфейс переносимых операционных систем UNIX

PT — Page Table — таблица страниц

PTE — Page Table Entry — строка таблицы страниц

PTP — Page Table Pointer — указатель таблицы страниц
RAM — Random Access Memory — память с произвольным доступом
RCU — read — copy — update (чтение — копирование — обновление) — механизм синхронизации, использующий разновидность механизма взаимных исключений
RDMA — remote direct memory access — прямой доступ к оперативному запоминающему устройству (ОЗУ) другого компьютера
RF — Register File — регистровый файл
RISC — Reduced Instruction Set Computer — архитектура компьютера с сокращенным набором команд
SB — South Bridge — периферийный контроллер «южный мост»
SBus — название шины обмена с периферийными устройствами в ВК с архитектурой SPARC
SCSI — Small Computer System Interface — интерфейс для малых вычислительных машин
SDRAM — Synchronous Dynamic Random Access Memory — синхронная динамическая память с произвольным доступом
SMP — Symmetric Multi-Processors — система процессоров с равными правами на использование разделяемых ресурсов
SPARC — Scalable Processor Architecture — масштабируемая процессорная архитектура
SRU — SubRoutine Unit — устройство подпрограмм
TLB — Translation Lookaside Buffer — буфер быстрого преобразования адреса
TLU — Table Look-Aside Unit — устройство обращения к таблице страниц
TU — Trap Unit — устройство обработки прерываний
UUCP — Unix to Unix CoPy — копирование между машинами, управляемыми ОС UNIX
VA — Virtual Address — виртуальный адрес
VLIW — Very Long Instruction Word — широкая команда
VNC — Virtual Network Computing — вычисления через виртуальную сеть
БД — база данных
ВК — вычислительный комплекс
ВС — вычислительная система
КСПП — комплекс сервисных и пользовательских программ
МВК — многопроцессорный вычислительный комплекс

МСВС — Мобильная система Вооруженных Сил
МЦСТ — Московский центр SPARC-технологий
ОПО — общее программное обеспечение
ОС — операционная система
ОСПО — общесистемное программное обеспечение
ПЛИС — программируемая логическая интегральная схема
ППО — прикладное программное обеспечение
ПС — программное средство
СПО — системное или специальное программное обеспечение (в зависимости от контекста использования сокращения)
СТДП — система тестовых и диагностических программ
СУБД — система управления базами данных
ФПО — функциональное программное обеспечение
ЦП — центральный процессор

ГЛАВА 1

Общая характеристика семейства «Эльбрус»

С момента зарождения вычислительной техники в нашей стране проектирование ее высокопроизводительных средств рассматривалось как одна из важнейших целей отечественной науки и технологии. С 90-х годов прошлого века она была связана с принципиально новым условием — необходимостью выполнения новых разработок на базе российских микропроцессоров, соблюсти которое особенно сложно, когда предполагается создание универсальных вычислительных комплексов, ориентированных на решение сложных задач с большим объемом данных.

В настоящее время всего в нескольких странах мира проектируют компьютеры на микропроцессорах собственной разработки — это США, Англия, Япония и Китай. С точки зрения национальных интересов очень важно, чтобы Россия была в их числе.

1.1. История

В нашей стране выдающееся значение в становлении и развитии вычислительной техники имели работы академика Сергея Алексеевича Лебедева [1]. В руководимом им Институте точной механики и вычислительной техники (ИТМ и ВТ) Академии наук СССР были созданы электронные вычислительные машины (ЭВМ) пятнадцати моделей — от первых, ламповых, до быстродействующих машин на интегральных схемах, проекты которых он развернул в последние годы жизни.

Идея архитектурной линии «Эльбрус», заложенной С. А. Лебедевым, родилась в 1969 году в связи с необходимостью оснащения стратегических систем специального назначения высокопроизводительной вычислительной техникой. Главным конструктором стал Всеволод Сергеевич Бурцев, выдающийся специалист по вычислительной технике, впоследствии академик РАН. В 1979 году в ИТМ и ВТ был предъявлен государственной комиссии многопроцессорный вычислительные комплекс (МВК) «Эльбрус-1», спроектированный на базе TTL-логики. Он был введен в ряд стратегических систем [2].

Через шесть лет успешно прошел государственные испытания МВК «Эльбрус-2», построенный уже на новой элементной базе отечественной разработки — быстродействующих интегральных схемах с эмиттерно-связанной логикой серии ИС-100. Его производительность в десятипроцессорной конфигурации составляла 125 млн операций в секунду. МВК строился по модульному принципу, с учетом обеспечения надежности. Благодаря своему быстродействию и отказоустойчивости, он в течение многих лет использовался

в центральных объектах стратегических систем страны. Уникальные для того времени характеристики комплекса, необходимые в этих применениях, достигались внедрением, развитием и оптимальной реализацией ряда передовых идей в организации вычислительного процесса, которые не раз относились к числу значительных результатов мирового компьютеростроения 70–80-х годов ведущими зарубежными и отечественными специалистами [3].

Следующим этапом развития серии стал проект МВК «Эльбрус-3». Руководил им член-корреспондент Академии наук СССР Борис Арташесович Бабаян, который внес фундаментальный вклад в разработку МВК «Эльбрус-1» и «Эльбрус-2». Оценив преимущества и недостатки разработанной им суперскалярной архитектуры, потенциал которой в полной мере был реализован в МВК «Эльбрус-2», он предложил передовую архитектурную реализацию концепции широкого командного слова [4]. Опытный образец машины изготовили в 1990 году, но ее отладка не была завершена по причине прекращения финансирования проекта из-за экономических проблем того периода.

Продолжение этой проектной линии связано с деятельностью закрытого акционерного общества «Московский центр Спарк-технологий», учрежденного ИТМ и ВТ АН СССР и НИИ СуперЭВМ и вскоре переименованного в ЗАО «МЦСТ». В его структуре ведущие разработчики серии «Эльбрус», сделав принципиальную ставку на использование микропроцессорных технологий, приступили к созданию двух серий микропроцессоров и вычислительных комплексов на их основе, по утвердившейся на практике традиции объединяемых понятием «семейства “Эльбрус”». Проектной основой первой серии стала открытая архитектура SPARC (Scalable Processor Architecture), специфицированная корпорацией Sun Microsystems, второй — оригинальная архитектура «Эльбрус», развивающая принципы, которые были апробированы и заложены в МВК «Эльбрус-3» (первоначально она именовалась «архитектура E2k») и предшествующих проектах [5].

С 2006 года в разработках ЗАО «МЦСТ» непосредственно участвует коллектив открытого акционерного общества «Институт электронных управляющих машин имени И. С. Брука» (ОАО «ИНЭУМ им. И. С. Брука»).

1.2. Разработки на базе микропроцессорной архитектуры SPARC

Опыт реализации достаточно простых, хорошо специфицированных и прошедших широкую апробацию спецификаций архитектуры SPARC v8 при

освоении микропроцессорных технологий и их дальнейшем усовершенствовании позволил компании существенно расширить свои проектные возможности. Обеспечив совместимость с аналогами с точки зрения системы команд RISC-класса и программной модели, специалисты ЗАО «МЦСТ» ввели в структуру микропроцессоров большое число усовершенствований, направленных на увеличение производительности. На первом этапе были освоены технологические нормы 350 нм при создании микропроцессора МЦСТ-R150 с тактовой частотой 150 МГц, затем компания перешла к использованию технологии 130 нм и разработала одноядерный микропроцессор МЦСТ-R500 с тактовой частотой 500 МГц (выбор тактовых частот определялся главным образом возможностями полузаказного проектирования, соответствующего ресурсам компании). Увеличение производительности микропроцессоров новых поколений достигалось в результате решения проблем, усложняющихся по всему фронту разработки: внедрения современных систем проектирования, создания архитектуры и микроархитектуры кристалла, цифрового проектирования с обеспечением электромагнитной совместимости и целостности сигналов, усовершенствования физического дизайна при наличии все более строгих ограничений на размеры и показатели энергопотребления.

До введения в действие отечественных микропроцессорных производств, соответствующих современным технологическим нормам, микропроцессоры по документации ЗАО «МЦСТ» изготавливаются компанией TSMC (Тайвань).

Одновременно с микропроцессорами был реализован набор контроллеров системного класса, обеспечивающих подключение оперативной памяти, периферийных шин, межсистемных каналов, каналов ввода/вывода, и периферийных контроллеров, адаптирующих ряд стандартных и специальных интерфейсов. Большое внимание уделялось разработке конструктива, определяемого назначением средства вычислительной техники (СВТ) и типами форм-фактора.

Микропроцессор МЦСТ-R500 стал основой серии ВК «Эльбрус-90микро» [6], производимых российскими предприятиями. Это системы, использующие магистральную шину для доступа к общей памяти и связи с периферией через вспомогательные шины. ВК «Эльбрус-90микро» выпускаются в различных конструктивах и конфигурациях, включающих предназначенные для применения в перебазируемых и встраиваемых СВТ.

Основными компонентами системного программного обеспечения стали оптимизирующий компилятор с языков C/C++, две разработанные версии

операционной системы на базе ОС Linux и ОС Solaris и операционная система МСВС (Мобильная система Вооруженных Сил), поддерживающие работу в реальном масштабе времени. Они дополнены средствами защиты от несанкционированного доступа и организации работы в многопроцессорных и многомашинных комплексах, а также большим набором специальных драйверов.

Дальнейшее развитие линии микропроцессоров со SPARC-совместимой архитектурой направлено на реализацию конструктивно-технологических ресурсов производительности. С 2008 года начато серийное производство микропроцессора МЦСТ-R500S (технология 130 нм) с тактовой частотой 500 МГц, представляющего собой систему на кристалле (СНК), которая включает два процессорных ядра, общую кэш-память второго уровня, системный коммутатор, обеспечивающий связь процессоров с общей памятью и периферией, и набор контроллеров, образующих почти полную схему современного компьютера (без оперативной памяти и внешних устройств). В этой разработке была существенно поднята скорость вычислений относительно показателей микропроцессора МЦСТ-R500, который до того времени обладал высшей производительностью среди отечественных микропроцессоров.

Для использования в многопроцессорных системах микросхема МЦСТ-R500S имеет два дуплексных байтовых канала прямого доступа к памяти аналогичных систем. На этой основе создан процессорный модуль МВС/С, включающий до четырех микросхем. По существу он представляет собой восьмипроцессорную универсальную вычислительную систему на одной плате, организованную по топологии типа «кольцо». В ней каждая из четырех машин имеет собственную оперативную память, работает под управлением собственной операционной системы и получает доступ к памяти других машин.

В 2012 году начинается серийное производство прошедшей государственные испытания четырехядерной микросхемы МЦСТ-R1000 (в процессе разработки обозначалась как МЦСТ-4R), которая соответствует 64-разрядной версии SPARC v9. Она имеет тактовую частоту 1 ГГц и выполнена по технологическим нормам 90 нм. В этом проекте особое внимание было уделено созданию системы на кристалле, позволяющей строить многопроцессорные конфигурации с распределенной когерентной общей оперативной памятью соответственно архитектуре NUMA. Имеющиеся три канала межсистемного обмена дают возможность формировать полносвязные четырехпроцессорные вычислительные системы простым соединением каналов. Они реализованы в составе процессорных модулей МВС4/С и МВС4-РС.

1.3. Разработки на базе отечественной микропроцессорной архитектуры «Эльбрус»

В качестве важнейшего результата компания ЗАО «МЦСТ» рассматривает разработку микропроцессорной архитектуры «Эльбрус», ориентированной на получение максимальной для данных аппаратных ресурсов показателей производительности [7]. В общей классификации она относится к категории архитектур, использующих принцип широкого командного слова (Very Large Instruction Word, VLIW), когда компилятор формирует для параллельного исполнения последовательности групп команд (широкие командные слова), в которых отсутствуют зависимости между командами внутри группы и сведены к минимуму зависимости между командами в разных группах. Таким образом в высокой степени используется параллелизм на уровне операций, присутствующий в данном программном коде, достигается большая архитектурная скорость за счет освобождения аппаратуры от функций распараллеливания, присущих суперскалярным архитектурам, и передачи их оптимизирующему компилятору. Это обусловило и другую важнейшую особенность, свойственную архитектуре «Эльбрус», — низкий уровень энергопотребления аппаратуры.

Наряду с эффективным использованием параллелизма операций в архитектуре «Эльбрус» заложена реализация и других видов (уровней) параллелизма, свойственных вычислительному процессу: векторного параллелизма, параллелизма потоков управления на общей памяти, параллелизма задач в многомашинном комплексе. В качестве принципиального требования к архитектуре разработчики изначально рассматривали обеспечение эффективной двоичной совместимости с архитектурой микропроцессора Intel x86, которая доминирует в сегодняшних универсальных компьютерах. Она реализуется на базе скрытой динамической трансляции и ее поддержки в аппаратуре микропроцессора «Эльбрус». И наконец, к определяющим свойствам новой отечественной архитектуры относится развитая аппаратная поддержка защищенных вычислений (модульного программирования), существенно облегчающая работу программистов при создании больших программных комплексов с ограниченными сроками исполнения.

Эти свойства были реализованы при проектировании одноядерного 64-разрядного микропроцессора «Эльбрус» (технология 130 нм, тактовая частота 300 МГц). На его основе создан вычислительный комплекса «Эльбрус-3М1», два процессора которого работают на общей памяти

с симметричным доступом. В процессе государственных испытаний ВК «Эльбрус-3М1» в однопроцессорном режиме показал ускорение времени выполнения задач в среднем в 5,2 раза относительно ВК «Эльбрус-90микро» (500 МГц) и в 1,44 раза относительно Pentium 4 (1,4 ГГц). Архитектурный уровень разработки характеризуется также малым энергопотреблением микропроцессора — на этот счет получено примечательное для универсальных одноядерных микропроцессоров отношение производительности к мощности более 0,4 Гфлопс на 1 Вт.

В ходе испытаний были подтверждены эффективность системы битовой компиляции, обеспечивающей полную двоичную совместимость с архитектурной платформой Intel IA-32 (x86), и наличие средств для защищенного программирования, позволяющих создавать надежные программные комплексы. Ошибки, которые не проявлялись при обычном исполнении, были выявлены в реальных программах из программного интерфейса диалоговой работы с пользователями ВК.

После успешных государственных испытаний микропроцессора и вычислительного комплекса компания начала серийное производство и поставку ВК «Эльбрус-3М1» заказчикам. В настоящее время двухпроцессорный ВК «Эльбрус-3М1» является наиболее мощным настольным универсальным отечественным компьютером.

Программное обеспечение комплекса включает две версии операционной системы на базе ОС Linux и ОС МСВС, оснащенные средствами поддержки работы в реальном масштабе времени и защищенных вычислений, оптимизирующие компиляторы с языков С, С++ и Фортран, систему двоичной трансляции с платформы Intel x86 на платформу «Эльбрус» и компоненты, способствующие созданию функционального программного обеспечения.

С использованием результатов работы по созданию ВК «Эльбрус-3М1» спроектирован и производится унифицированный с ВК «Эльбрус-3М1» высокопроизводительный контроллер УВК/С, который обладает набором интерфейсов, рассчитанным на специальные применения.

Возможности микросхемы «Эльбрус» существенно развиты в системе на кристалле «Эльбрус-S», выполненной по технологии 90 нм и имеющей тактовую частоту 500 МГц, которая включает расширенную кэш-память и ряд контроллеров, обеспечивающих доступ к локальной секции оперативной памяти, подсистеме ввода-вывода и межпроцессорный обмен. На ее основе создан четырехпроцессорный модуль МВС3S/С.

Одним из последних результатов является создание на основе технологии 90 нм первого российского гибридного высокопроизводительного микропроцессора «Эльбрус-2С+», который интегрирует два процессорных ядра архитектуры «Эльбрус» и кластер четырех цифровых сигнальных процессоров (DSP), разработанных ГУП «ЭЛВИС». Основная сфера применения «Эльбрус-2С+» — системы цифровой интеллектуальной обработки сигналов в радарах, анализаторах изображений и других системах [8]. В этой разработке получен успешный опыт реализации архитектуры «Эльбрус» в составе многоядерных систем на кристалле, спроектированных на передовых технологических нормах. Он, несомненно, будет развит в СНК следующих поколений при создании многопроцессорных серверов и вычислительных комплексов терафлопного и петафлопного диапазонов [9]–[11].

ГЛАВА 2

Микропроцессоры с архитектурой SPARC, вычислительные средства на их основе

2.1. Микропроцессор МЦСТ-R500

2.1.1. Основные свойства архитектуры

В соответствии со спецификациями SPARCv8 [12, 13] микропроцессор МЦСТ-R500 характеризуется следующими свойствами:

- линейное 32-разрядное адресное пространство;
- небольшое количество простых форматов команд RISC-класса. Все команды 32-разрядные и выровнены в памяти по границе 32-разрядных слов. Имеется всего три базовых формата команд, в которых поля кода операции и регистровых операндов всегда находятся в одних и тех же разрядах. Доступ к памяти и ввод/вывод могут осуществляться только командами чтения/записи;
- небольшое количество способов адресации. Адрес по памяти вычисляется либо как «регистр + регистр», либо как «регистр + непосредственное значение, литерал»;
- трехадресная регистровая команда — команды большей частью выполняют действия с двумя операндами (двумя регистрами или одним регистром и константой), помещая результат в третий регистр;
- 136-регистровый файл с 8 окнами по 16 регистров и окном из 8 глобальных регистров. В каждый отдельный момент времени программа «видит» 8 глобальных целочисленных регистров, 16-регистровое текущее окно и 8 регистров из окна предыдущей процедуры. Регистровое окно может трактоваться как способ ускоренного доступа к параметрам процедуры, локальным значениям и адресам возврата;
- отдельный регистровый файл вещественных регистров. Файл может трактоваться в программах как набор из 32 регистров одинарного формата (32-разрядных), или 16 регистров двойного формата (64-разрядных), или как смесь тех или иных;
- задержанная передача управления. Процессор всегда выбирает команду, следующую за командой передачи управления. Эта команда может быть выполнена или не выполнена в зависимости от состояния аннулирующего разряда в команде передачи управления;
- быстрые обработчики прерываний. Прерывания собраны в линейную таблицу, их генерация приводит к захвату в регистровом файле нового регистрового окна.

Для каждой процедуры отводится область в памяти, называемая стеком вызова процедур. При определенных условиях возможна оптимизация, когда удается вместо создания собственной области стека использовать область вызвавшей процедуры.

Во время трансляции в области стека каждой процедуры всегда отводится место для 16 слов, нужных для сохранения входных и локальных регистров процедуры в случае выхода за верхнюю границу окон. Кроме того, в общем случае в стеке процедур отводится место:

- для одного слова для передачи скрытого (неявного) параметра. Этот параметр используется, если вызвавшая процедура ожидает получения от вызванной составного объекта в качестве возвращаемого значения. Скрытое слово содержит адрес по памяти в области стека вызвавшей процедуры;
- шести слов, куда вызванная процедура может записывать те параметры, для которых требуется вычислять адреса.

При необходимости в стеке может отводиться память:

- для дополнительных выходных параметров (сверх шести);
- динамических массивов, динамических составных объектов, автоматически размещаемых скаляров, для которых требуется вычислять адреса, и автоматически размещаемых скаляров, которым не находится места на регистрах;
- создаваемых транслятором временных переменных (обычно их слишком много, чтобы транслятору удавалось размещать все их на регистрах);
- вещественных регистров, сохраняемых при вызовах процедур (выполняется, если в процедуре используются команды вещественной арифметики).

Динамически (во время выполнения) в стеке может захватываться пространство для памяти, распределяемой с помощью библиотечной функции alloca() языка C. Доступ к адресуемым автоматически переменным, размещенным в стеке, обеспечивается с помощью отрицательных смещений относительно регистра локальной базы %fp.

Динамически распределяемая память адресуется с помощью положительных смещений относительно указателя, полученного от функции alloca().

Все остальные объекты, размещенные в стек, адресуются с помощью положительных смещений относительно регистра указателя магазина %sp.

Размещение информации в стеке активной процедуры представлено на рис. 2.1.

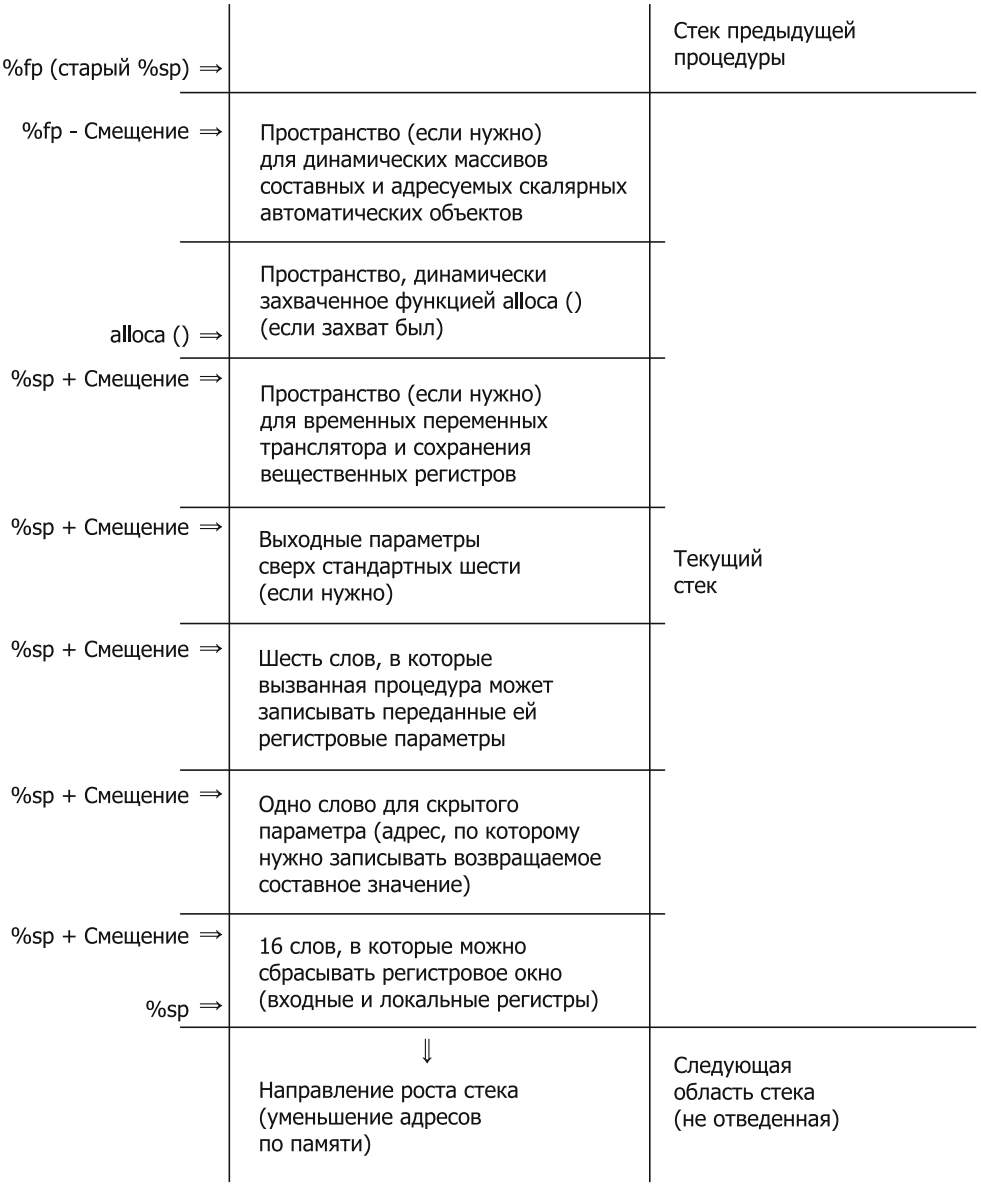


Рис. 2.1. Размещение информации в стеке активной процедуры

В каждый конкретный момент времени исполняемой программе доступны восемь глобальных регистров и 24-регистровое окно, организованное на r-регистрах целочисленного устройства. Глобальные регистры %g не являются частью какого-либо регистрового окна и представляют набор из восьми регистров, обладающих всеобъемлющей областью видимости, подобно регистровым наборам более традиционных архитектур процессоров.

Глобальный регистр %g0 имеет аппаратно «зашитое» нулевое значение. Чтение этого регистра всегда выдает нуль, запись в него не имеет никакого эффекта.

Глобальные регистры (за исключением регистра %g0) по соглашению считаются не сохраняющими своих значений при вызовах процедур. Однако если возникает необходимость использовать их по всей процедуре и считать при этом, что значения их не меняются из-за вызовов других процедур, то либо вызывающая, либо вызванная процедура должны обеспечивать сохранение и восстановление их значений.

Глобальные регистры могут использоваться для хранения временных переменных, глобальных переменных и глобальных указателей. Это могут быть и переменные пользователя, и значения, являющиеся частью программного окружения времени выполнения. Регистровое окно охватывает восемь входных и восемь локальных регистров отдельного регистрового набора, а также восемь входных регистров соседнего регистрового набора, которые в текущем регистровом окне адресуются как выходные регистры.

Текущее окно в r-регистрах задается указателем текущего окна CWP, представляющим собой трехразрядное поле счетчика в слове состояния процессора PSR. Указатель текущего окна CWP увеличивается на единицу при выполнении команды восстановления окна вызвавшей процедуры (RESTORE или RETT) и уменьшается на единицу при выполнении команды сохранения окна вызвавшей процедуры (SAVE) или фиксации события.

Выход счетчика за верхнюю и нижнюю границу определяется через регистр маски незначимых окон WIM, управляемый системными программами.

Входные и выходные регистры каждого окна являются общими с двумя соседними окнами. Выходные окна CWP+1 адресуются в текущем окне как входные, выходные текущего окна являются входными для окна CWP-1. Локальные регистры доступны только в одном (текущем) окне.

Поскольку операции над указателем текущего окна CWP осуществляются по модулю количества окон (NWINDOWS), окно с максимальным номером

перекрывается с нулевым окном. Выходные регистры для окна с номером NWINDOWS-1 являются входными для окна с нулевым номером. Окна должны нумероваться непрерывно в диапазоне от 0 до NWINDOWS-1. Схема перекрытия соседних окон и адресация регистров представлены на рис. 2.2.

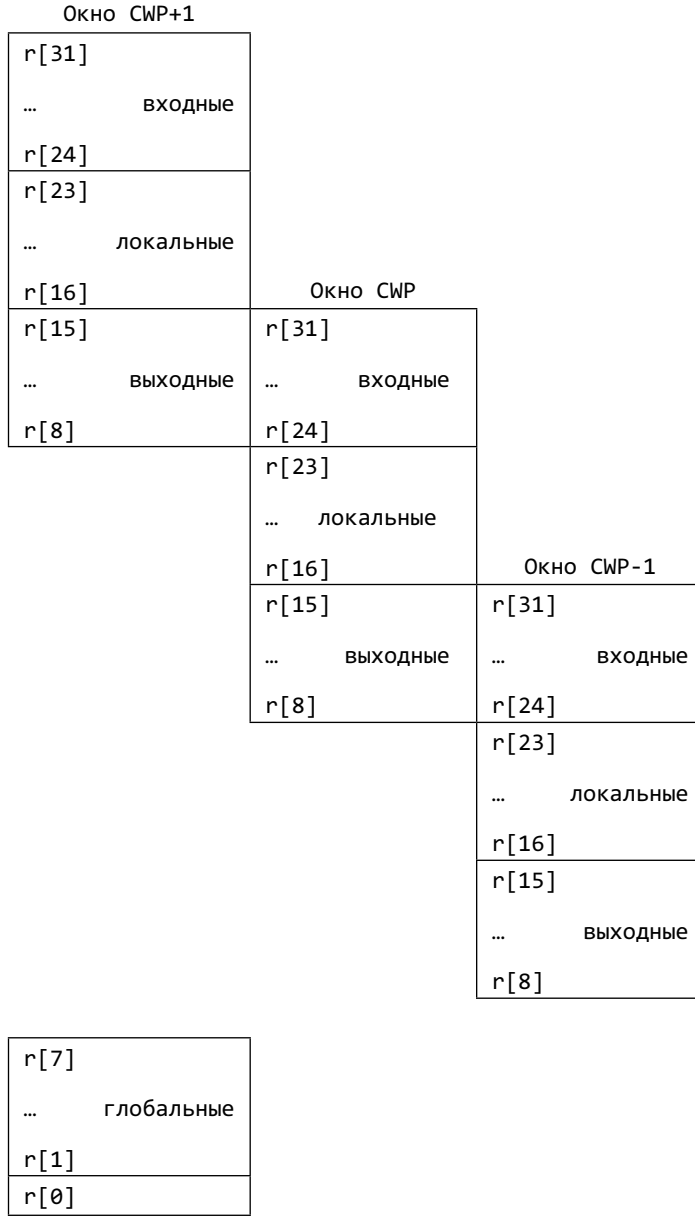


Рис. 2.2. Перекрывающиеся окна и глобальные регистры

Реализованная в архитектуре SPARC организация регистровых окон позволяет обеспечить ускоренный доступ к параметрам процедур, локальным значениям и адресам возврата.

2.1.2. Структура

Структурная схема микропроцессора МЦСТ-R500 представлена на рис. 2.3.

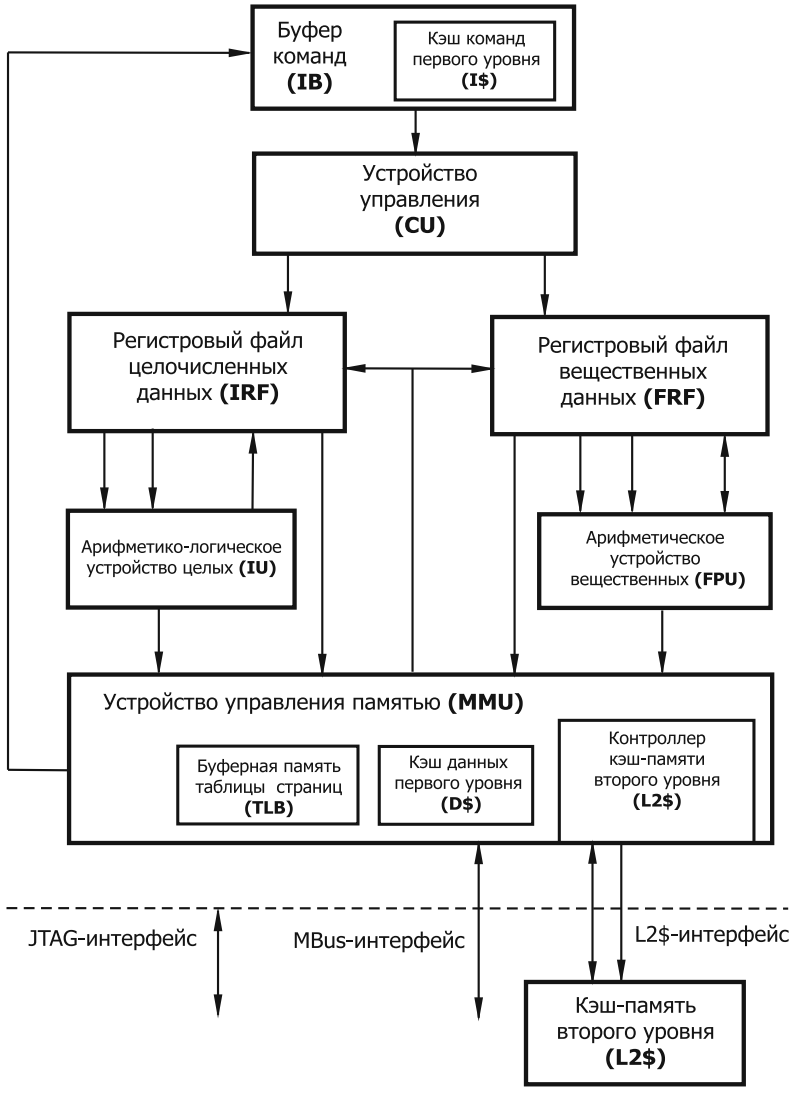


Рис. 2.3. Структурная схема микропроцессора МЦСТ-R500

В соответствии с классическими принципами организации микропроцессоров данной архитектуры [14] она включает следующие основные узлы:

- буфер команд IB (InstructionBuffer), в состав которого входит кэш команд первого уровня I$ (InstructionCache);
- устройство управления CU (ControlUnit);
- регистровый файл целочисленных данных IRF (IntegerRegisterFile);
- регистровый файл вещественных данных FRF (FloatingPointRegisterFile);
- арифметико-логическое устройство целых IU (IntegerUnit);
- арифметическое устройство (АУ) вещественных FPU (FloatingPointUnit);
- устройство управления памятью MMU (MemoryManagementUnit), в состав которого входят кэш таблицы страниц TLB (TableLookasideBuffer), кэш данных первого уровня D$ (DataCache) и контроллер кэш-памяти второго уровня L2$ (L2$ controller).

Кэш-память второго уровня L2$(L2Cache) размещается вне микропроцессора.

Внешние связи микропроцессора включают в себя интерфейс с внешней кэш-памятью второго уровня — L2$-интерфейс, MBus-интерфейс для связи с оперативной памятью и другими устройствами вычислительной системы и JTAG-интерфейс для проверки микропроцессора.

Рассмотрим работу микропроцессора (см. рис. 2.3).

Устройство управления памятью MMU выполняет обращения в память за командами и данными, получая запросы от буфера команд IB и арифметико-логического устройства целых IU. Команды поступают в буфер команд IB, данные — в регистровые файлы целых IRF и вещественных FRF. Кроме того, команды и данные заносятся в кэш команд I$ и кэш данных D$ соответственно, а также во внешний кэш L2$ на случай повторного обращения.

Буфер команд IB выдает команды в устройство управления CU. Устройство управления CU читает операнды из регистрового файла IRF для арифметико-логического устройства целых IU и из регистрового файла FRF для арифметического устройства вещественных FPU и выдает команды в эти устройства.

Арифметико-логическое устройство целых IU получает команды из устройства управления CU и операнды из регистрового файла IRF, выполняет заданные операции и записывает результаты в регистровый файл IRF. Кроме того, устройство целых IU выполняет команды обращения в память по чтению и записи для обоих регистровых файлов IRF и FRF. Арифметическое устройство вещественных FPU получает команды из устройства управления CU и операнды из регистрового файла FRF, выполняет заданные операции и записывает результаты в регистровый файл FRF.

2.1.3. Конвейеры выполнения команд

В микропроцессоре МЦСТ-R500 реализован принцип конвейерного выполнения команд [15]. Количество стадий конвейера зависит от типа выполняемой операции.

Команды над целочисленными операндами (целые команды) имеют 5-стадийный конвейер выполнения (рис. 2.4):

- F(Fetch) — выборка команды из кэша команд I$;
- D(Decode) — дешифрация (декодирование) команды и чтение операндов из регистрового файла целых IRF;
- E(Execute) — выполнение целой команды в АЛУ целых IU;
- C (Cache) — запись результата целой команды в буферный регистр;
- W(Write) — запись результата в целый регистровый файл IRF.

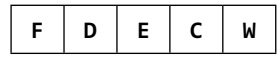


Рис. 2.4. Конвейер целых команд

Для того чтобы исключить конфликты обращения по записи в целый регистровый файл, результаты целых команд дополнительно задерживаются на один такт. Однако это удлинение конвейера целых команд не влияет на использование результатов в последующих командах, так как шины байпаса, ускоряющие передачу результата путем его пересылки непосредственно на исполнительное устройство, со стадий E и C пересылают результат на входные регистры АЛУ целых.

Конвейер выполнения вещественных команд включает семь стадий. Он представлен на рис. 2.5.

- F — выборка команды из кэша команд I$;
- D (Decode) — дешифрация команды;
- R (Read) — чтение операндов из вещественного регистрового файла FRF;
- E1, E2, E3 (Execute) — выполнение команды в вещественном АУ FPU;
- W (Write) — запись результата в вещественный регистровый файл FRF.

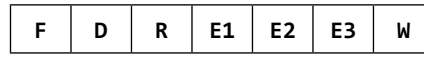


Рис. 2.5. Конвейер вещественных команд

Как и в случае целых команд, имеются шины байпаса, которые позволяют передать результаты вещественных команд на входные регистры со стадий E2 и E3 вещественного АУ. В отличие от целых команд, выполнение вещественных команд в АУ начинается на такт позже. Это дает возможность отменить выполнение вещественных команд в случае прерывания в конвейере целых команд и тем самым реализовать точное прерывание для целых команд. Другим отличием является большее время выполнения вещественных команд в АУ (3 такта для основных операций вещественного сложения и умножения).

Команды загрузки из кэш-памяти первого уровня L1$ имеют 5-стадийный конвейер выполнения (рис. 2.6):

- F — выборка команды из кэша команд I$;
- D (Decode) — дешифрирование команды;
- E (Execute) — вычисление виртуального адреса и обращение в кэш данных первого уровня D$;
- C (Cache) — формирование результата команды загрузки (формирование физического адреса запроса на случай промаха в кэше данных первого уровня D$);
- W (Write) — запись результата в целый регистровый файл IRF.

Рис. 2.6. Конвейер команд загрузки из кэша данных первого уровня D$

Конвейер выполнения команды загрузки из кэш-памяти второго уровня L2$ представлен на рис. 2.7. Дополнительно к общим стадиям здесь имеются:

- C — формирование физического адреса запроса к кэш-памяти второго уровня L2$;
- C1 — формирование запроса в контроллере кэш-памяти второго уровня L2$;
- C2, C3 — выдача запроса в кэш-память второго уровня L2$;
- C4, C5 — доступ в кэш-память второго уровня L2$;
- C6, C7 — передача тегов и данных из кэш-памяти второго уровня L2$ в микропроцессор;
- C8 — сравнение тегов из кэш-памяти второго уровня L2$ с адресом запроса;
- C9 — формирование результата команды загрузки.

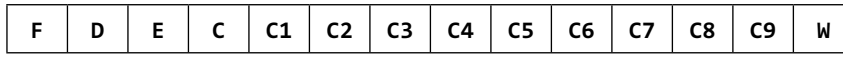


Рис. 2.7. Конвейер команд загрузки из кэш-памяти

По сравнению с конвейером загрузки из кэша первого уровня D$ данный конвейер длиннее на 9 стадий. Это вызвано формированием запроса в контроллере кэш-памяти второго уровня L2$, обращением в память тегов и данных L2$, размещенных вне микропроцессора.

2.1.4. Буфер команд

Буфер команд предназначен для выдачи запросов на получение команд в устройство управления памятью, приема команд, их промежуточного хранения и выдачи в устройство управления. Структурная схема буфера команд представлена на рис. 2.8.

Буфер команд содержит кэш команд первого уровня I$, включающий блок данных I$_data и блок тегов I$_tag, а также кэш таблицы страниц трансляции виртуальных адресов в физические I$_TLB размером в четыре строки с ассоциативным поиском. Для хранения виртуального адреса команды, подлежащей выборке в составе буфера команд, имеется регистр номера следующей команды nPC.

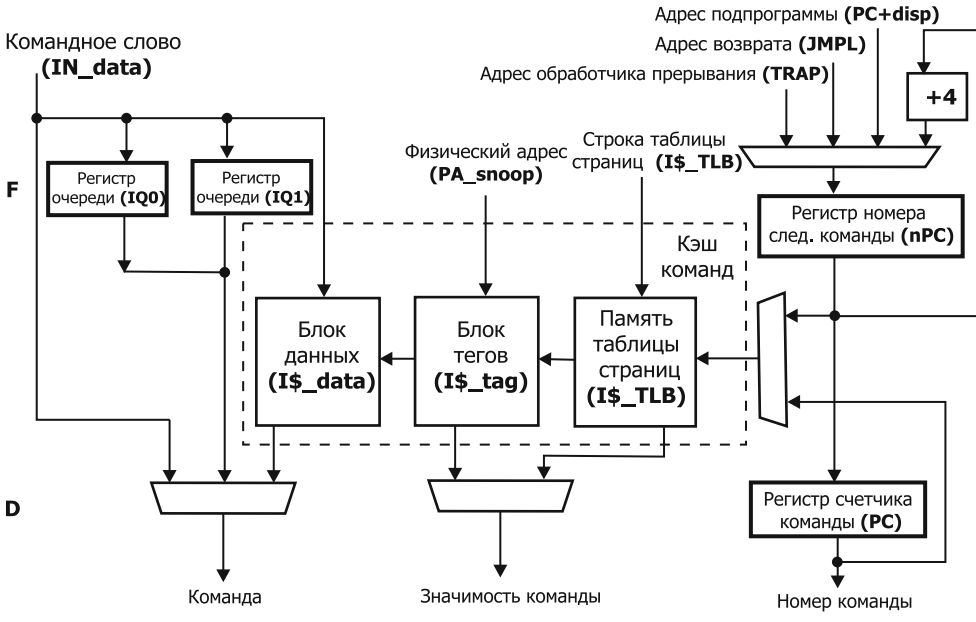


Рис. 2.8. Структурная схема буфера команд

Для временного хранения команд, подкачиваемых в кэш команд, используются регистры очереди команд IQ0 и IQ1, которые обеспечивают возможность дешифрации команды на фоне записи в накопитель. Блок данных I$_data имеет однопортовую организацию и в момент подкачки блока программного кода занят по записи. В этот момент выборка программного кода производится из регистров очереди команд.

Рассмотренные узлы буфера команд реализуют стадию выборки команды из кэша команд (F) конвейера команд.

Регистр счетчика команды PC хранит виртуальный адрес дешифруемой команды, которая с признаком ее значимости через выходные коммутаторы выдается в устройство управления. Данные узлы реализуют стадию дешифрации команды (D) конвейера команд.

Кэш команд первого уровня имеет следующие характеристики:

- емкость 16 Кбайт;
- столбцово-ассоциативная организация в виде четырех колонок по 128 строк;

- размер блока данных 32 байта;
- кэш таблицы трансляции виртуальных адресов в физические I$_TLB на четыре строки;
- теги с физическими адресами;
- 64-разрядный доступ по записи и 32-разрядный по считыванию.

Организация кэша команд представлена на рис. 2.9.

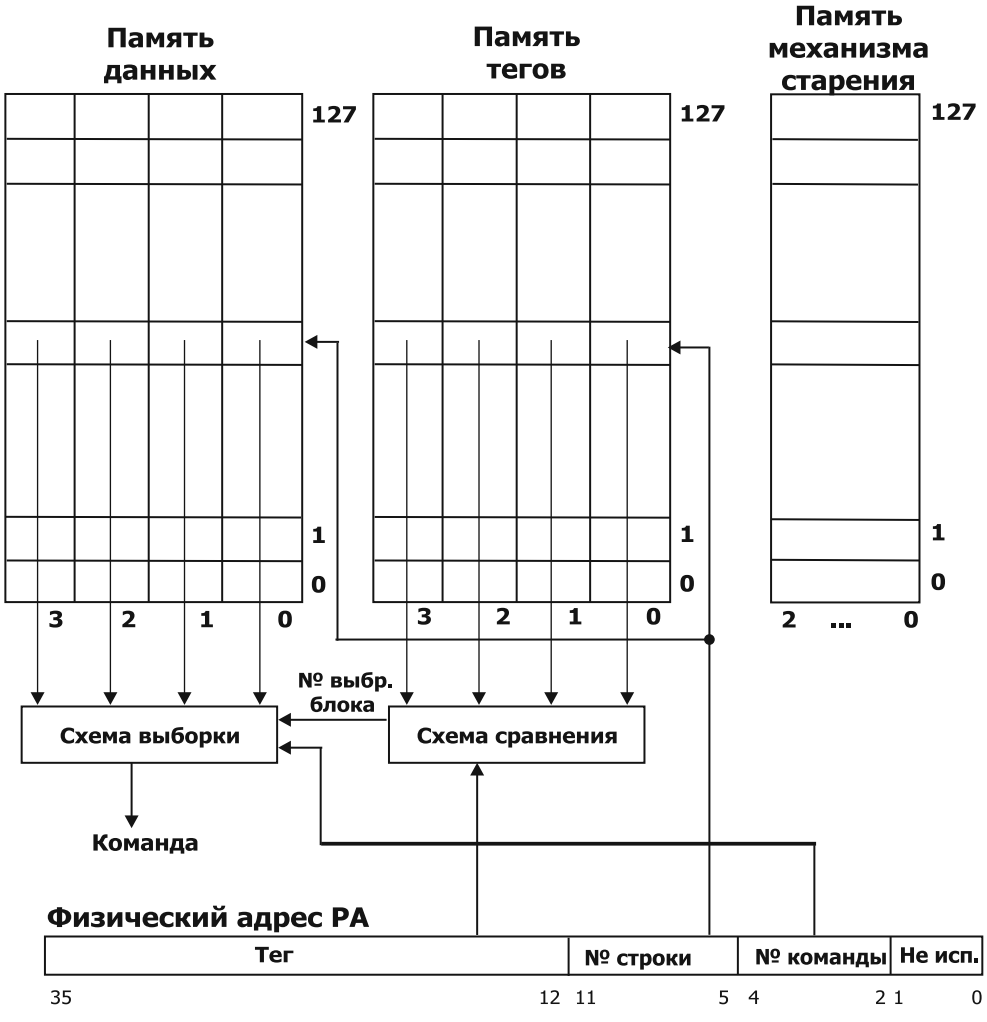


Рис. 2.9. Организация кэша команд

Кэш команд имеет три блока памяти: память данных, память тегов и память механизма старения. Все три блока памяти состоят из колонок по 128 строк.

Память данных состоит из четырех колонок и содержит 32-байтовые блоки программного кода. Память тегов также состоит из четырех колонок и содержит физические адреса блоков программного кода (старшие 24 разряда физического адреса и разряд значимости блока данных V). Память механизма старения имеет одну колонку, хранит историю обращения к блокам данных в каждой строке и содержит по 3 разряда истории использования блоков для каждой строки.

Обращение к кэшу команд включает следующие шаги.

1. Виртуальный адрес транслируется в 36-разрядный физический PA[35:0]. Для этого используется кэш-память таблицы трансляции виртуальных адресов в физические I$_TLB, которая содержит четыре регистра, хранящие физические адреса текущей и следующей страниц программного кода и таблицы страниц TLB устройства управления памятью, являющиеся копиями соответствующих строк.

2. Одновременно разряды адреса команды PA[11:5] выбирают строку в кэше команд. Разряды адреса команды PA[4:2] выбирают команду из всех четырех блоков строки памяти данных.

3. Адреса из всех четырех блоков памяти тегов выбранной строки сравниваются с разрядами физического адреса PA[35:12].

4. Если имеется соответствие адреса, хранимого в одном из блоков памяти тегов, и блок является значимым, то выбранная из соответствующего блока команда передается на дешифрацию. В противном случае фиксируется отказ в кэш-памяти команд и выполняется обращение во внешнюю кэш-память или в оперативную память.

При обработке отказов требуемый блок (32 байта) читается из иерархии памяти и направляется одновременно в кэш-память команд и регистр дешифрации команд IR. В случае нахождения команд во внешней кэш-памяти задержка их получения на 11 тактов больше по сравнению с получением команды из кэш-памяти команд первого уровня. Команды не будут выдаваться на дешифрацию до тех пор, пока очередная команда не станет доступной для дешифрации.

Подкачка из иерархии памяти выполняется двойными словами и начинается с двойного слова, содержащего требуемую команду. Первое подкачиваемое

слово заносится в кэш-память команд и одновременно поступает непосредственно на регистр дешифрации команд. Это экономит несколько тактов при обращении за командой во внешний кэш.

Если все блоки в строке памяти данных, которая выбирается при поступлении блока из иерархии памяти, оказываются значимыми, то требуется, чтобы один из значимых блоков был замещен поступившим. Выбор кандидата на замещение определяется механизмом старения.

Механизм старения определяет, какие блоки в памяти данных могут быть замещены. Память механизма старения содержит по 3 разряда использования блоков для каждой строки. Код в разрядах определяет номер блока для замещения (самый «старый» блок) в строке в соответствии с иерархией «старая пара блоков — старый блок в паре». В табл. 2.1 приведено назначение разрядов.

Таблица 2.1. Назначение разрядов механизма старения

Разряд	2	1	0
Назначение	Номер старой пары блоков	Номер старого блока в паре 1	Номер старого блока в паре 0

Значения разрядов «пара блоков» и «блок в паре» меняются на противоположные после назначения блока для замещения. Незначащие блоки в строке являются первыми кандидатами на замещение.

Рассмотрим пример работы механизма старения. Будем считать, что в исходном состоянии все блоки памяти данных являются незначащими. Состояния разрядов механизма старения после выполнения каждой операции записи приведены на рис. 2.10.

В исходном состоянии (при начале работы) все блоки в строке являются незначащими, при этом механизм старения показывает, что «самым старым» блоком является нулевой блок (нулевая пара и нулевой блок в паре). На первом шаге происходит запись в нулевой (самый старый) блок строки. При этом нулевой и второй разряды (указывающие на заменяемый блок) инвертируются. Таким образом, после записи в нулевой блок разряды механизма старения указывают, что следующим претендентом на замещение будет нулевой блок в первой паре, то есть второй блок. Дальше работа происходит по такому же алгоритму. При этом на шагах 1–4 производится замена незначимых блоков, а на шагах 5–8 — значимых блоков. Данный пример демонстрирует случай, когда блоки не переиспользуются.

Разряды механизма старения		
2	1	0
0	0	0
1	0	1
0	1	1
1	1	0
0	0	0
1	0	1
0	1	1
1	1	0
0	0	0

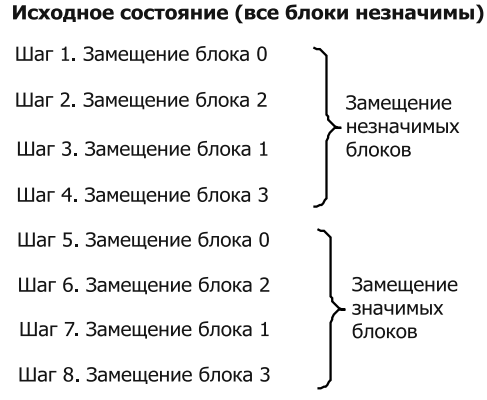


Рис. 2.10. Состояние разрядов механизма старения при замещении блоков

Кэш команд является полностью согласованным с внутренним кэшем данных, внешним кэшем и оперативной памятью. Это обеспечивается протоколом согласованности шины MBus.

Согласованность кэша команд поддерживается аппаратурой. Транзакции, поддерживающие согласованность, используют физические адреса. Если проверка данных в кэше команд является удачной, блок будет погашен (сброшен разряд значимости), когда на шине MBus имеется транзакция гашения.

В кэш-память команд нельзя записывать, поэтому она никогда не нуждается в выдаче содержимого на системную шину. Все удачные шинные проверки кэш-памяти команд обрабатываются гашением соответствующих блоков.

2.1.5. Устройство управления

Устройство управления (ControlUnit, CU) дешифрует поступающие из кэша команд инструкции, формирует пуски и блокировки устройств, организует чтение операндов из целочисленного регистрового файла IRF и запись результатов в этот файл, управляет байпасами устройства целых команд. Это устройство содержит регистры состояния процессора и организует их запись, чтение и модификацию. Кроме того, оно управляет обработкой программных и аппаратных исключительных ситуаций и внешних прерываний. Структурная схема устройства управления приведена на рис. 2.11.

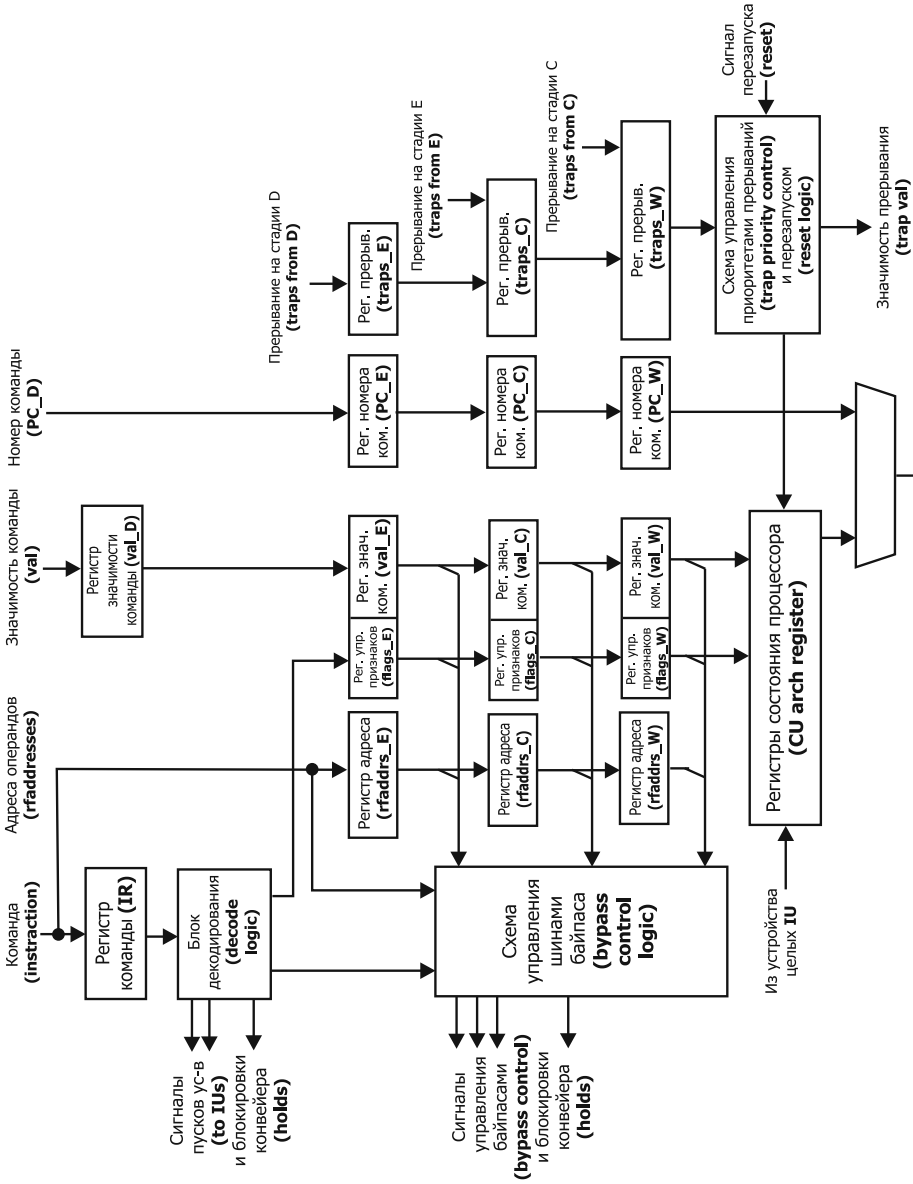


Рис. 2.11. Структурная схема устройства управления

Из буфера команд в устройство управления поступают команда, ее значимость и номер команды. Команда принимается на регистр IR, значимость команды — на регистр val_D. Регистр счетчика команды, выполняемой на стадии D (PC), физически находится в буфере команд. Адреса по целочисленному регистровому файлу принимаются на регистр, соответствующий стадии D и находящийся непосредственно в модулях памяти, на которых реализован целочисленный регистровый файл IRF.

Поступившая на регистр IR команда декодируется, при этом определяется, какие устройства должны быть запущены, какие из возможных операндов значимы или представлены литерально. Одновременно определяется доступность необходимых команде ресурсов. На основании этой информации формируются сигналы пусков исполнительных устройств и блокировки конвейера.

Продвижение команды по конвейеру отслеживается в устройстве управления. При этом необходимая информация о команде (значимость (val), управляющие признаки (flags), адреса по целочисленному регистровому файлу (rfaddrs)) передается внутри устройства управления по всем стадиям конвейера. Значимости и адреса находящихся в конвейере результатов и операндов вновь поступающей команды определяют работу схемы управления целочисленным байпасом. Вычисленные, но еще не записанные в регистровый файл результаты передаются следующим операциям через шины байпаса со стадий E, C или W.

Вместе с информацией о команде по конвейеру передаются ее номер (PC) и данные о произошедших исключительных ситуациях. Внешние прерывания привязываются к команде, находящейся на стадии D. На стадии W определяется наличие исключительных ситуаций для данной команды и привязанных к ней задержанных исключительных ситуаций из устройства плавающей арифметики. Ситуации ранжируются по определенному системой команд приоритету, и при необходимости выдается сигнал входа в прерывание, записываются в регистровый файл значения регистров PC и nPC для команды, к которой привязано прерывание, и отменяется деятельность всех команд, запущенных после нее.

В устройстве управления размещены также регистры состояния процессора (PSR), маски незначащих окон (WIM), базы прерываний (TBR), умножения–деления (Y) и некоторые другие, состояние которых модифицируется не только в ходе реализации операций записи в регистр, но и в результате возникновения в микропроцессоре некоторых ситуаций. Значения этих регистров используются для определения деталей выполнения тех или

иных действий. Устройство управления также определяет условие передачи управления для команд условного перехода на основании дешифруемой команды и текущего значения регистра условий перехода (CCR).

2.1.6. Арифметико-логическое устройство целых команд

Для выполнения команд целочисленного сложения/вычитания, логических команд, команд сдвига, а также вычисления адреса в командах обращения к памяти и передачи управления используется АЛУ, или устройство целых команд. Его структурная схема представлена на рис. 2.12.

Устройство целых команд реализует следующие стадии конвейера целочисленных команд.

1. Чтение операндов из целого регистрового файла (D).
2. Выполнение целой команды в АЛУ целых (E).
3. Запись результата целой команды в буферный регистр (C).
4. Запись результата в целый регистровый файл (W).

На стадии дешифрации команды и чтения операндов из целого регистрового файла (D) команда размещается в регистре команд IR устройства управления, выполняется ее дешифрация, производятся чтение операндов из регистрового файла целых IRFи передача их на входные сборки регистров первого операнда S1, второго операнда S2, а далее следует запись в соответствующие регистры. Код операции и адрес результата записываются в регистры операции Op и адреса результата rd.

На стадии (E) выполняется заданная команда, результат передается в буферный регистр результата res_buf.

Код операции в команде определяет, в каком функциональном блоке она будет выполняться.

Устройство целых команд включает следующие функциональные блоки:

- сдвигатель SH;
- блок логических преобразований LOG;
- сумматор данных SUM.

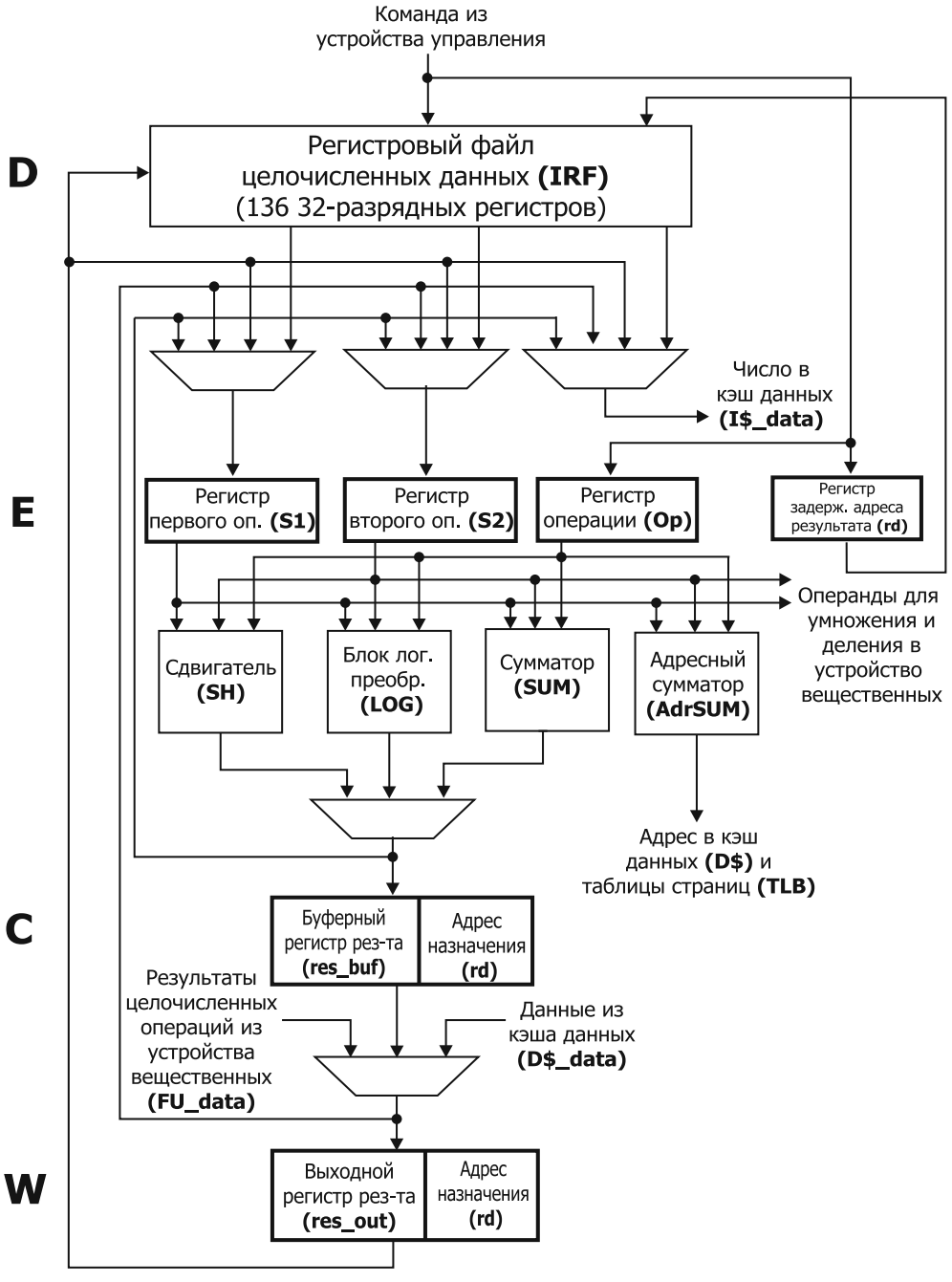


Рис. 2.12. Устройство целых команд

Результат выполнения команды выдается из отработавшего функционального блока на входную сборку буферного регистра результата res_buf с последующей записью в этот регистр.

На стадии записи результата команды в буферный регистр (C) результат команды передается из буферного регистра результата res_buf в выходной регистр результата res_out.

На стадии записи результата в целый регистровый файл (W) результат команды из выходного регистра res_out передается в целый регистровый файл IRF, где записывается в один из регистров.

Выполнение команд обращения в память имеет некоторые отличия от описанной процедуры. Адрес обращения в память формируется в адресном сумматоре AdrSUM и передается в устройство управления памятью. Записываемое число в этом же такте считывается из целого регистрового файла IRF (на такт позже, чем операнды формирования адреса обращения в память) и также передается в устройство обращения в память.

Устройство целых команд имеет три шины байпаса, используемые для сокращения времени передачи результата текущей команды в последующие для использования в качестве операндов. Первая шина байпаса передает текущий результат из входной сборки буферного регистра результата res_buf во входные регистры операндов S1 и S2, а также в регистр записываемого значения IS_data. Вторая шина байпаса передает результат из входной сборки выходного регистра результата res_out. Третья шина байпаса передает результат из самого выходного регистра результата res_out. Таким образом обеспечивается минимальная задержка передачи очередного результата в последующие команды для использования в качестве операндов. Это позволяет сократить количество тактов ожидания в конвейере выполнения целых команд в случае, когда следующая выполняемая команда использует результаты выполнения предыдущей команды, и повысить производительность.

2.1.7. Арифметическое устройство вещественных команд

Для выполнения команд с операциями над вещественными числами, а также команд умножения и деления целых чисел используется АУ, обозначаемое как устройство вещественных команд. Его структурная схема представлена на рис. 2.13.

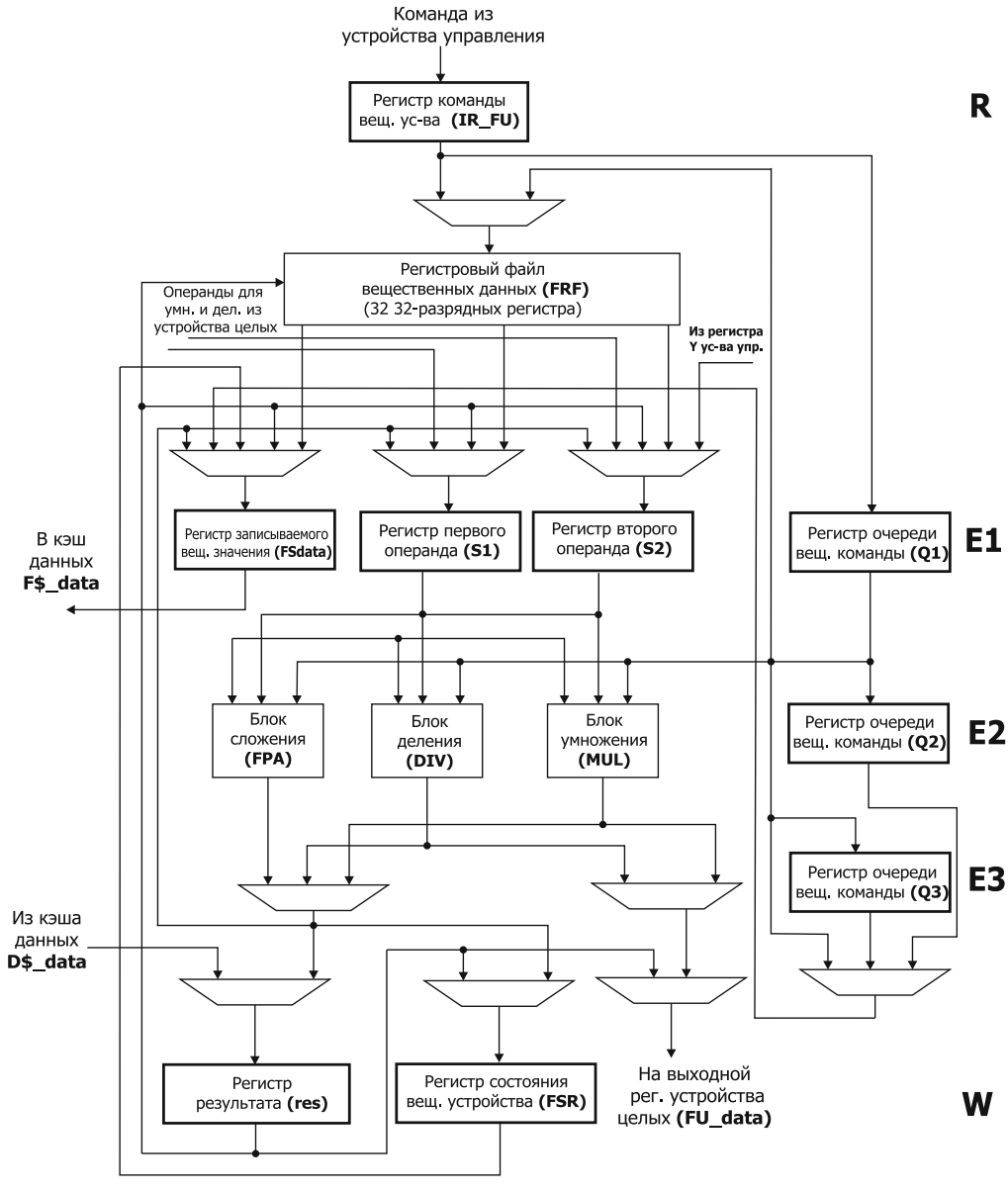


Рис. 2.13. Устройство вещественных команд

В табл. 2.2 приведено время выполнения команд в устройстве вещественных команд.

Таблица 2.2. Время выполнения операций в устройстве вещественных команд

Операция	**Количество тактов**
Сравнение двух чисел (FCMP)	2
Пересылка числа (FMOV, FNEG, FABS)	3
Преобразование числа (FiTOs, FdTOs, FiTOd, FsTOd, FsTOi, FdTOi)	3
Плавающее сложение, вычитание (FADD, FSUB)	3
Плавающее умножение (FMULs, FMULd, FsMULd)	3 (4)
Плавающее деление (FDIVs, FDIVd)	8 (11)
Извлечение квадратного корня (FSQRTs, FSQRTd)	10 (17)
Целое умножение (UMUL, SMUL)	3
Целое деление (UDIV, SDIV)	8 (11)

Устройство вещественных команд реализует следующие стадии конвейера вещественных команд: чтение операндов из вещественного регистрового файла (R), выполнение вещественной команды (E1, E2, E3) и запись результата в вещественный регистровый файл (W). Дешифрация команды выполняется в устройстве управления.

На стадии чтения операндов (R) команда записывается в регистр команды устройства IR_FU и выполняется считывание операндов из вещественного регистрового файла FRF. Считанные операнды через входные сборки регистров операндов поступают в регистры первого и второго операндов S1 и S2 соответственно.

На стадиях выполнения вещественной команды E1, E2 и E3 операции непосредственно выполняются в функциональных блоках сложения FPA, деления DIV и умножения MUL. Операции сложения и умножения выполняются за 3 такта, операция деления — за 8...11 тактов, извлечения квадратного корня — за 10...17 тактов в зависимости от формата исходных операндов. При выполнении операций деления и извлечения квадратного корня вещественное устройство остается занятым на все время операции. Это приводит к приостановке конвейера команд, если для выполнения последующей команды необходимо вещественное устройство.

Функциональные блоки деления DIV и умножения MUL используются также для выполнения соответствующих целых операций. При этом целые операнды поступают из целого устройства, а результат операции по шине FU_data передается в регистр результата res целого устройства.

На стадии записи результата (W) используются выходной регистр результата res и регистр состояния устройства FSR. Результат выполнения команды из выходного регистра res записывается в вещественный регистровый файл FRF.

Устройство вещественных команд имеет две шины байпаса, которые сокращают время передачи результата текущей команды в последующие для использования в качестве операндов. Первая шина байпаса передает текущий результат из входной сборки регистра результата res во входные регистры операндов S1 и S2, а также в регистр записываемого вещественного значения FSdata. Вторая шина байпаса передает результат от самого выходного регистра результата res. Таким образом обеспечивается минимальная задержка. Это позволяет сократить количество тактов ожидания в конвейере выполнения вещественных команд в случае, когда следующая выполняемая команда использует результаты предыдущей команды, и повысить производительность.

Очередь выполнения вещественных команд содержит программное представление этих команд. В случае прерывания в вещественном устройстве содержимое очереди может быть прочитано программным образом и использовано для анализа причины прерывания. Для хранения информации, необходимой при обработке прерывания, используются три регистра очереди вещественных команд Q1, Q2 и Q3.

2.1.8. Устройство управления памятью

В состав микропроцессора входит устройство управления памятью (УУП), которое выполняет следующие функции.

1. Определяет множество адресных контекстов, которые соответствуют программным процессам. Адреса в адресных контекстах уникальны.

2. Преобразует32-разрядные виртуальные адреса активных процессов в 36-разрядные физические адреса оперативной памяти. Соответствие виртуальной памяти физической обеспечивается для страниц размером 4 Кбайт, сегментов размером 256 Кбайт, областей размером 16 Мбайт и контекстов размером 4 Гбайт.

3. Задает совместное использование памяти. Области адресных пространств для пар контекстов могут быть установлены разделяемыми с точностью до страницы.

4. Организует защиту памяти. Доступ к адресуемой памяти назначается с установкой ограничений. Каждому контексту можно индивидуально разрешить доступ к страницам по чтению, записи или выполнению.

5. Организует подкачку отсутствующих в основной памяти страниц из внешней памяти.

6. Обеспечивает поддержку обращений операционной системы к другим контекстам в специальных режимах и режимах защиты.

Процесс отображения памяти. Физические адреса состоят из номера физической страницы PPN (PhysicalPageNumber) и смещения внутри страницы (offset) (рис. 2.14).

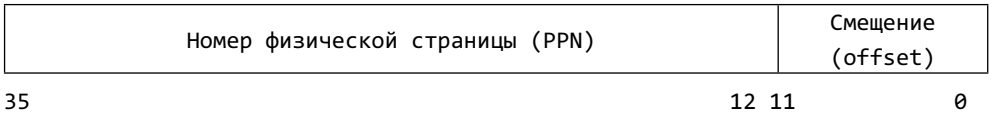


Рис. 2.14. Структура физического адреса

Виртуальные адреса состоят из номера виртуальной страницы VPN (VirtualPageNumber) и смещения внутри страницы (offset) (рис. 2.15).

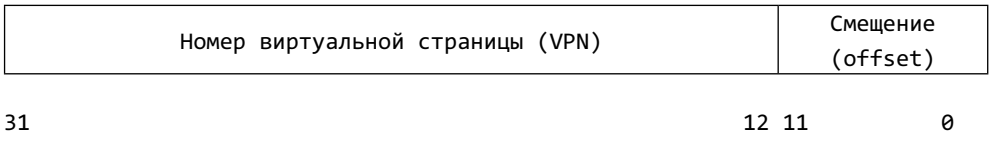


Рис. 2.15. Структура виртуального адреса

Устройство управления памятью выполняет трансляцию адресов путем обращения к древовидной таблице трансляции в оперативной памяти, называемой таблицей страниц PT (PageTable). Когда трансляция установлена, УУП загружает строку таблицы страниц в свой кэш таблицы страниц TLB (TranslationLookasideBuffer) с тем, чтобы можно было избежать обращения в память к PT при последующих обращениях. Виртуальные адреса, хранимые в кэше таблицы страниц TLB, помечаются 16-разрядным номером контекста.

Для оптимизации размещения таблиц страниц в оперативной памяти преобразование адресов выполняется с помощью 4-уровневых таблиц страниц. Такая структура обеспечивает отображение памяти для страниц размером 4 Кбайт (рис. 2.16).

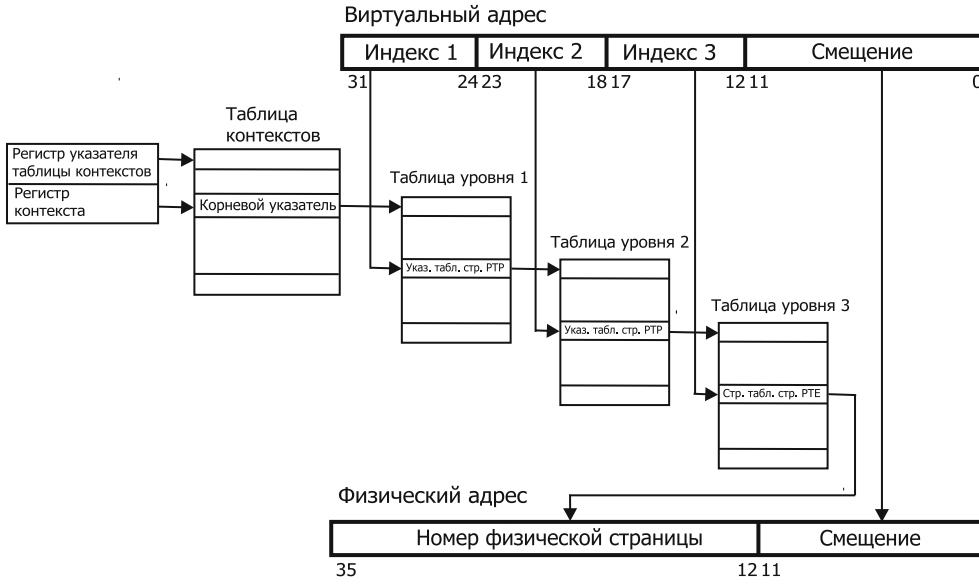


Рис. 2.16. Адресная трансляция с использованием 4-уровневых таблиц страниц

Каждое виртуальное адресное пространство определяется номером контекста, который хранится в регистре контекста. Размер регистра контекста — 16 разрядов.

Таблицы страниц могут содержать указатели таблицы страниц PTP (PageTablePointer) или строки таблицы страниц PTE (PageTableEntry). PTE отличаются от PTP двумя младшими разрядами. PTP содержит физический адрес таблицы страниц следующего уровня, тогда как PTE содержит физический адрес страницы с ее правами доступа. Структура PTE представлена на рис. 2.17.

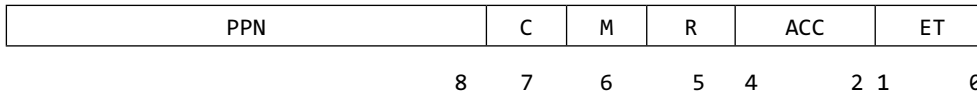


Рис. 2.17. Структура PTE

Здесь PPN — физический номер страницы. Это старшие 24 разряда 36-разрядного физического адреса. Если PTE отображает 256-килобайтовый сегмент, 16-мегабайтовую область или 4-гигабайтовый контекст, то младшие 6, 12 или 20 разрядов соответственно в PPN игнорируются.

C — кэшируемая страница. Если этот разряд установлен в 1, то страница является кэшируемой во внутреннем и внешнем кэшах. В противном случае страница не является кэшируемой.

M — модифицированная страница. Когда к странице обращаются по записи и разряд «модифицированная (M)» не установлен, УУП устанавливает его как в кэше таблицы страниц TLB, так и строке таблицы страниц в оперативной памяти.

R — использованная страница. Этот разряд устанавливается аппаратурой, когда к странице обращаются (по чтению или записи) и PTE отсутствует в кэше таблицы страниц TLB. Как и в случае с разрядом M, разряд устанавливается в строке таблицы страниц в кэше таблицы страниц TLB и в оперативной памяти.

ACC — разрешение доступа. Это поле кодируется так, как показано в табл. 2.3.

Таблица 2.3. Кодировка поля ACC

Режим выполнения команды	**Тип обращения**	**Кодировка ACC**							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Пользователь	Чтение	×	×	×	×		×		
	Запись		×						
	Выполнение			×	×	×			
Супервизор	Чтение	×	×	×	×		×	×	×
	Запись		×		×		×		×
	Выполнение			×	×	×		×	×

УУП проверяет, является ли обращение разрешенным в соответствии с режимом, в котором выполняется команда (пользователь или супервизор), и типом обращения (обращение за командой или данными). Если разрешение нарушается, то вырабатывается сигнал прерывания по доступу за данными или командой.

Структура PTP представлена на рис. 2.18.

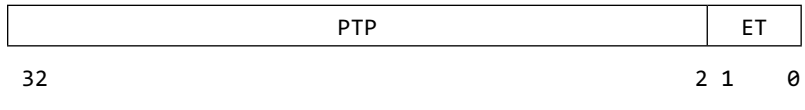


Рис. 2.18. Структура указателя таблицы страниц PTP

Здесь PTP — указатель таблицы страниц. Он является физическим адресом базы таблицы страниц следующего уровня. ET — тип строки. Это поле используется для того, чтобы отличить указатель таблицы PTP от строки таблицы страниц PTE и указать, является ли строка таблицы значимой. Кодировка поля ET представлена в табл. 2.4.

Таблица 2.4. Кодировка поля ET

ET	Тип строки
0	Незначимая
1	Указатель таблицы страниц
2	Строка таблицы страниц
3	Резерв

УУП поддерживает отображение с размерами большими, чем размер 4-килобайтовой страницы. Это делается включением строки таблицы страниц PTE (поле ET = 2) в таблицу контекстов, таблицу уровня 1 или таблицу уровня 2. Если строка таблицы страниц PTE находится в таблице контекстов, то отображение виртуального адреса в физический адрес выполняется для 4-гигабайтового контекста (рис. 2.19).

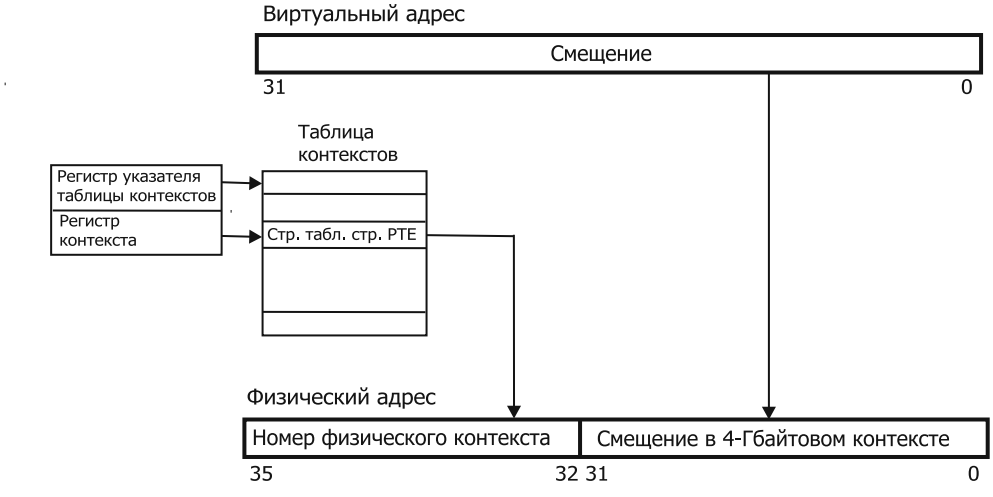


Рис. 2.19. Трансляция адреса при использовании 4-гигабайтовых контекстов

Если строка таблицы страниц PTE находится в таблице уровня 1, то отображение виртуального адреса в физический выполняется для 16-мегабайтовой области (рис. 2.20).

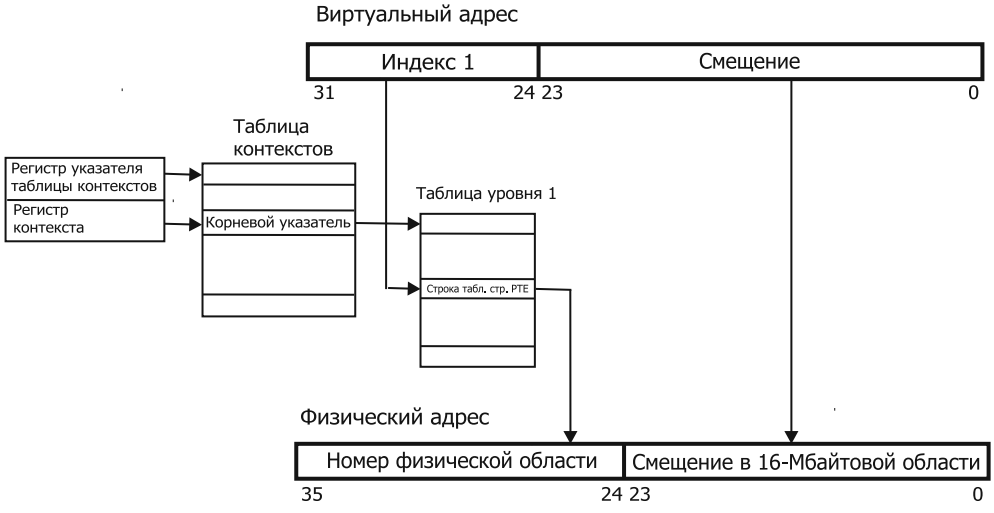


Рис. 2.20. Трансляция адреса при использовании 16-мегабайтовых областей

Если строка таблицы страниц PTE находится в таблице уровня 2, то отображение виртуального адреса в физический выполняется для 256-килобайтового сегмента (рис. 2.21).

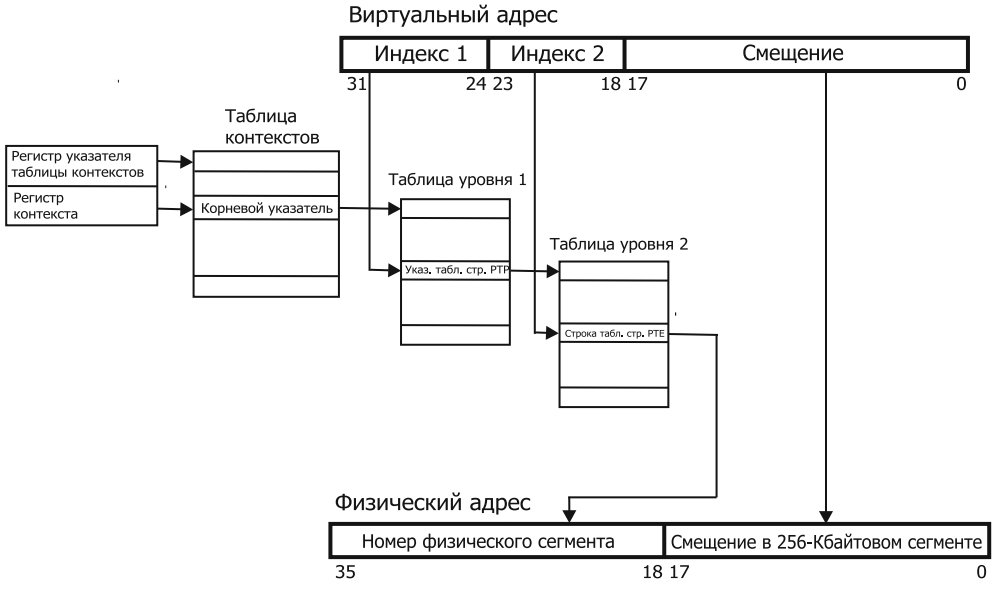


Рис. 2.21. Трансляция адреса при использовании 256-килобайтовых сегментов

Таким образом, УУП осуществляет трансляцию виртуальных адресов в физические для страниц размером 4 Кбайт, сегментов размером 256 Кбайт, областей размером 16 Мбайт и контекстов размером 4 Гбайт. Размер используемых областей для отображения определяется операционной системой.

Кэш данных и таблица страниц. В состав устройства управления памятью входят кэш данных D$ и таблицы страниц TLB, структурная схема которых представлена на рис. 2.22. В кэше данных реализуются стадии E, C и W конвейера команд загрузки из кэша данных первого уровня.

Кэш данных D$ имеет следующие характеристики:

- емкость 32 Кбайт;
- столбцово-ассоциативная организация в виде 8 колонок по 128 строк;
- размер блока данных 32 байта;
- теги с физическими адресами;
- 64-разрядный доступ.

Организация кэша данных представлена на рис. 2.23.

Кэш данных имеет три блока памяти: память данных, память тегов и память механизма старения. Все три блока памяти состоят из колонок по 128 строк. Память данных состоит из 8 колонок и содержит 32-байтовые блоки данных. Память тегов также состоит из 8 колонок и содержит физические адреса блоков данных (старшие 24 разряда физического адреса и разряды состояния блока данных: V — разряд значимости блока и S — разряд, указывающий разделяемый блок). Память механизма старения имеет одну колонку, хранит историю обращения к блокам данных в каждой строке и содержит по 7 разрядов истории использования блоков для каждой строки.

Обращение к кэшу данных включает следующие шаги.

1. Виртуальный адрес транслируется в 36-разрядный физический адрес PA[35:0].
2. Разряды физического адреса PA[11:5] выбирают одну из 128 строк.
3. Адреса из всех 8 блоков тегов в строке сравниваются с разрядами физического адреса PA[35:12].

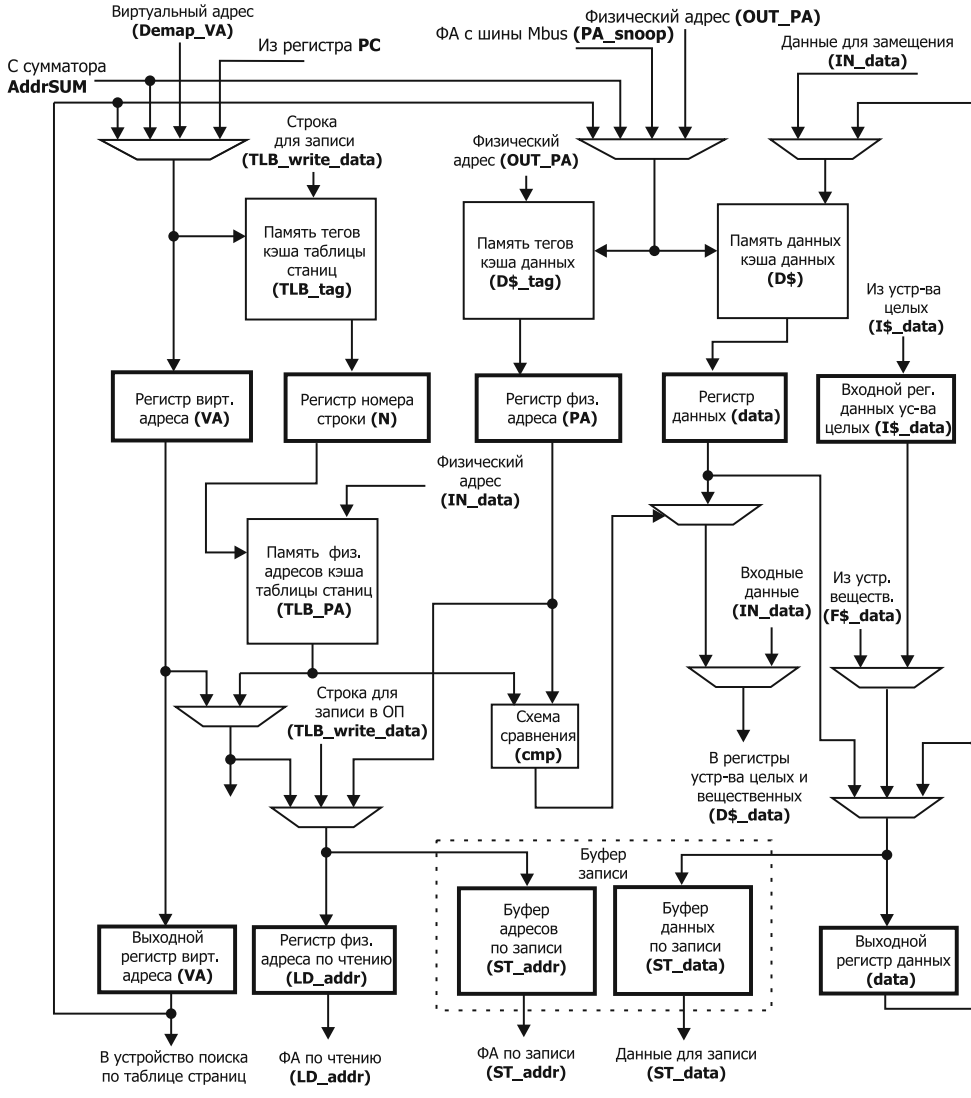


Рис. 2.22. Кэш данных и таблицы страниц

4. Если имеется соответствие адреса, хранимого в одном из блоков памяти тегов, и он значим, то соответствующий блок данных выбирается. В противном случае фиксируется промах в кэше данных и блок читается из внешнего кэша или оперативной памяти.

5. Разряды физического адреса PA[4:0] и размер транзакции определяют, какие байты возвращаются (табл. 2.5).

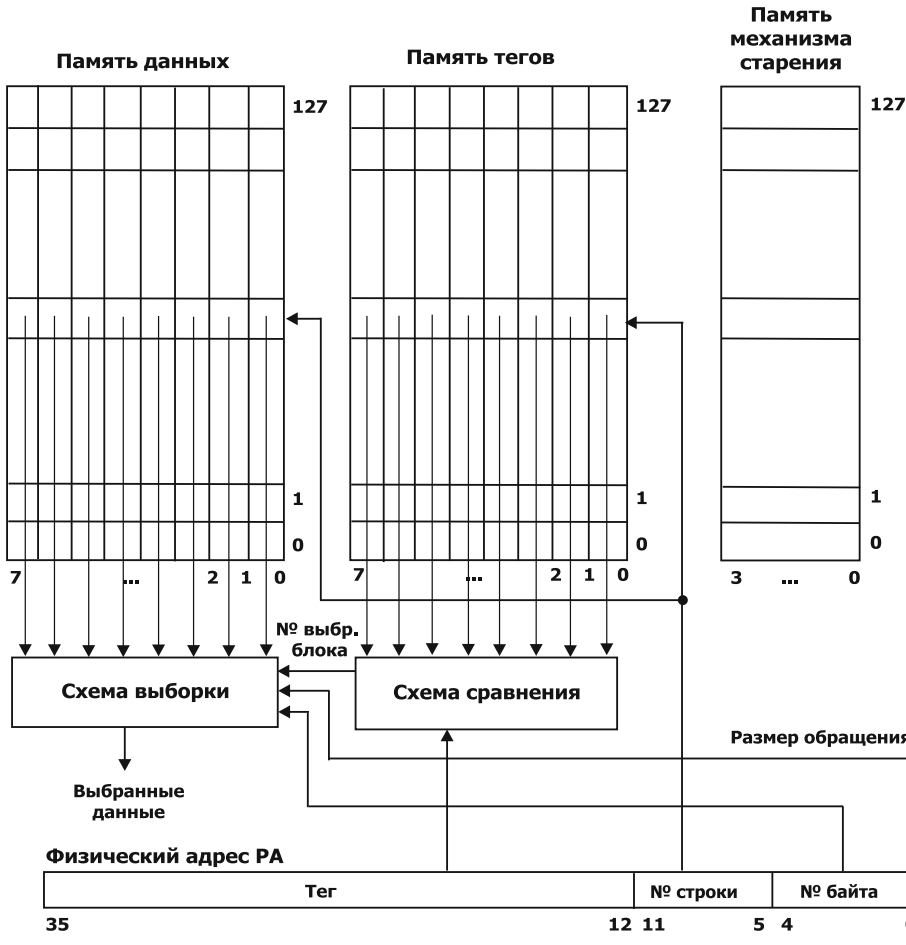


Рис. 2.23. Организация кэша данных

Таблица 2.5. Байты данных блока кэша данных, возвращаемые при доступе

PA[4:0]	Размер обращения			
	Байт	16-разрядное полуслово	32-разрядное слово	64-разрядное двойное слово
0	0	0:1	0:3	0:7
1	1	*	*	*
2	2	2:3	*	*
3	3	*	*	*
4	4	4:5	4:7	*

PA[4:0]	**Размер обращения**			
	Байт	**16-разрядное полуслово**	**32-разрядное слово**	**64-разрядное двойное слово**
5	5	*	*	*
6	6	6:7	*	*
7	7	*	*	*
8	8	8:9	8:11	8:15
9	9	*	*	*
10	10	10:11	*	*
11	11	*	*	*
12	12	12:13	12:15	*
13	13	*	*	*
14	14	14:15	*	*
15	15	*	*	*
16	16	16:17	16:19	16:23
17	17	*	*	*
18	18	18:19	*	*
19	19	*	*	*
20	20	20:21	20:23	*
21	21	*	*	*
22	22	22:23	*	*
23	23	*	*	*
24	24	24:25	24:27	24:31
25	25	*	*	*
26	26	26:27	*	*
27	27	*	*	*
28	28	28:29	28:31	*
29	29	*	*	*
30	30	30:31	*	*
31	31	*	*	*

Примечания:

* — отмечает запрещенное адресное выравнивание;

n — числа указывают номера байтов внутри блока кэша;

n:n — диапазоны чисел указывают диапазон байтов в блоке кэша.

При обработке промаха в кэше данных требуемый блок читается из следующего уровня иерархии памяти и размещается в кэше. Это может потребовать замещения значимого блока в кэше. Поскольку кэш данных имеет 8 ассоциативных колонок, имеются 8 кандидатов на замещение. Выбор кандидата на замещение определяется стратегией замещения.

Для определения кандидата на замещение используется механизм старения, который определяет, какие блоки в кэше могут быть замещены. Память механизма старения содержит по 7 разрядов использования блоков для каждой строки. Код в разрядах механизма старения определяет номер блока для замещения (самый старый блок) в строке в соответствии с иерархией «старая четверка блоков — старая пара блоков — старый блок в паре». В табл. 2.6 приведено назначение каждого разряда. Значения разрядов «четверка блоков», «пара блоков» и «блок в паре» меняются на противоположные после назначения блока для замещения. Незначащие блоки в строке являются первыми кандидатами на замещение.

Таблица 2.6. Назначение разрядов механизма старения

Разряд	Назначение
6	Номер старой четверки блоков
5	Номер старой пары в четверке 1
4	Номер старой пары в четверке 0
3	Номер старого блока в паре 1 четверки 1
2	Номер старого блока в паре 0 четверки 1
1	Номер старого блока в паре 1 четверки 0
0	Номер старого блока в паре 0 четверки 0

Для реализации многопроцессорных систем в аппаратуре микропроцессора реализован протокол удержания в согласованном состоянии локальных кэшей и оперативной памяти. Часть этого протокола использует просмотр транзакций шины. Целью просмотра является гарантирование того, что содержимое кэша данных согласовано с внешними кэшами и оперативной памятью. Кэш данных и буфер записи проверяют входящие транзакции, которые запрашивают гашение данных новой записью по их адресам.

Все адреса, поступающие на просмотр, сравниваются с тегами в кэше команд и кэше данных. Транзакции согласованного состояния кэша используют физические адреса, поэтому теги участвуют в сравнении адресов.

Проверка считается удачной, если адрес на шине соответствует значимому тегу в строке кэша.

Обеспечение согласованного состояния для внутреннего кэша заключается в гашении блока, если другой процессор выполняет запись по его адресу.

Кэш данных работает в режиме буферизованной сквозной записи. Все записи в него заносятся также в буфер записи. Буфер записи перешлет их во внешний кэш, как только ресурсы будут доступны.

Буфер записи является ассоциативным кэшем на 8 строк двойных слов. Назначение буфера записи — передача записываемых данных во внешний кэш и оперативную память.

Каждая строка буфера записи хранит состояние невыполненной операции записи. Строка содержит адрес записи и записываемые данные. Каждая команда записи требует одной строки в буфере записи независимо от того, какие данные записываются: байт, полуслово, слово или двойное слово.

Буфер записи является буфером типа FIFO (First In First Out — «первым вошел — первым вышел»). Порядок записей в буфере не меняется. Самая старая запись в буфере должна быть выполнена раньше остальных. Строка буфера записи удаляется из него по завершении записи во внешний кэш или получении подтверждения шины при записи в память, которое означает, что завершена запись в оперативную память и проверены все кэши.

Если адрес двойного слова транзакции чтения соответствует адресу двойного слова транзакции записи, хранящейся в буфере, чтение будет ждать, пока запись не завершится. Никакие данные не возвращаются из буфера записи в конвейер процессора. Когда буфер наполнен, новые операции записи вызовут остановку конвейера до тех пор, пока в буфере не станет доступной строка.

С целью сокращения затрат времени на обращение к строкам таблиц страниц в состав устройства управления памятью включен ассоциативный кэш таблицы страниц TLB на 32 строки. В нем хранятся наиболее часто используемые строки таблиц страниц.

Когда новая строка таблицы страниц PTE, отсутствующая в TLB, поступает в УУП, она должна быть записана в TLB. Если одна или более строк TLB являются незначащими (свободными), то новая строка трансляции будет записана в незначащую строку TLB с наименьшим номером. Когда все

строки TLB являются значащими, одна из них должна быть выбрана для замещения новой строкой таблицы страниц.

Для определения строки кандидата на замещение используется алгоритм LRU (LastRecentlyUsed) (наиболее давнее использование) с ограниченной историей. Каждая строка TLB имеет связанный с этим разряд использования. Он устанавливается в строке TLB, которая откликнулась при поиске в этой таблице. Когда разряды использования всех строк TLB установлены, все разряды использования, за исключением установленного последним и заблокированных, сбрасываются в 0. Это и есть случай, когда предшествующая история теряется. Теперь, когда все строки TLB являются значащими, для замещения будет выбрана строка TLB с наименьшим номером, в которой разряд использования установлен в 0. Поскольку отклик в TLB вызывает установку в 1 разряд использования, это представление наиболее давнего использования основано на доступной ограниченной истории.

Чтобы уменьшить время, требующееся для каждого поиска в таблице страниц при обращении за ее строками, УУП микропроцессора заносит в кэш два специальных указателя: корневой указатель (RootPointer) и указатель таблицы страниц второго уровня PTP2. В кэш заносится корневой указатель выполняемого процесса. Он гасится при каждом переключении контекста, и первый поиск в таблице страниц нового процесса будет использован для занесения в кэш нового корневого указателя. Кэширование корневого указателя экономит для УУП выполнение одного уровня поиска в таблице при каждом отказе в TLB для этого заданного контекста. Занесение в кэш указателя таблицы страниц второго уровня PTP2 может сэкономить для УУП обращения в память при поисках в таблице страниц.

2.1.9. Контроллер внешнего кэша и шины MBus

Внешний кэш имеет следующие характеристики:

- емкость 4 Мбайт;
- одноколоночная организация (прямое отображение данных) в виде 131 072 строк;
- размер блока данных 32 байта;
- теги с физическими адресами (физический адрес и управляющие разряды — всего 19 разрядов);

- доступ к памяти данных по следующим обращениям: байт, полуслово, слово и двойное слово.

Он реализует стратегии записи по назначению (writeallocate) и обратной записи (write-back или copy-back), которая означает, что кэшируемая запись требует размещения во внешнем кэше модифицируемого блока и записи в этот блок оперативной памяти не выполняются, пока не будет замещен блок во внешнем кэше. Организация внешнего кэша представлена на рис. 2.24.

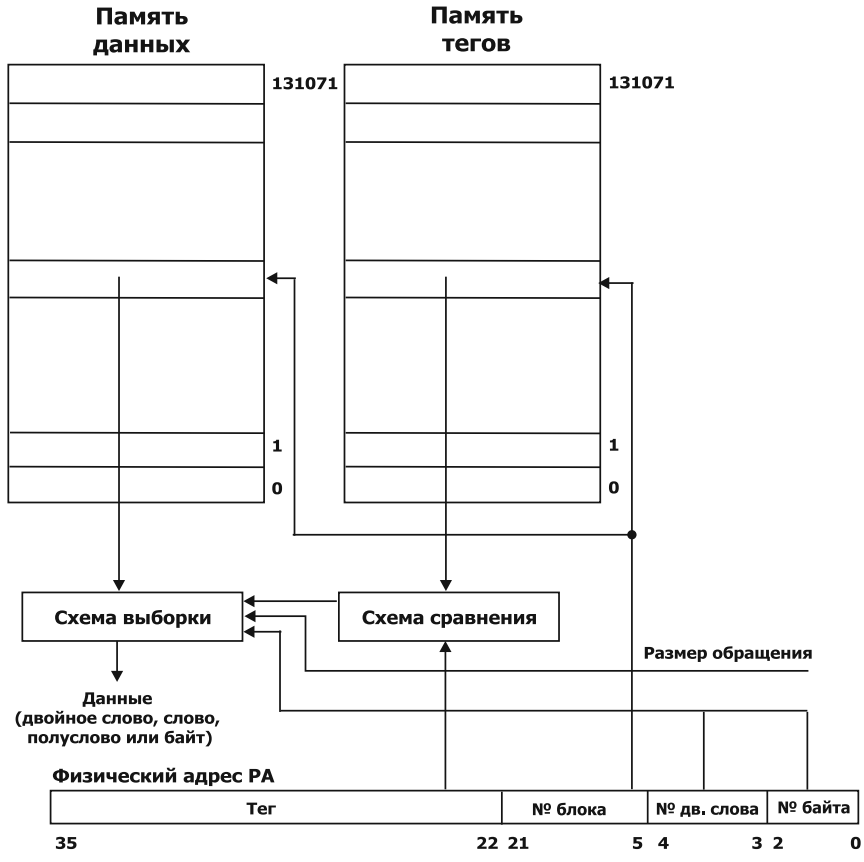


Рис. 2.24. Организация внешнего кэша

Кэш команд имеет два блока памяти: память данных и память тегов. Обе памяти состоят из 131 072 строк. Память данных хранит 32-байтовые блоки данных. Память тегов хранит физические адреса блоков данных (старшие 14 разрядов физического адреса и пять управляющих разрядов).

Обращение к внешнему кэшу включает следующие шаги.

1. Разряды физического адреса PA[21:5] выбирают строку в кэше.
2. Адрес из блока памяти тегов выбранной строки сравнивается с разрядами физического адреса PA[35:22].
3. Разряды физического адреса PA[4:3] определяют двойное слово в строке кэша.
4. Если имеется соответствие адреса, хранимого в блоке памяти тегов, и блок является значимым, то разряды физического адреса PA[2:0] и размер обращения определяют, какие выбираются данные. В противном случае фиксируется промах во внешнем кэше и выполняется обращение к оперативной памяти.

Структурная схема контроллера внешнего кэша и шины MBus представлена на рис. 2.25. (Описание шины MBus приведено в приложении 2.)

Контроллер внешнего кэша и шины MBus содержит интерфейсное оборудование для выполнения обменов с памятью данных и памятью тегов внешнего кэша, а также шиной MBus. Выходные регистры тегов tag_out, адреса a_out и данных dt_out используются для передачи данных в память данных и память тегов внешнего кэша. Входные регистры тегов tag_in и данных dt_in используются для приема данных и тегов из внешнего кэша.

Входной и выходной регистры адреса PA_in и PA_out, буфера данных D_in и D_out используются при выполнении транзакций на шине MBus.

Данные, считываемые из внешнего кэша или шины MBus, поступают в кэш команд и кэш данных с входного регистра данных IN_data. Физический адрес для установления когерентного состояния кэша данных и кэша команд выдается с буфера физического адреса PA_in.

При выполнении команды загрузки в случае промаха в кэше первого уровня адрес считывания с регистра физического адреса по чтению Ld_addr передается на выходной регистр адреса a_out. При выполнении операции записи адрес записи из буфера адреса записи St_addr также передается в выходной регистр адреса a_out.

Обращение к внешнему кэшу всегда выполняется с проверкой соответствия адреса обращения хранимым в кэше данным. Эта проверка выполняется в контроллере путем сравнения адреса выбранного блока данных, поступившего на входной регистр тегов tag_in, с адресом обращения.

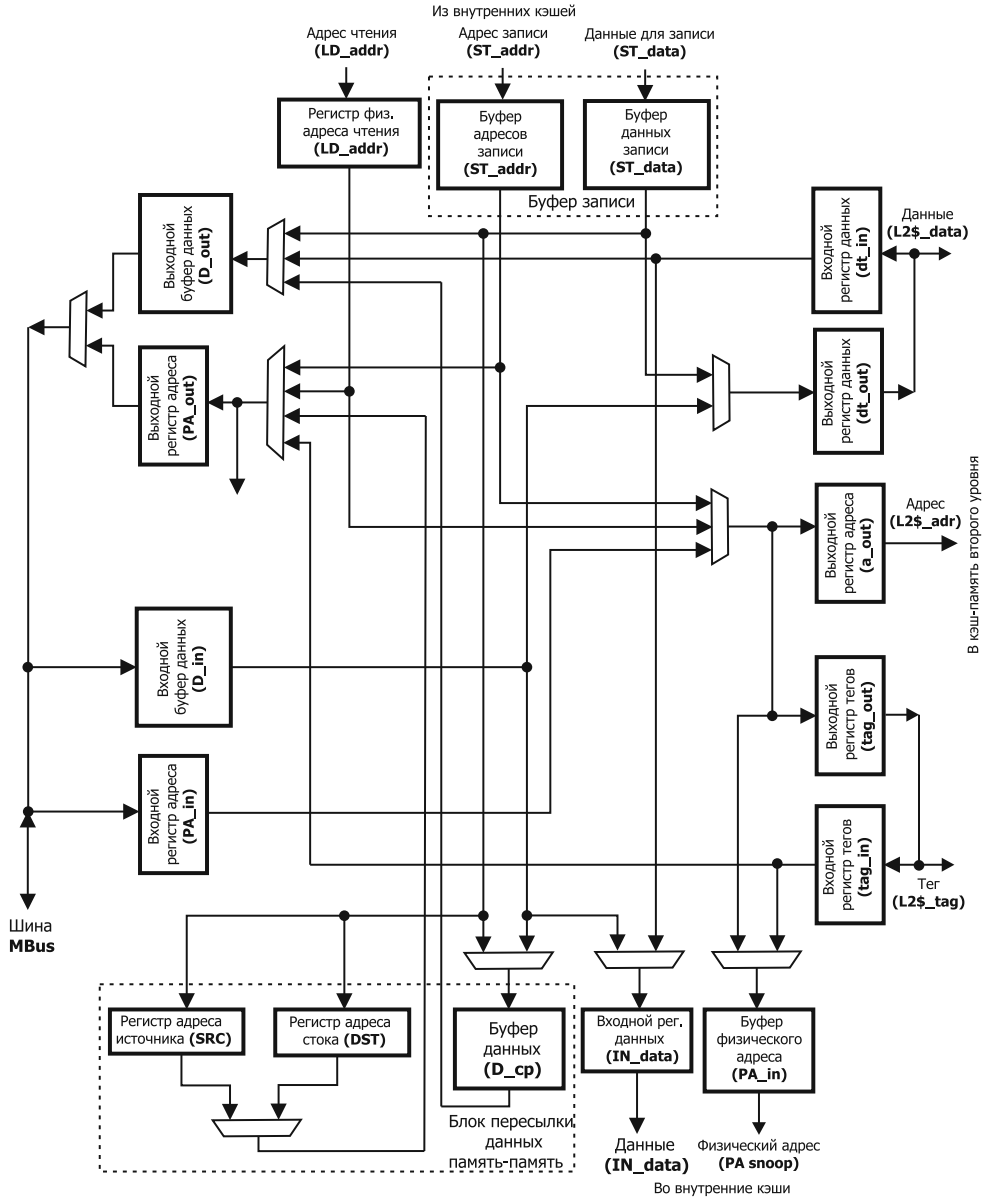


Рис. 2.25. Структурная схема контроллера внешнего кэша и шины MBus

При обращении по чтению адрес выбранного блока поступает в контроллер вместе со считанным значением, а при обращении по записи вначале считывается адрес выбранного блока для сравнения и лишь затем, в случае сравнения, выполняется запись в заданный блок.

В случае промаха в кэше второго уровня или выполнении некэшируемых обращений в память адрес обращения и данные (в случае операции записи) передаются соответственно в выходной регистр адреса PA_out и выходной буфер D_out, а далее на шину MBus. Данные с нее поступают во входной буфер D_in и через входной регистр данных IN_data — далее в процессор.

При выполнении другим процессором транзакций, требующих проверки наличия адресуемого блока данных в кэшах данного микропроцессора (snooping), адрес с шины MBusco входного регистра адреса PA_in передается в память тегов внешнего кэша через регистр адреса a_out и в кэш первого уровня — через буфер физического адреса PA_in. Кроме того, в состав контроллера внешнего кэша и шины MBus включен блок пересылки данных «память — память», состоящий из регистра адреса источника SRC, регистра адреса стока DST и буфера данных D_cp.

2.1.10. Технические характеристики микропроцессора

Приведенные в предыдущих разделах параметры устройств микропроцессора и некоторые из его общих показателей сведены в табл. 2.7.

Таблица 2.7. Технические характеристики микропроцессора МЦСТ-R500

Параметр	**Значение**
Технологический процесс, мкм	0,13
Тактовая частота, МГц	500
Число процессорных ядер	1
Производительность, MIPS/MFLOPS	440/205
Размер слов, бит	32/64
Объем внутрипроцессорной кэш-памяти, Кбайт: — команд; — данных	 16 32
Объем внешней кэш-памяти, Мбайт	4
Пропускная способность внешней кэш-памяти, Гбайт/с	1,6
Пропускная способность шины MBus, Гбайт/с	0,8
Число транзисторов, млн	4,9
Площадь кристалла, мм^2	25
Количество слоев металла	8
Корпус/количество выводов	BGA/376
Напряжение питания, В	1/2,5
Рассеиваемая мощность, Вт	< 1

2.2. Вычислительный комплекс «Эльбрус-90микро»

2.2.1. Структура и технические характеристики

Вследствие того, что все варианты ВК «Эльбрус-90микро» построены по единым архитектурным принципам [16], далее будет детально описан наиболее развитый по структуре вариант комплекса (шкафное исполнение для стационарных применений) [17]. Краткое представление других модификаций дается в приложении 1. Структурная схема ВК приведена на рис. 2.26.

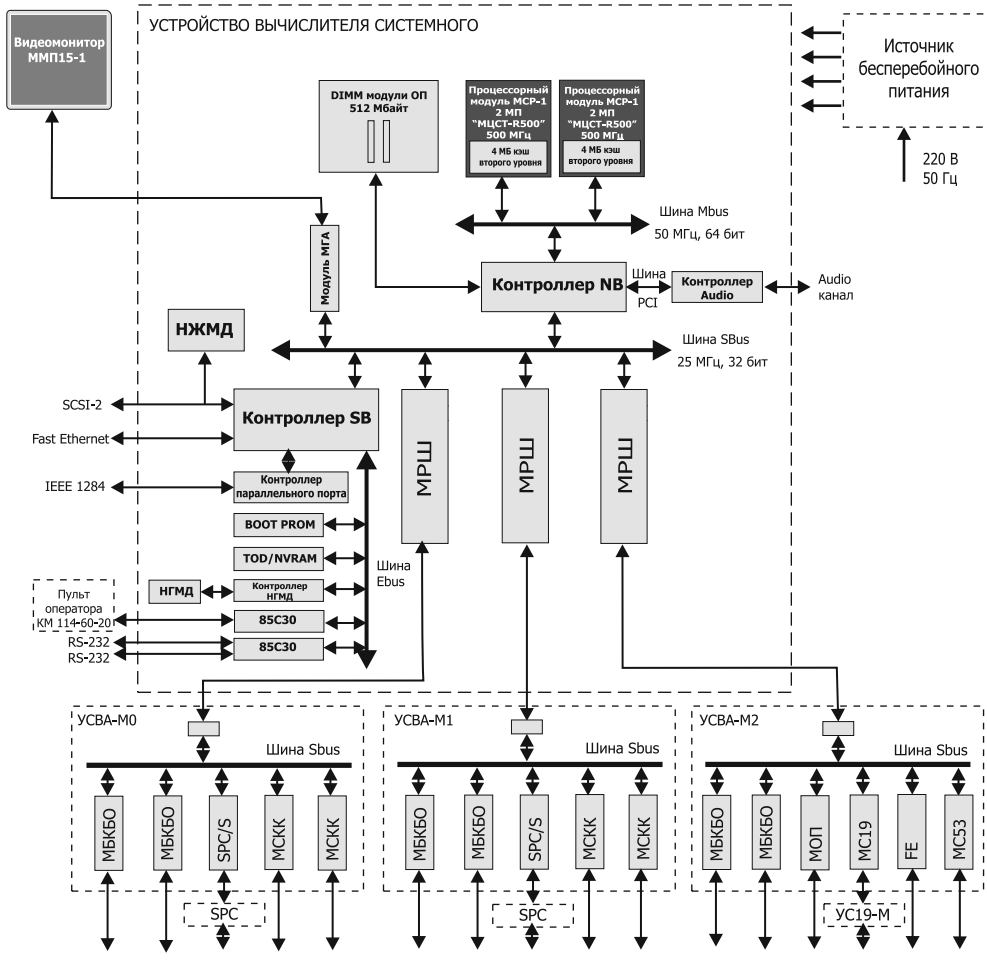


Рис. 2.26. Структурная схема ВК «Эльбрус-90микро»

Основные технические характеристики комплекса представлены в табл. 2.8.

Таблица 2.8. Технические характеристики ВК «Эльбрус-90микро»

Параметр	Значение
Тип процессора	МЦСТ-R500
Количество процессоров	2...4
Тактовая частота процессора, МГц	500
Производительность одного процессора, MIPS/MFLOPS	440/205
Оперативная память, Мбайт	512
Внутрипроцессорная кэш-память, Кбайт	48 (32 для данных, 16 для команд)
Внешняя кэш-память, Мбайт	4
Встроенная в УВС-М внешняя память, Гбайт, не менее	36
Напряжение питающей сети, В	220 ± 22
Частота питающей сети, Гц	50 ± 1
Потребляемая мощность, Вт, не более	800
Система охлаждения	Встроенная воздушного типа
Каналы ввода/вывода	Fast Ethernet, SCSI, ЕСЭВМ, RS-232/RS-423, Centronics, БК-3М, сАПД 5Ц19, 5Ц53
Средняя наработка на отказ, ч	9000
Срок службы, лет	12

2.2.2. Устройство вычислителя системного УВС-М

В качестве процессорных модулей в этом устройстве устанавливаются сменные ячейки МСР-1, использующие микропроцессоры МЦСТ-R500. При этом кэш-контроллер встроен в микропроцессор, а сама кэш-память выполнена в виде двух микросхем статической памяти общим объемом 4 Мбайт (плюс одна микросхема памяти тегов 384 Кбайт). Ячейки МСР-1 существуют двух типов: однопроцессорные и двухпроцессорные. Таким образом, максимальная конфигурация из четырех микропроцессоров требует наличия двух двухпроцессорных ячеек МСР-1.

Шина MBus, описанная в приложении 2, объединяет микропроцессоры сОП через системный контроллер NB (NorthBridge). Далее приведен список ее параметров:

- полная синхронность;
- тактовая частота 50 МГц;

- количество разрядов шины 100;
- мультиплексированные адрес и данные;
- разрядность данных 64 бита;
- разрядность физического адреса 36 бит;
- пиковая скорость передачи данных 400 Мбайт/с;
- максимальное число процессоров 4;
- централизованный арбитраж, обнуление.

Системный контроллер NB обменивается управляющими сигналами с накопителем памяти, выдает и принимает данные по двунаправленной 72-битовой информационной шине (64 разряда данных и 8 битов корректирующего кода). Накопитель имеет два разъема для подключения 168-выводных сменных модулей памяти. В зависимости от емкости и количества микросхем памяти (9 или 18 микросхем емкостью 64 или 256 Мбит) емкость одного модуля составляет от 64 до 512 Мбайт. В накопителе используются синхронные SDRAMDIMM-модули с регистрами, работающими в режиме входных усилителей для управляющих сигналов.

Максимальный объем ОП составляет 1024 Мбайт. Схема коррекции позволяет исправлять одиночные ошибки и обнаруживать двойные и блочные (блок — 4 бита) ошибки.

В системном контроллере NB размещен хост-мост MBus-PCI, выполняющий сопряжение шины MBus с шиной PCI. Реализация шины PCI поддерживает 32-разрядную шину данных и рабочую частоту 33 МГц, что соответствует спецификации локальной шины PCI версии 2.1. К шине PCI подключен контроллер Audio, имеющий соединение с аудиоканалом.

Для подключения контроллеров ввода/вывода используется 32-разрядная шина SBus с тактовой частотой 25 МГц, обеспечивающая скорость передачи данных до 100 Мбайт/с. Связь двух шин (MBus–SBus) реализует системный контроллер NB, содержащий внутренние буферы чтения и записи. Кроме того, контроллер NB выполняет функцию контроллера шины SBus.

К шине SBus подключены 4 разъема (SBus-слота) для расширения системы ввода/вывода и периферийный контроллер SB (SouthBridge), который содержит следующие внутренние контроллеры:

- контроллер стандартного ввода/вывода SEC;
- контроллер канала SBus — FastEthernet, работающий по стандарту IEEE802.3u;
- контроллер канала SBus — SCSI-2.

К системному контроллеру SB подключен контроллер параллельного порта IEEE 1284, соединитель которого используется для подключения печатающего устройства или других устройств с темпом обмена до 2 Мбайт/с.

Контроллер стандартного ввода/вывода SEC имеет выход на 8-разрядную шину ввода/вывода EBus, обеспечивающую скорость передачи данных до 10 Мбайт/с на тактовых частотах до 20 МГц. Контроллер SEC обеспечивает подключение стандартных контроллеров, в состав которых входят:

- контроллер последовательного порта 85C30, обеспечивающий подключение двух стыков RS-232C;
- контроллер последовательного порта 85C30, обеспечивающий подключение клавиатуры;
- часы и конфигурационная память TOD/NVRAM, сохраняющая информацию при отключении питания;
- постоянное запоминающее устройство BOOTPROM, содержащее код инициализации системы при включении питания;
- контроллер накопителя на гибких магнитных дисках (НГМД).

Кроме того, узел SEC содержит контроллер прерываний и таймеры (четыре процессорных и один системный).

Параметры встроенных каналов ввода-вывода устройства УВС-М приведены в табл. 2.9.

Таблица 2.9. Параметры встроенных каналов ввода-вывода устройства УВС-М

Канал ввода/вывода	Количество	Скорость	Возможное удаление от УВС-М, м
FastEthernet	1	100 Мбит/с	Десятки до разветвителя
SCSI-2	1	10 Мбайт/с	До 3
IEEE 1284	1	2 Мбайт/с	До 3
RS-232	2	9600 бит/с	До 50

Размещенный непосредственно на системной плате соединитель SCSI обеспечивает возможность подключения внутреннего накопителя на жестких магнитных дисках (НЖМД) с интерфейсом SCSI-2.

SBus-слоты позволяют установить до четырех SBus-ячеек. Один SBus-слот использован для подключения ячейки МГА (модуль графического адаптера), три — для подключения ячеек МРШ (модуль расширения шины). Ячейка МГА предназначена для вывода графической или текстовой информации на экран видеомонитора. Она представляет собой однослотовую SBus-ячейку с двумя соединителями, которая выполняет функции видеоадаптера. Ячейка МРШ предназначена для промежуточного хранения информации, передаваемой между SBus-шиной и устройством УСВА-М, и сигналов прерываний, поступающих с SBus-шины и устройства УСВА-М.

2.2.3. Устройство сопряжения с внешними абонентами УСВА-М

В рассматриваемый вариант комплекса входят три устройства УСВА-М, каждое из которых обеспечивает возможность расширения системы ввода/вывода ВК путем подключения до шести адаптерных ячеек, устанавливаемых в слоты SBus. В составе устройства реализован мост-коммутатор типа SBus —SBus, связанный с УВС-М (ячейка МРШ).

Устройство УСВА-М допускает подключение до шести ячеек следующих типов. Ячейка модуля быстрого канала МБКБО связывает УСВА-М с одним или несколькими аналогичными устройствам и может вести обмен информацией по волоконно-оптическому кабелю на расстояниях до 500 м с темпом до 1200 Мбит/с в конфигурациях «канал — канал» (межкомплексный обмен) и «канал — абонент» (связь с высокоскоростными внешними устройствами). Она позволяет создавать соединения типов «точка — точка» и «шлейф». В интерфейсе МБКБО допускаются:

- монопольный (селекторный) полудуплексный режим обмена;
- блок-мультиплексный полудуплексный режим обмена;
- дуплексный режим обмена (в основном для МКО).

Ячейка модуля стандартного канала МСКК предназначена для связи между ВК и каналом ввода/вывода ЕС ЭВМ (ЕС ЭВМ-канал).

Ячейка модуля обработки прерываний МОП осуществляет прием и обработку внешних прерываний в соответствии с заданной полярностью

и маской прерывания и обеспечивает передачу в SBus-шину сигнала прерывания, а также выходных прерываний.

Ячейка FE предназначена для работы в составе вычислительного комплекса при обмене информацией по протоколу FastEthernet по одному каналу на расстояние до 100 м с темпом обмена до 100 Мбит/с.

Ячейки модулей связи MC53 и MC19 предназначены для обмена информацией с двумя различными типами аппаратуры передачи данных.

Система SPC обеспечивает возможность увеличения числа последовательных портов ВК для подключения терминалов, модемов, принтеров и других периферийных устройств. Она имеет до восьми последовательных полнодуплексных портов, а также один однонаправленный принтерный порт, совместимый с интерфейсом Centronics, и может обеспечить подключение многопользовательских прикладных устройств с относительно малым темпом передачи информации, например для реализации задач ввода данных и управления процессом.

2.3. Система на кристалле МЦСТ-R500S

При разработке микросхемы МЦСТ-R500S была поставлена цель создания высокопроизводительных одноплатных ЭВМ для носимых и встроенных применений, предполагающая также построение многомашинных конфигураций [18]. В связи с тем, что исходным требованием было всемерное использование результатов предшествующей разработки микропроцессора МЦСТ-R500, в новом проекте за основу была взята архитектура SPARC.v8, реализованная при технологических нормах 130 нм и тактовой частоте 500 МГц. Конструктивно для микросхемы был выбран пластмассовый корпус типа 900LHFCBGA с 900 шариковыми выводами (рис. 2.27). Ее технические характеристики приведены в табл. 2.10.

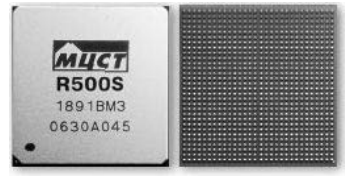


Рис. 2.27. Внешний вид системы на кристалле МЦСТ-R500S

Таблица 2.10. Технические характеристики системы на кристалле МЦСТ-R500S

Параметр	**Значение**
Тип процессора	МЦСТ-R500
Количество ядер	2
Тактовая частота, МГц	500
Производительность, MIPS/MFLOPS	1000/380
Внутренняя кэш-память первого уровня (для одного ядра), Кбайт	48 (32 для данных, 16 для команд)
Кэш-память второго уровня, Кбайт	512
Объем оперативной памяти, Гбайт	До 2
Пропускная способность канала оперативной памяти, Гбайт/с	2,664
Канал удаленного доступа	RDMA
Количество каналов	2
Пропускная способность канала в одном направлении, Мбайт/с	667
Пропускная способность: — шины PCI, Мбайт/с; — канала Ethernet, Мбит/с; — шины SCSI-2, Мбайт/с; — канала RS-232, Кбит/с; — шины EBus, Мбайт/с	 264 100 10 115 10
Потребляемая мощность, Вт	5
Количество транзисторов, млн шт.	45
Напряжение питания, В: — для внутренних схем; — для периферийных схем	 1,05 2,5/3,3
Тип корпуса	HFCBGA
Количество выводов	900
Технология	КМОП, 0,13 мкм; 8 слоев металла
Площадь кристалла, мм	9×9

Усовершенствование технических характеристик по сравнению с МЦСТ-500R в основном было достигнуто в результате следующих структурных нововведений:

- реализации двухъядерной процессорной части;

- создания системы на кристалле, включающей общую для обоих процессоров кэш-память второго уровня, контроллер оперативной памяти и набор периферийных контроллеров для доступа к внутренним узлам компьютера, внешним каналам и линиям связи;
- замены магистральной шины, объединяющей основные узлы микросхемы, быстродействующим системным коммутатором;
- введения каналов прямого доступа к памяти аналогичных систем, позволяющего строить многомашинные конфигурации.

Соответствующая этим решениям структурная схема системы на кристалле МЦСТ-R500S приведена на рис. 2.28.

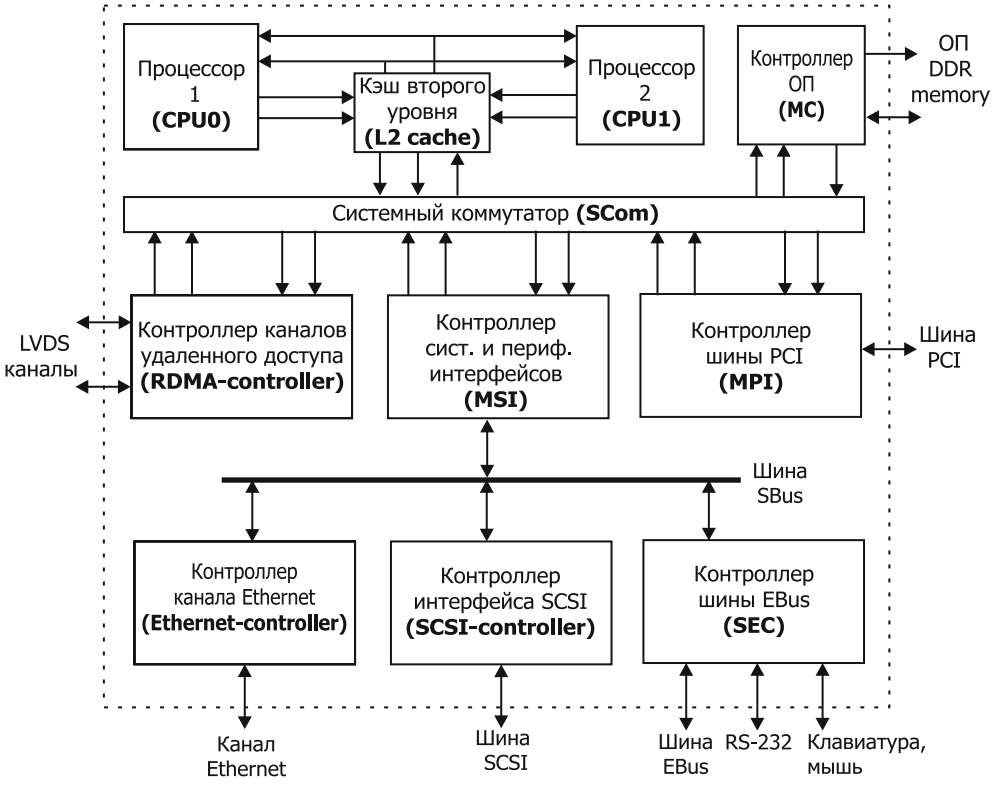


Рис. 2.28. Структурная схема системы на кристалле МЦСТ-R500S

Универсальный процессор CPU реализует архитектуру SPARCv8. Целочисленная и адресная арифметика — 32-разрядная, вещественная — 32- и 64-разрядная.

Кэш-память второго уровня L2$ предназначена для буферизации данных, часто используемых процессорами, и организации доступа в оперативную память при промахе или при обращении в пространство ввода-вывода.

Запросы в L2$ от каждого процессорного ядра поступают по двум каналам — чтения и записи.

Системный коммутатор SCom обеспечивает взаимодействие между процессорным модулем, состоящим из двух процессоров и общей кэш-памяти второго уровня, контроллером каналов удаленного доступа RDMA, контроллером памяти MC, контроллером шины SBusMSI и контроллером шины PCIMPI.

Коммутатор SCom может обслуживать одновременно до восьми запросов от процессорного модуля, до четырех запросов от контроллера RDMA, по одному запросу от контроллеров MSI и MPI.

Обслуживание одновременно нескольких запросов обеспечивается наличием в коммутаторе SCom-буферов для их хранения. Все запросы сначала попадают в соответствующий буфер. Далее они декодируются, проходят арбитраж с запросами от других устройств и направляются к местам назначения.

Контроллер памяти MC поддерживает DDRSDRAM-модули памяти объемом от 128 Мбайт до 1 Гбайт с одним или двумя физическими банками.

В контроллере памяти MC организована очередь, позволяющая принимать следующий запрос до выполнения предыдущего. Буфер данных очереди запросов рассчитан на одновременное нахождение в очереди четырех запросов записи.

Контроллер удаленного доступа RDMA выполняет функцию высокоскоростного DMA-канала для связи вычислительных комплексов или подключения внешних быстрых контроллеров. В ходе транзакции контроллер RDMA обеспечивает когерентную передачу информации. Между транзакциями когерентность не поддерживается (модификация оперативной памяти в одной системе не вызывает запросов просмотра кэшей (snoop-запросов) в другую систему). Это позволяет существенно упростить протокол при построении многомашинных систем и адаптировать его для быстрой передачи большого объема данных между системами с использованием DMA. Для организации сети произвольного размера предполагается использование коммутирующих устройств. Организация сети прямого доступа (до

4 систем на кристалле МЦСТ-R500S) возможна путем непосредственной коммутации RDMA-каналов.

Физически RDMA-интерфейс представляет собой два независимых настраиваемых полнодуплексных LVDS-канала. Каждый канал включает по десять дифференциальных пар для приема и передачи: сигнала синхронизации, строб значимости и восьми разрядов данных. Стробирование данных осуществляется по обоим фронтам сигнала синхронизации. Соответственно, за каждый такт сигнала синхронизации передаются два байта пакета в каждом канале.

Возможны три основных режима работы RDMA-контроллера: DMA-режим, BYPASS-режим и BUS-режим.

В DMA-режиме контроллер передает данные в системную память или из нее. В DMA-режиме внешний RDMA-запросчик может организовывать только запись приходящих из канала данных в оперативную память.

В BYPASS-режиме контроллер используется для связи двух соседних с ним систем. Информация передается из одного LVDS-канала в другой. В BYPASS-режиме оперативная память системы не используется.

В BUS-режиме LVDS-канал используется в качестве быстрой периферийной шины, к которой могут быть подключены контроллеры, способные организовывать транзакции записи/чтения оперативной памяти.

Оба канала контроллера RDMA могут работать независимо друг от друга. С оперативной памятью контроллер обменивается блоками размером 32 байта.

Контроллер MPI выполняет функции моста между системным коммутатором SCom и внешней периферийной шиной PCI. Мост принимает из системного коммутатора транзакции, обращающиеся в адресное пространство PCI, и преобразует их в транзакции на шине PCI. В этом случае мост является ведомым для коммутатора SCom и ведущим для шины PCI.

При обращении PCI-устройств в оперативную память мост преобразует приходящие с шины PCI транзакции в формат транзакций системного коммутатора. В этом случае мост является ведомым для шины PCI и ведущим для коммутатора SCom.

Максимальный передаваемый за транзакцию объем информации — 8 байт из системы в шину PCI и 32 байта при передаче с шины PCI в систему.

Все обращения процессора в адресное пространство PCI производятся по физическим адресам. Запрос от PCI-устройства может сопровождаться как физическим, так и виртуальным адресом.

Контроллер MSI выполняет функции моста между системным коммутатором SCom и низкоскоростными контроллерами, объединенными на внутренней периферийной шине SBus. Он выполняет трансляцию запросов системы к низкоскоростным контроллерам в формат шины SBus и запросов этих контроллеров к оперативной памяти.

Контроллер MSI поддерживает обращения в следующие адресные пространства:

- регистры моста MSI;
- периферийные SBus-устройства;
- пространство контроллеров SCSI, Ethernet, RS-232, клавиатуры и мыши;
- пространство контроллера шины EBus и прерываний.

При обращении процессора в адресное пространство шины SBus поддерживаются транзакции размером до 8 байт.

Операции чтения производятся синхронно, операции записи — асинхронно, для чего предусмотрен буфер на одну транзакцию.

DMA-операции производятся контроллером по запросу периферийных устройств. Периферийные устройства могут использовать как виртуальные, так и физические адреса. DMA-записи могут проводиться как в синхронном, так и в асинхронном режиме. Синхронный режим реализован посредством откладывания последнего подтверждения данных на периферийной шине до окончания операции. Для работы в асинхронном режиме предусмотрен буфер на две транзакции.

Контроллер SCSI предназначен для организации передачи информации по шине SCSI с поддержкой протокола FASTSCSI-2. Ширина шины — 8 бит. Контроллер SCSI поддерживает асинхронный и синхронный режимы. Максимальная скорость передачи информации — 10 Мбайт/с. Максимальный блок передаваемой информации — 16 Мбайт.

Контроллер канала Ethernet реализует независимый от среды передачи информации интерфейс 100 MbitEthernet.

Контроллер SEC периферийной шины EBus обеспечивает взаимодействие между 32-разрядной периферийной шиной Sbus и 8-разрядной шиной Ebus, которая используется для подключения медленных устройств, работающих только в режиме программного доступа (без DMA-режима).

К шине EBus подключаются как внутренние, так и внешние устройства. К внутренним устройствам относятся:

- контроллер последовательных каналов;
- контроллер клавиатуры и графического манипулятора.

В качестве внешних устройстве к шине EBus могут быть подключены:

- часы и статическая память NVRAM (Non Volatile Random Access Memory);
- ППЗУ для хранения программ начального тестирования, инициализации и загрузки операционной системы BOOTPROM;
- контроллер НГМД.

В состав контроллера SEC входят также блок счетчиков/регистров для организации таймеров и контроллер прерываний, который получает все прерывания системного уровня, кодирует их и направляет назначенному процессору. Контроллер SEC поддерживает операции считывания/записи форматом от 1 до 8 байт.

2.4. Процессорный модуль МВС/С

На базе системы на кристалле МЦСТ-R500S создан процессорный модуль МВС/С, который является восьмипроцессорной одноплатной универсальной вычислительной системой с оперативной памятью до 8 Гбайт и набором периферийных контроллеров. Конструктивно модуль МВС/С представляет собой плату «Евромеханика» размера 6U, занимающую один слот в крейте CompactPC. Его внешний вид и структурная схема представлены на рис. 2.29 и 2.30 соответственно.

Процессорный модуль состоит из 4 машин, каждая из которых имеет собственную оперативную память и работает под управлением собственной копии операционной системы. Машина содержит двухпроцессорную систему

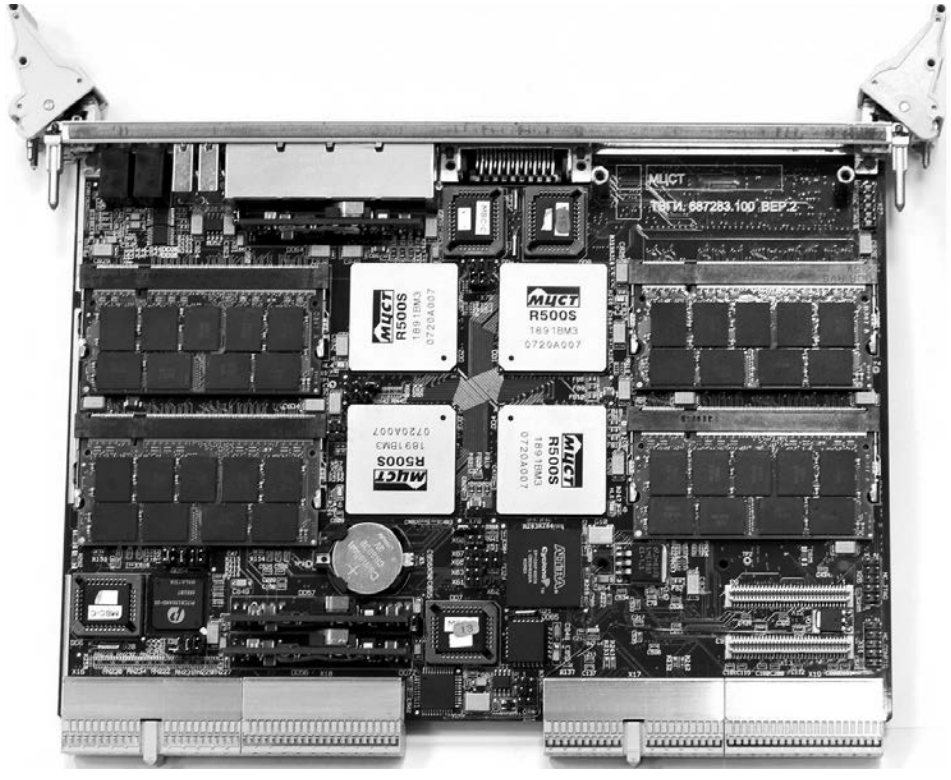


Рис. 2.29. Внешний вид процессорного модуля МВС/С

на кристалле МЦСТ-R500S, ПЗУ начальной загрузки и блок оперативной памяти емкостью до 2 Гбайт.

Машины соединены контроллерами RDMA в конфигурации «кольцо» через дуплексные байтовые каналы с пропускной способностью в одном направлении 667 Мбайт/с. Суммарная пропускная способность обоих контроллеров 2667 Мбайт/с. Контроллер может работать в режиме DMA, обеспечивая каналу доступ в оперативную память через системный коммутатор, или в режиме BYPASS, обеспечивая транзитную передачу данных между двумя каналами межсистемного обмена.

Для обеспечения ввода-вывода модуля используются периферийные контроллеры только одной системы на кристалле МЦСТ-R500S. Кроме того, процессорный модуль имеет два посадочных места на шине PCI для установки дополнительных контроллеров. Основные технические характеристики процессорного модуля МВС/С представлены в табл. 2.11.

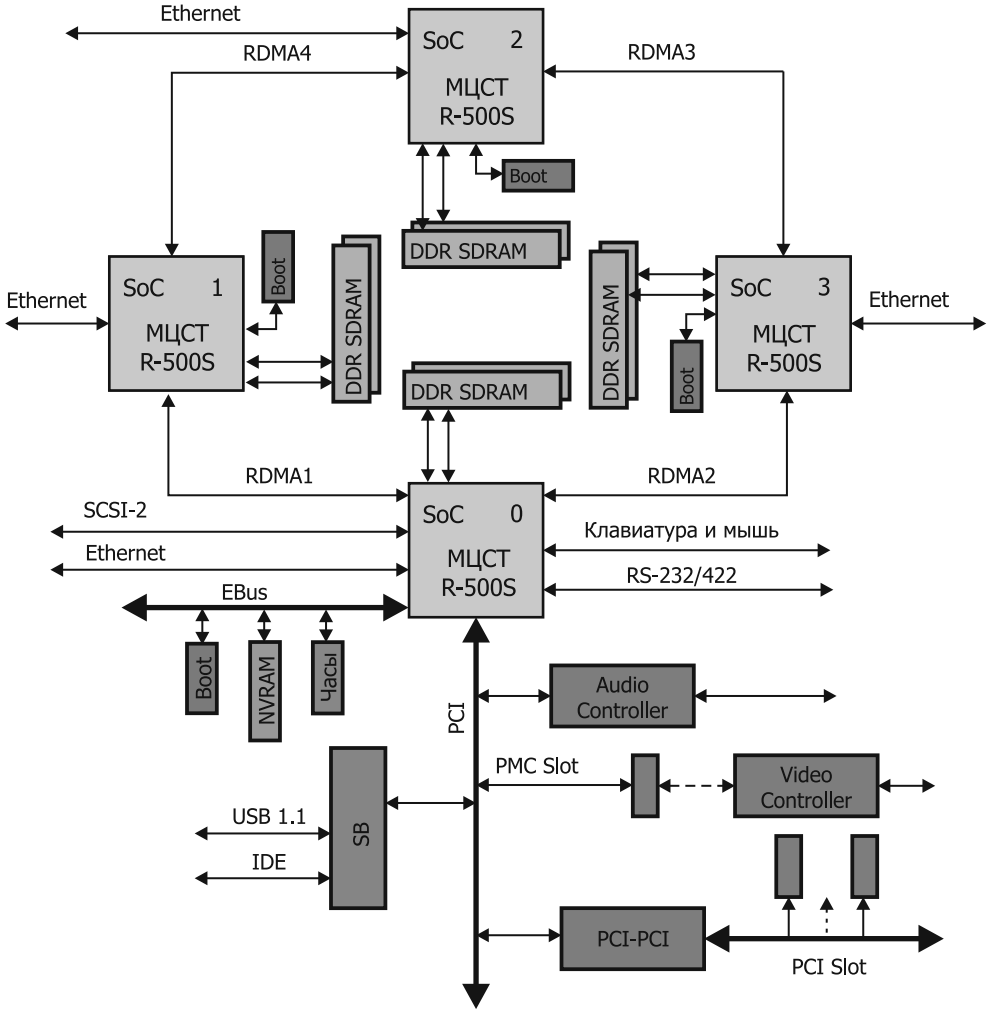


Рис. 2.30. Структурная схема процессорного модуля МВС/С

Таблица 2.11. Технические характеристики процессорного модуля МВС/С

Параметр	**Значение**
Количество машин	4
Количество процессоров	В модуле — 8; в машине — 2
Производительность	Модуля — 4400 MIPS/1600 MFLOPS; машины — 1190 MIPS/400 MFLOPS

Параметр	Значение
Память	Объем: модуля — до 8 Гбайт; машины — до 2 Гбайт Пропускная способность канала обмена 4 · 2,664 Гбайт/с
Флеш-память	512 Кбайт (OpenBootPROM, стандарт IEEE 1275-1994)
Часы и NVRAM	8 Кбайт NVRAM с автономным питанием (от батарейки)
Периферийная шина PCI	Количество слотов 8; пропускная способность шины 264 Мбайт/с
SCSI-2	Пропускная способность шины 10 Мбайт/с
Ethernet 100	Количество каналов 4; пропускная способность канала 100 Мбит/с
RS-232	Количество каналов 2; пропускная способность канала 115 Кбит/с
Мышь и клавиатура	Пропускная способность канала 5 Мбайт/с
USB 1.1	Пропускная способность канала 12 Мбит/с
IDE	Пропускная способность канала 33/66 Мбайт/с
Аудиоконтроллер	Контроллер PCICS4281, кодек CS4297A
Видеоконтроллер (PMC-модуль)	SVGA, truecolour, 1600×1200, 32 бита на пиксел, 2D-ускоритель

2.5. Система на кристалле МЦСТ-R1000

Система на кристалле МЦСТ-R1000 (в процессе разработки имела обозначение МЦСТ-4R) предназначена для использования в многопроцессорных системах с распределенной когерентной общей оперативной памятью [19], рассчитанных на высокие показатели производительности. Основные решения, принятые в этом направлении, состоят в реализации четырехъядерной процессорной части с архитектурой SPARC.v9 [20] и переходе на тактовую частоту 1 ГГц. Важной новацией является введение трех быстрых каналов межсистемного обмена, позволяющих строить четырехпроцессорные вычислительные системы простым соединением каналов. Системы с большим количеством процессоров могут быть реализованы при использовании дополнительного коммутатора.

В архитектуру системы также внесен ряд оптимизаций, способствующих увеличению производительности. В их числе надо особо отметить решения, обеспечивающие наполненность потока инструкций, который поступает в процессоры в фазе дешифрации [21].

Структура и технические характеристики системы на кристалле МЦСТ-R1000 представлены на рис. 2.31 и в табл. 2.12 соответственно. Структура микросхемы содержит:

- четыре процессорных ядра — CPU0—CPU3;
- контроллер кэш-памяти второго уровня L2CacheControl;
- общий кэш второго уровня L2Cache;
- контроллер когерентности CC;
- системный коммутатор SCom;
- контроллер оперативной памяти MC;
- контроллер канала ввода-вывода IOCC;
- три контроллера каналов межсистемного обмена — ISCC0, ISCC1 и ISCC2.

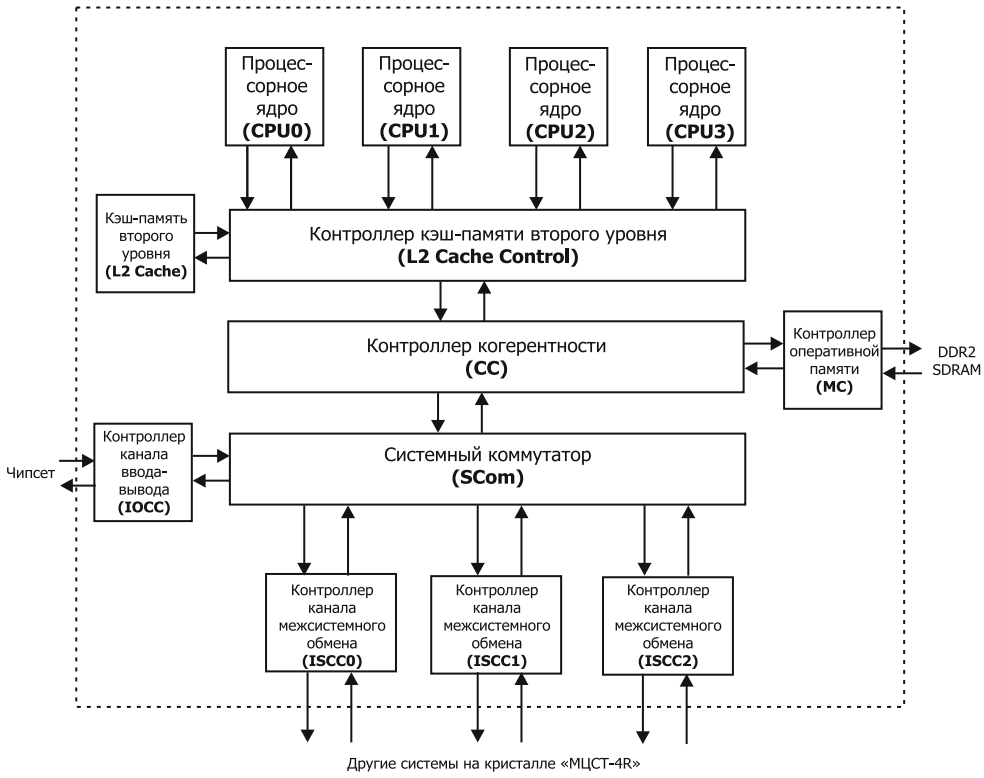


Рис. 2.31. Структурная схема системы на кристалле МЦСТ-R1000

Таблица 2.12. Технические характеристики системы на кристалле МЦСТ-R1000

Параметр	**Значение**
Процессорное ядро	SPARCv9 + VIS2
Тактовая частота, ГГц	1
Количество процессорных ядер	4
Производительность одного ядра: — Dhrystone, GIPS; — 32 разряда, GFLOPS; — 64 разряда, GFLOPS	 2 4 2
Кэш команд первого уровня (одно ядро), Кбайт	16 (I)
Кэш данных первого уровня (одно ядро), Кбайт	32 (D)
Кэш второго уровня (общий), Мбайт	2
Пропускная способность канала ОЗУ, Гбайт/с	4
Тип ОЗУ	DDR2-800
Пропускная способность канала ввода-вывода, Гбайт/с	2
Пропускная способность канала межпроцессорного обмена (3 канала в микросхеме), Гбайт/с	4
Потребляемая мощность, Вт, не более	20
Количество транзисторов, млн шт.	180
Напряжение питания, В	1,0/1,8/2,5
Тип корпуса/количество выводов	HFCBGA/1156
Технология	90 нм, 10 слоев металла
Площадь кристалла, мм^2	128

Процессорные ядра CPU0—CPU3 реализуют 64-разрядную архитектуру SPARCv9 и имеют суперскалярную организацию. Максимальный темп дешифрации команд в процессорном ядре — 2 команды за такт. Обмен с кэшем второго уровня для каждого процессорного ядра выполняется блоками по 32 байта (4 слова параллельно) и выполняется на рабочей частоте ядра 1000 МГц.

Контроллер кэш-памяти второго уровня L2CacheControl управляет работой кэш-памяти второго уровня L2Cache.

Кэш второго уровня L2Cache является общим для четырех процессорных ядер и имеет емкость 2 Мбайт. Он организован в виде 8 колонок по 4096 строк, размер кэш-блока — 64 байта данных.

Контроллер когерентности СС обеспечивает согласованность данных в многопроцессорных системах (в том числе с несимметричным доступом к памяти), построенных на базе систем на кристалле МЦСТ-R1000.

Системный коммутатор SCom обеспечивает доступ в оперативную память процессорных ядер, контроллера канала ввода-вывода IOCC и трех контроллеров межсистемного обмена ISCC0—ISCC2.

Контроллер памяти МС обеспечивает доступ к двум слотам оперативной памяти типа DDR2SDRAM с общим объемом до 8 Гбайт. Обмен с памятью выполняется на частоте 250 МГц, пропускная способность канала — 9×2 байт в каждый такт обмена.

Контроллер канала ввода-вывода IOCC обеспечивает обмен с подсистемой ввода-вывода (контроллером «южного моста») или другими вычислительными комплексами.

Контроллеры каналов межсистемного обмена ISCC0—ISCC2 предназначены для связи с другими системами на кристалле МЦСТ-R1000. Каждый контроллер удаленного доступа имеет дуплексный байтовый LVDS-канал. Обмен выполняется по методу DDR (DoubleDataRate) на частоте 500 МГц. Пропускная способность одного канала в одном направлении 1×2 байт каждый такт, суммарная пропускная способность контроллера 2000 Мбайт/с.

Различия между контроллерами IOCC и ISCC связаны в основном со спецификой пакетов данных и сигнальных сообщений, передаваемых в канале обмена. По каналу ввода-вывода IOC выполняется передача данных для периферийных устройств (массивы данных или отдельные команды). По каналу межсистемного обмена ISC выполняется доступ в память к другим системам и от других систем на кристалле МЦСТ-R1000.

2.6. Вычислительные системы на базе микросхем МЦСТ-R1000

Четырехпроцессорная вычислительная система на базе микросхем МЦСТ-R1000 может быть образована простым соединением каналов межсистемного обмена. При этом к каждому процессору может быть подключена секция оперативной памяти, которая составляет часть когерентной распределенной общей оперативной памяти всей вычислительной системы. Каналы ввода-вывода каждой микросхемы могут использоваться для подключения

доступной остальным подсистемы ввода-вывода и организации многомашинных вычислительных систем.

Структура на базе четырех микросхем МЦСТ-R1000 представлена на рис. 2.32.

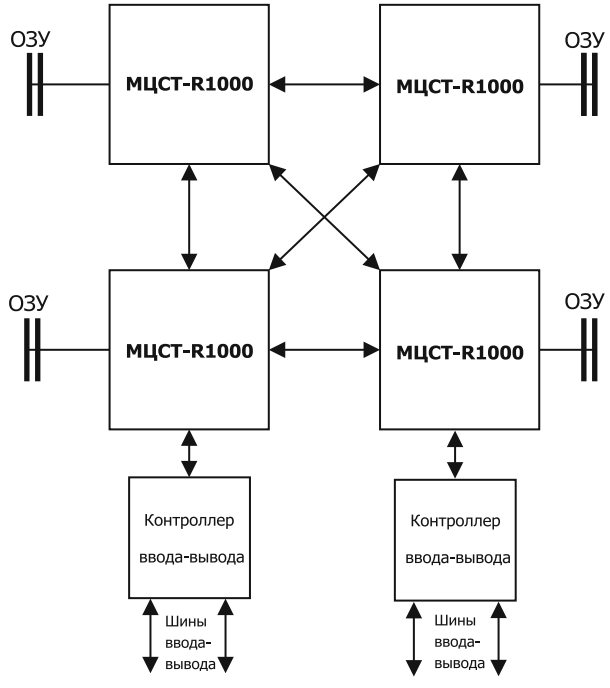
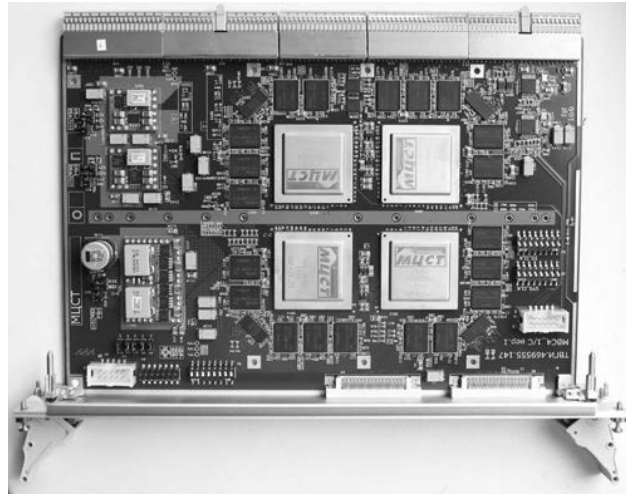


Рис. 2.32. Вычислительная система на базе микросхем МЦСТ-R1000

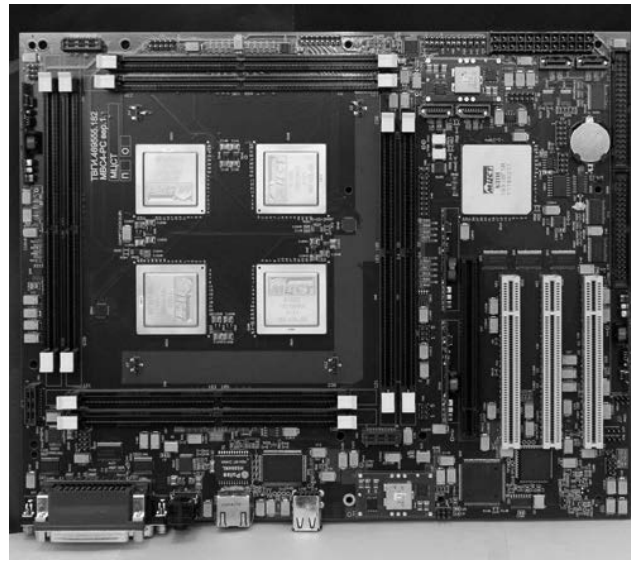
Эта вычислительная система реализована в двух типах процессорных модулей, разработанных ЗАО «МЦСТ». Модули процессорные МВС4/С и МВС4-РС с архитектурной платформой, унифицированной с ВК «Эльбрус-90микро», обеспечивают многопроцессорные, многопользовательские и многопрограммные вычисления в режиме жесткого реального времени. Реализация этих возможностей поддерживается общим программным обеспечением, включающим операционные системы ОС «Эльбрус» и ОС МСВС 3.0, оптимизирующий компилятор с языков С/С++, высокопроизводительные математические библиотеки и ряд других компонентов.

Входящий в состав модулей набор специальных и стандартных интерфейсов, поддержанных средствами ОС, позволяет создавать на их основе

вычислительные системы различной производительности, функциональности и надежности. Внешний вид и технические характеристики модулей приведены на рис. 2.33, *а* и *б*, и в табл. 2.13 и 2.14 соответственно.



а



б

Рис. 2.33. Модули: *а* — МВС4/С; *б* — МВС4-РС

Таблица 2.13. Технические характеристики модуля МВС4/С

Параметр	**Значение**
Количество микросхем МЦСТ-R1000/процессорных ядер, шт.	4/16
Производительность модуля, MIPS	24 992,4
Архитектура процессоров	SPARCv9
Объем оперативной памяти, Гбайт	4
Объем постоянной памяти, Кбайт	512
Объем флеш-памяти, Гбайт	1
Канал ввода-вывода IOC Пропускная способность в одном направлении, Мбайт/с	Дуплексный 1000
Пропускная способность шины PCI, Мбайт/с	264
Пропускная способность канала IEEE1284, Мбайт/с	5
Внешние каналы ввода/вывода	RS-232, USB 2.0, Ethernet 1000/100/10, SATA
Конструктив	«Евромеханика-6U» с системной шиной CompactPCI на двух системных платах

Таблица 2.14. Технические характеристики модуля МВС4-РС

Параметр	**Значение**
Количество микросхем МЦСТ-R1000/процессорных ядер, шт.	4/16
Производительность модуля, MIPS	25 004,3
Архитектура процессоров	SPARCv9
Объем оперативной памяти, Гбайт	16
Объем постоянной памяти, Кбайт	512
Объем флеш-памяти, Гбайт	1
Канал ввода-вывода IOC Пропускная способность в одном направлении, Мбайт/с	Дуплексный 1000
Пропускная способность шины PCI, Мбайт/с	264
Пропускная способность шины PCIExpress, Мбайт/с	500
Пропускная способность канала IEEE 1284, Мбайт/с	5
Внешние каналы ввода/вывода	RS-232, USB 2.0, Ethernet 1000/100/10, SATA
Конструктив	ATXPC на одной плате

Контрольные вопросы и задания

1. Перечислите основные свойства архитектуры микропроцессора МЦСТ-R500.
2. Поясните назначение регистровых окон. Сколько регистровых окон поддерживается в микропроцессоре МЦСТ-R500?
3. Перечислите основные узлы микропроцессора МЦСТ-R500 и кратко охарактеризуйте их назначение.
4. Какие конвейеры выполнения команд реализованы в микропроцессоре МЦСТ-R500? Назовите стадии выполнения команд каждого из конвейеров.
5. С какой целью используются шины байпаса?
6. Перечислите основные узлы буфера команд и кратко охарактеризуйте их назначение.
7. Охарактеризуйте кэш команд первого уровня. Поясните, как происходит обращение к нему.
8. Поясните назначение и объясните принципы работы механизма старения кэша команд первого уровня.
9. Сформулируйте назначение устройства управления. Кратко охарактеризуйте назначение его основных узлов.
10. Сформулируйте назначение АЛУ целых и поясните принципы его работы.
11. Какие команды выполняются в АУ вещественных? Поясните принципы его работы.
12. Перечислите функции, выполняемые устройством управления памятью.
13. Поясните принципы преобразования виртуальных адресов в физические.
14. Охарактеризуйте кэш данных первого уровня. Поясните, как происходит обращение к нему.
15. Поясните назначение и объясните принципы работы механизма старения кэша данных первого уровня.
16. С какой целью в кэше данных первого уровня используется буфер записи?

17. Поясните назначение и организацию кэша таблицы страниц.
18. Сформулируйте принципы работы механизма старения кэша таблицы страниц.
19. Охарактеризуйте внешний кэш. Поясните, как к нему происходит обращение.
20. Перечислите основные узлы контроллера внешнего кэша и шины MBus и кратко охарактеризуйте их назначение.
21. Назовите основные технические характеристики микропроцессора МЦСТ-R500.
22. Перечислите основные узлы ВК «Эльбрус-90микро» и кратко охарактеризуйте их назначение.
23. Назовите основные технические характеристики ВК «Эльбрус-90микро».
24. Перечислите внутренние шины, используемые в ВК «Эльбрус-90микро». Поясните их назначение.
25. Поясните назначение устройства связи с внешними абонентами. Назовите, какое максимальное количество SBus-ячеек можно подключить, используя это устройство.
26. Перечислите основные ячейки, устанавливаемые в устройство связи с внешними абонентами, кратко охарактеризуйте их назначение.
27. Перечислите основные конструктивные исполнения ВК «Эльбрус-90микро» и области их применения.
28. Перечислите основные узлы системы на кристалле МЦСТ-R500S и кратко охарактеризуйте их назначение и технические характеристики.
29. Охарактеризуйте процессорный модуль МВС/С. Какой тип межпроцессорных связей в нем использован?
30. Перечислите основные узлы системы на кристалле МЦСТ-R1000 и кратко охарактеризуйте их назначение и технические характеристики.
31. Охарактеризуйте процессорные модули МВС4/С и МВС4-РС. Как в них организовано взаимодействие микросхем МЦСТ-R1000?

ГЛАВА 3

Микропроцессоры с архитектурой «Эльбрус», вычислительные средства на их основе

3.1. Поддержка определяющих свойств архитектуры «Эльбрус» в аппаратуре микропроцессора

Приведенные в главе 1 свойства новой архитектуры были достигнуты благодаря ряду решений, которые введены в базовые составляющие вычислительных систем — аппаратуру, операционную систему и компилятор, спроектированные и действующие совместно. Вклад каждой из них в общий результат рассматривается в главе 4. Там же дается развернутое описание программных составляющих. В данном разделе главы 3 характеризуются принципиальные элементы архитектуры «Эльбрус» в аппаратуре. В последующих разделах они будут отмечаться в составе рассматриваемых устройств.

3.1.1. Широкое командное слово

Принципиальной особенностью архитектуры «Эльбрус» является возможность при компиляции каждого фрагмента программы предопределить максимальное распараллеливание вычислительного процесса на всем поле доступных аппаратных ресурсов, которая базируется на использовании широкого командного слова. Соответственно, в общепринятой классификации архитектуру «Эльбрус» можно отнести к категории VLIW (Very Long Instruction Word) [22], [15].

Широкое командное слово, или широкая команда (ШК), содержит набор операций (с их адресными, литеральными и функциональными параметрами), которые одновременно дешифруются и параллельно выполняются, каждая в своем отдельном конвейере. Это принципиальный фактор реализации параллелизма, свойственного данному программному коду. Микропроцессор «Эльбрус» имеет шесть каналов для выполнения арифметико-логических операций. Помимо них в одной ШК могут быть заданы операции и других типов с фиксированными временами выполнения, что позволяет статически, во время трансляции, планировать параллельную работу исполнительных устройств.

Разрядность объектов архитектуры определена в следующих единицах: байт, полуслово (16 разрядов), одинарное слово, или просто слово (32 разряда), двойное слово (64 разряда), квадрослово (128 разрядов).

Широкая команда выровнена по границе двойного слова и имеет длину от 1 до 8 двойных слов. Она состоит из слогов длиной 4 байта и полуслогов

длиной 2 байта, которые по отдельности или в комбинациях кодируют операции. Перечень типов слогов и полуслогов с указанием их количества в составе ШК представлен в табл. 3.1.

Таблица 3.1. Типы и возможное количество слогов в составе ШК

Обозначение	Наименование	Количество слогов в составе широкой команды
HS	Слог-заголовок	1
ALS	Слоги арифметико-логических каналов	До 6
ALES	Полуслоги расширения для арифметико-логических каналов	До 4
CS	Слоги команд управления	До 2
CDS	Слоги условного исполнения	До 3
PLS	Слоги каналов обработки логических предикатов	До 3
LTS	Слоги литералов	До 4
AAS	Полуслоги каналов обращения к элементам массивов	До 6
SS	Слог коротких операций	1

Слог-заголовок HS задает структуру и длину широкой команды — параметры, которые используются при ее распаковке и дешифрации в конвейере процессора.

Слог CDS содержит указание на то, какая из операций в этой же команде выполняется условно и под управлением какого предиката.

Слог PLS задает операцию обработки логической величины — предиката, считываемого из файла предикатов. Результат также помещается в файл предикатов.

Слог LTS содержит литеральное значение, которое может использоваться арифметико-логическим каналом в качестве операнда. Каждый слог может содержать либо 32-разрядное знаковое значение, либо половину 64-разрядного значения.

Полуслог AAS кодирует операцию, которая пересылает элементы массивов, предварительно подкачанные в буфер, в регистровый файл для обработки.

Слог SS предназначен для операций, кодировка которых занимает всего несколько разрядов, например передачи управления или продвижения вращающейся базы регистрового файла.

Заголовок HS всегда присутствует в команде в качестве ее первого слога, наличие других слогов зависит от назначения команды. Определен следующий порядок размещения слогов: HS, ALS0, ALS1, ALS2, ALS3, ALS4, ALS5, CS0, CS1, SS, ALES0, ALES1, ALES3, ALES4, AAS0, AAS1, AAS2, AAS3, AAS4, AAS5, LTS3, LTS2, LTS1, LTS0, PLS2, PLS1, PLS0, CDS2, CDS1, CDS0.

Кодировка операции может занимать часть слога, слог или состоять из нескольких слогов. Так, слог ALS определяет операцию арифметико-логического канала над операндами из регистрового файла или с использованием короткого литерала в качестве операнда. Добавление слога (слогов) LTS позволяет задать литералы длиной 2 байта, 4 байта или 8 байт. Добавление полуслога ALES позволяет определить комбинированную (составную) операцию или расширить код операции. Добавление слога CDS позволяет задать условный режим выполнения операции, представленной в слоге ALS, и предикат, определяющий возможность выполнения заданной операции.

Операции скалярного обращения в память кодируются также в слогах ALS. Добавление слога CS определяет режимы выполнения обращения в память: без поиска в кэше, без трансляции виртуального адреса в физический адрес, специальные операции в кэше и таблицы страниц типа откачки и чистки, обращения к регистрам подсистемы памяти и т. д.

Слог коротких операций SS, наоборот, позволяет задать сразу несколько коротких операций. В их число, например, входят операция передачи управления (задаются номер регистра подготовки передачи управления и адрес предиката для условной передачи управления) и операции продвижения текущих баз во вращающихся областях регистрового и предикатного файлов.

3.1.2. Средства аппаратной поддержки параллельных вычислений

Архитектура «Эльбрус» включает ряд универсальных решений, свойственных современным высокопроизводительным микропроцессорам.

Регистровый файл. Параллельное исполнение операций по сравнению с последовательным требует необходимого количества оперативных регистров.

Архитектура определяет регистровый файл объемом 256 регистров для целочисленных и вещественных данных, 32 регистра предназначены для глобальных данных и 224 регистра — для стека процедур. Процедура может использовать регистровое окно произвольного размера (до 224). Регистровые окна процедур организованы в виде стека, имеющего продолжение в памяти; переполнение/исчерпание регистрового файла вызывает автоматическую откачку/подкачку. Для поддержки вычислений в цикле к произвольной области в окне может быть применена относительная циклическая адресация.

Предикатный файл. Имеет объем 32 двухразрядных регистра. Процедура может использовать все 32 предиката. Предикаты процедур сохраняются вместе с информацией для возврата, организованной в виде стека, вершина которого расположена в регистровом файле связующей информации, а продолжение — в оперативной памяти. Переполнение/исчерпание файла связующей информации вызывает его автоматическую откачку/подкачку.

Спекулятивный режим выполнения команд. Параллельному выполнению команд препятствуют не определяемые при компиляции зависимости по управлению и зависимости по данным. Выполняя операции раньше, чем становится известно направление условного перехода, или считывая данные из памяти раньше предшествующей записи в память, можно ускорить выполнение программы. Но подобное перемещение операций не всегда допустимо из-за неопределенности их поведения при исполнении. В первом случае (выполнение раньше условного перехода) операция, которая не должна выполняться, может вызвать прерывание. Во втором случае может быть считано неправильное значение, если адреса совпадут.

Архитектура «Эльбрус» определяет спекулятивный режим выполнения, в котором факт прерывания фиксируется, но реакция на него откладывается до продолжения выполнения ветви программы в неспекулятивном режиме. Дополнительно для операций обращения в оперативную память вводится частично ассоциативная память (кэш-память), которая позволяет определять нарушения заданного в программе порядка обращения с совпадающими адресами по чтению и записи. Факты нарушений фиксируются и проверяются в неспекулятивном режиме. При обнаружении нарушений выполняются повторные чтения из памяти и, если это необходимо, операции, которые уже использовали результат неверного чтения, отменяются.

Подготовка передачи управления. Предварительная подкачка кода в направлении ветвления, а также его первичная обработка на дополнительном конвейере (на фоне выполнения основной ветви) скрывают задержку по

доступу к коду программы при передачах управления и тем самым позволяют выполнить передачу управления без остановки конвейера выполнения, когда уже известно условие ветвления. Архитектура микропроцессора определяет средства предварительной подкачки кода для трех команд передачи управления.

Предварительная подкачка массивов. Обращения к массивам, локализованным в оперативной памяти, играют существенную роль в векторных вычислениях. Это в первую очередь обусловлено большим временем доступа и колебаниями его значения. Кроме того, требуется на длительное время резервировать ячейки регистрового файла под вызываемые элементы массивов, что снижает эффективность использования регистров. В архитектуре определен буфер предварительной подкачки, в который данные из массивов вызываются отдельной (асинхронной) ветвью программы, выполняемой параллельно на фоне вычислений. Определены средства управления, позволяющие подстраивать опережение вызова элементов массивов по отношению к их использованию в вычислениях. Вызванные элементы массивов пересылаются в регистровый файл из основной ветви программы непосредственно перед их использованием.

Наряду с универсальными механизмами архитектура «Эльбрус» характеризуется решениями, которые связаны с реализацией преимуществ, обусловленных использованием широкого командного слова.

Расширение традиционного набора команд. Операции с упакованными значениями, которые определены для целых и вещественных данных, используют операнды и формируют результат в формате двойного слова (8 байт). В упакованном формате могут быть представлены байты и полуслова (только для операций над целыми значениями), одинарные слова и двойные слова. Операции с упакованными значениями повышают производительность за счет параллельного выполнения индивидуальных операций применительно ко всему набору упакованных данных (например, для упакованных байтов одновременно выполняются 8 операций).

Комбинированные трехоперандные операции задают две последовательно выполняемые арифметические и логические операции. Первая из них использует два операнда, а вторая — результат первой операции и третий операнд. Использование комбинированных операций повышает пропускную способность конвейера выполнения команд.

Выполнение циклов методом программного конвейера. Этот механизм является действенным средством ускорения векторных вычислений.

Последовательные итерации цикла выполняются с некоторым наложением друг на друга. Шаг, с которым накладываются итерации, определяет общий темп их выполнения, он может быть существенно выше, чем при строго последовательном режиме. В каждом шаге программного конвейера выполняются операции из разных наложенных итераций, и их количество может быть большим. В этом смысле использование широкой команды имеет очевидное преимущество. Архитектура микропроцессора содержит средства управления режимами выполнения заголовка цикла и конца цикла, позволяющие единым образом программировать выполнение всего цикла. В регистровом и предикатном файлах определены области для загрузки элементов массивов и предикатов в цикле с базами, определяющими начало данных для текущей итерации цикла. Это позволяет обращаться к данным различных итераций цикла, используя разные смещения относительно базы цикла, что необходимо, поскольку в широкой команде могут присутствовать команды, относящиеся к разным итерациям.

Предикатный режим выполнения команд. В этом режиме до вычисления условия могут исполняться обе ветви, но после вычисления результаты выполнения «неправильной» ветви в ее конвейере аннулируются. Для реализации этого принципа используется дополнительный операнд — предикат, который разрешает либо отменяет исполнение. Архитектура микропроцессора определяет в каждой широкой команде до 6 предикатов, которые могут управлять выполнением 8 операций.

3.1.3. Двоичная совместимость с Intel x86

Техника двоичной трансляции сводится к преобразованию двоичных кодов исходной архитектуры в функционально эквивалентные последовательности кодов целевой архитектуры, впоследствии исполняемые на аппаратуре целевой платформы. При этом, в отличие от такого распространенного метода обеспечения двоичной совместимости, как покомандная интерпретация, двоичная трансляция способна обеспечить довольно высокую эффективность исполнения исходных кодов за счет оптимизации, сохранения и возможности многократного исполнения единожды оттранслированных целевых кодов. Программные алгоритмы оптимизации обеспечивают просмотр значительно более крупных регионов кодов по сравнению с «окном» распараллеливания операций в суперскалярных архитектурах и позволяют задействовать всю номенклатуру исполнительных устройств.

Аппаратные архитектурные свойства, которые используются системой двоичной трансляции, подразделяются на три группы:

- стандартные средства архитектуры, обеспечивающие совместимость;
- специальные средства, обеспечивающие совместимость и производительность;
- стандартные средства, повышающие эффективность данной системы.

Стандартные средства архитектуры, обеспечивающие совместимость, включают следующие компоненты:

- комплект операций целочисленной арифметики, совместимых с x86;
- комплект 80-разрядных вещественных операций, совместимых с x87, с соответствующими управляющими и статусными регистрами и системой прерываний;
- 80-разрядный регистровый файл;
- комплект целочисленных операций, совместимых с командами для обработки мультимедийных данных MMX (Multi Media Extension);
- комплект 32/64-разрядных скалярных и вещественных операций, совместимых с соответствующими управляющими и статусными регистрами и системой прерываний.

Средства, включенные в первую группу, являются неотъемлемым базисом для надежной и производительной работы системы двоичной трансляции и используются всеми ее компонентами.

Специальные средства архитектуры, обеспечивающие как совместимость, так и производительность:

- специализированный набор арифметических операций для выработки флагов x86;
- второе, совместимое с платформой x86, виртуальное пространство:
- комплект сегментных регистров для обращений в память;
- комплект операций обращения в память, использующих сегментные регистры и второе виртуальное пространство;
- механизм слежения за «неожиданными» модификациями памяти;
- механизм защиты (сокрытия) области памяти, занятой двоичнотранслирующей системой;

- комплект глобальных регистров;
- имитация стека регистров x87 (сопроцессора) — относительная адресация, прерывания по битам значимости;
- операции обработки флагов вещественной арифметики;
- специализированный управляющий регистр, модифицирующий режимы и алгоритмы исполнения операций;
- специализированная кэш-память адресов памяти, контролирующая корректность оптимизаций двоичного транслятора при спекулятивном выполнении команд считывания раньше записей по неизвестным адресам;
- операции и регистры статуса, предназначенные для эффективной реализации контрольных точек;
- специализированная кэш-память таблиц;
- специализированное средство «раскраски» исполняемого кода (бит в заголовке широкой команды);
- механизм отложенных прерываний.

Средства, включенные во вторую группу, дают возможность существенно увеличить эффективность исполнения исходного кода, разгружая последний от избыточных динамических проверок и разрешая использование техники агрессивных оптимизаций двоично-транслированного кода без потери надежности исполнения. Операции выработки флагов позволяют не прибегать в коде к сложным программным механизмам пересчета их значений, что особенно важно для обеспечения эффективной работы начальных уровней многоуровневой двоично-транслирующей системы (интерпретатор и шаблонный компилятор).

Совместимые с платформой x86 модель сегментации памяти и вторичное пространство позволяют разгрузить операции работы с памятью от сложных динамических проверок и ускорить работу с памятью, что особенно критично для кодов платформы x86, обыкновенно изобилующих обращениями в память из-за дефицита рабочих регистров. Механизм слежения за модификациями памяти реализует простой, эффективный и надежный метод поддержания когерентности двоично-транслированных кодов, предоставляя в распоряжение двоично-транслирующей системы средство динамического распознавания попыток модификации памяти, содержащей

ранее транслированные участки исходного кода. Механизм сокрытия двоично-транслирующей системы позволяет поддерживать иллюзию монопольного владения машиной исходными кодами, аппаратно помогая скрывать наличие в системе областей оперативной памяти, задействованных под собственно двоично-транслирующую систему.

Комплект глобальных регистров в регистровом файле, включая область с имитацией правил доступа к вещественному стеку платформы x87, используется для оптимального отображения регистрового контекста исходной платформы. Он особенно важен для эффективной реализации неоптимизированных средств исполнения исходного кода и динамической поддержки сложных операций.

Таблица блокировок памяти применяется в оптимизированном двоично-транслированном коде для контроля надежности принимаемых транслятором решений по переносу работы с переменными из памяти на регистры. Ограниченность регистрового файла архитектуры x86 приводит к необходимости хранить большое число локальных переменных и параметров в оперативной памяти, что негативно влияет на скорость работы получаемого кода. Применение механизма блокировок памяти позволяет двоичному транслятору исправлять эту неэффективность путем исключения из кода лишних операций работы с памятью, при котором рабочие значения переносятся на локальные регистры без опасения, что будет потеряна надежность исполнения при возможном возникновении динамических конфликтов по адресам.

Механизмы эффективной реализации контрольных точек позволяют обеспечивать поддержку точного прерывания в оптимизированных кодах с минимальными потерями эффективности исполнения последних. Аппаратурой предоставляется возможность отслеживать прохождение двоично-транслированным кодом контрольных точек с учетом динамики исполнения (отмены операций) и в случае возникновения исключительной ситуации надежно определять последнюю успешно пройденную контрольную точку.

Кэш-память таблиц применяется двоично-транслирующей системой для хранения пар соответствия точек входа в исходном и двоично-транслированном кодах, предоставляя возможность быстрой выборки пары по исходному адресу. Она используется также для оптимизации в двоично-транслированном коде таких динамических переходов, целевой адрес которых не может быть надежно предсказан статически.

Средство «раскраски» исполняемого кода применяется при работе с кодами отладочного режима x86, использующими точки останова по

адресу команды. В данном случае «раскраска» позволяет эффективно подсвечивать границы исходных команд в неоптимизированных двоично-транслированных кодах и сохранить надежность исполнения без переключения на режим интерпретации.

Механизм отложенного прерывания позволяет организовать прием внешних асинхронных прерываний оптимальным с точки зрения двоично-транслированного кода образом, когда факт приема прерывания вначале запоминается обработчиком на особом статусном регистре микропроцессора и лишь затем проявляется при помощи специальной операции в подходящих для этого местах двоично-транслированного кода.

Стандартные средства архитектуры, повышающие эффективность системы двоичной трансляции. В числе стандартных средств, поддерживающих исполнение двоично-транслированных кодов, следует отметить:

- кэш команд;
- таблицу перекодировки адресов;
- механизм связывания переходов;
- профилирующие механизмы.

Кэш команд используется для хранения транслированных кодов во время исполнения в памяти системы. В системе реализованы три кэша команд, разделенных по уровню оптимизации хранящихся в них кодов.

Таблица перекодировки адресов используется для задания и определения соответствия между адресами исходного кода и точками входа в транслированный код. Команды передачи управления обращаются к таблице с целью определить, может ли исполнение быть продолжено в транслированном коде или необходимо вызвать интерпретатор. Механизм связывания переходов позволяет отказаться от дорогостоящего (занимающего много времени) обращения к таблице перекодировки адресов и осуществлять переходы между различными транслированными кодами напрямую.

Профилирующие механизмы обеспечивают построение профильного графа в системе, его поддержание в актуальном состоянии и формирование регионов для трансляции быстрым либо оптимизирующим компиляторам на базе накопленной в нем профильной информации. Отметим, что регионы для этих двух компиляторов формируются по несколько отличным правилам с учетом специфики каждого из них.

3.1.4. Поддержка защищенного режима исполнения программ в аппаратуре

Архитектура «Эльбрус» защищает контекст программного модуля, образованный объектами, которые допускается в нем использовать согласно правилам языка программирования. Это делается путем организации доступа к низкоуровневому представлению объекта в памяти или регистрах через «проходную» его дескриптора — служебного слова, содержащего ссылку и информацию об объекте, соответствующую его типу. Доступ возможен только через дескриптор. Аппаратные операции создания и использования дескрипторов специализированы таким образом, что не оставляют возможности непосредственного воздействия на объект.

Принципиальным фактором защиты, реализуемым через аппаратуру, является тегирование. Оно выполняется путем добавления к информационным разрядам дополнительного поля — тега, определяющего тип данных. Каждое 4-байтовое слово в памяти, регистрах и шинах сопровождается 2-разрядным тегом.

Тег слова кодирует следующие признаки:

- 0 — слово содержит числовую информацию либо само по себе, либо являясь фрагментом формата;
- 1 — слово содержит нечисловую информацию с форматом одинарного слова;
- 2 — слово содержит фрагмент нечисловой информации формата двойное слово;
- 3 — слово содержит фрагмент нечисловой информации формата квадро.

Ни длина, ни тип числовых данных архитектурно не различимы. Семантическое наполнение числовой переменной отслеживается компилятором и проявляется, когда она становится операндом какой-либо операции.

В отличие от числовых данных, нечисловые данные строго типизируются. Прежде всего гарантируется целостность составных форматов (двойное слово и квадро), что предполагает следующее:

- все фрагменты должны иметь тип, соответствующий формату;

- при любых манипуляциях фрагменты должны сохранять свой порядок.

Второе качество реализуется выровненным расположением данных как в памяти, так и в регистровом файле микропроцессора. Так, младшее слово адресной переменной с форматом двойного слова всегда помещается по адресу, кратному 8 байтам, а старшее — следом за ним, смежным образом. Пересылки отдельных фрагментов приводят к разрушению их типов (превращению в неспециализированные данные). Операции, требующие специализированных операндов (например, адресных данных), строго контролируют их тип и целостность.

В рамках каждой из форматных групп конкретный тип специализированных данных определяется значением выделенной группы информационных разрядов — внутреннего тега.

Кроме обычных дескрипторов, описывающих регулярные массивы в памяти, введены специальные дескрипторы, позволяющие описывать структуры данных в соответствии с требованиями объектно-ориентированного программирования. Это существенно упрощает доступ к таким данным. Дескриптор объекта содержит описание открытых, защищенных и приватных областей данных.

Регистры контекста выполняемой программы дополнены регистрами индекса текущего модуля компиляции, дескриптора текущего модуля компиляции, дескриптора глобальных данных текущего модуля компиляции, дескриптора типов и текущего типа. Введены операции обращения в память с использованием этих дескрипторов. Все дескрипторы имеют внутренние теги, позволяющие контролировать корректность их использования (только в специальных операциях обращения в память и преобразований в сторону сокращения прав их использования).

Общая характеристика операций микропроцессора «Эльбрус», включающая действия с объектами архитектуры, введенными в разделе 3.1, приведена в приложении 3.

3.2. Микропроцессор «Эльбрус»

3.2.1. Структура

Блок-схема микропроцессора «Эльбрус» представлена на рис. 3.1.

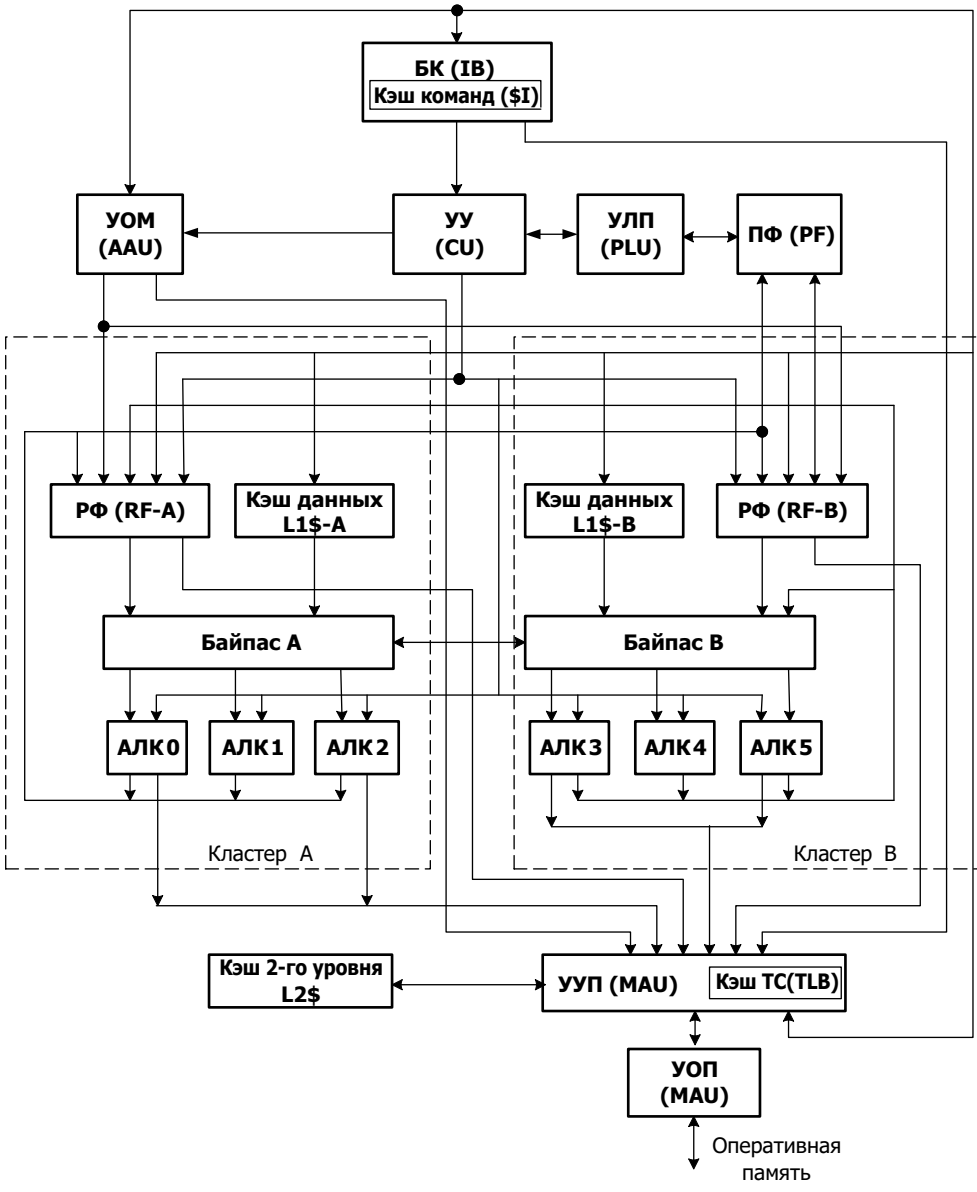


Рис. 3.1. Блок-схема микропроцессора «Эльбрус»

БК — буфер команд (IB — Instruction Buffer) предназначен для вызова программного кода из памяти, буферизации на время, достаточное для обеспечения непрерывности дешифрации, и выдачи его в устройство управления для последующей обработки. Накопление программного (объектного) кода обеспечивает кэш команд I$ емкостью 64 Кбайт.

УУ — устройство управления (CU — Control Unit) выполняет считывание программного кода из буферной памяти БК, распаковку широких команд, их дешифрацию и переключение ветвей программы при выполнении команд переходов. С этой целью в структуре УУ предусмотрены обработка последовательности ШК основной ветви, предварительная подкачка ШК трех ветвей предполагаемого ветвления и распаковка первой из команд каждой ветви. Заметим, что подготовка перехода выполняется на фоне выполнения основной ветви и поэтому не приводит к замедлению вычислительного процесса. С выхода УУ в исполнительные устройства выдается распакованная ШК основного потока или одного из подготовленных потоков (при наличии в основном потоке операции передачи управления с реализовавшимся условием перехода).

ПФ — предикатный файл (PF — Predicate File) хранит первичные предикаты — битовые значения, выработанные операциями сравнения, и вторичные предикаты — результаты логических операций над первичными предикатами.

Предикатный файл — набор 32 двухразрядных регистров (по одному разряду для предиката и тега). С помощью предикатных значений может быть задан режим условного выполнения операции или условий для команд ветвлений.

УЛП — устройство логических предикатов (PLU — Predicate Logic Unit) предназначено для выполнения операций считывания предикатов из файла ПФ и логических операций формирования вторичных предикатов, которые также могут быть записаны в предикатный файл. Эти операции выполняются над малоформатными значениями, поэтому в целях разгрузки арифметико-логических каналов обработка предикатов возложена на УЛП. Ее результаты направляются в устройство управления для задания режима условного выполнения команд, отрабатываемых в арифметико-логических каналах, или выполнения команд передачи управления.

Арифметико-логические каналы (АЛК) предназначены для исполнения обычных арифметических и логических операций, операций обращения к памяти и обработки адресных данных (дескрипторов, указателей и др.). В состав микропроцессора входят шесть арифметико-логических каналов (АЛК0—АЛК5), разделенных на два кластера. Арифметико-логические каналы работают параллельно и исполняют в основном одинаковый набор операций. В качестве операндов служат данные из РгФ или результаты других исполнительных устройств. Не все каналы идентичны, поскольку не все операции выполняются одинаково часто. Например, довольно редкая

операция деления реализована только в АЛК5. Арифметико-логические каналы имеют отдельные устройства для исполнения целочисленных и вещественных операций. Это не относится к целочисленным операциям умножения и деления — для них введены соответствующие блоки в вещественных исполнительных устройствах.

Регистровый файл (РгФ) предназначен для хранения локальных данных процедуры и результатов выполненных операций. Он представляет собой сверхоперативное запоминающее устройство с произвольным доступом, обращение которому осуществляется через порты. Многочисленность абонентов регистрового файла требует большого количества портов для обслуживания всех запросов одновременно. С целью сокращения их количества регистровый файл реализован в виде двух одинаковых блоков — РгФ-А и РгФ-В, по одному в каждом кластере. Причем, в отличие от используемых во многих микропроцессорах, блок является общим для целочисленных и вещественных устройств арифметики, что позволило повысить эффективность его использования. Блоки регистрового файла содержат одни и те же данные, поскольку результат любой операции записывается одновременно в оба блока. Этим обеспечивается когерентность данных — свойство, которое используется при работе с общими данными.

Блок РгФ содержит 256 регистров. При запуске процедуры ей выделяется определенный участок смежных регистров, называемый окном. Активная (выполняемая в данный момент) процедура может обращаться только к регистрам своего окна. Если все регистры окажутся занятыми, автоматически выполняется откачка. В результате откачки данные пересылаются в стек процедур, размещенный в оперативной памяти. Таким образом, регистровый файл служит в качестве аппаратной вершины стека процедуры. Все ранее работавшие, но не завершенные процедуры сохраняют свои окна в регистровом файле. Возврат процедур к своим окнам осуществляется по мере окончания работы запущенных ими процедур. Иными словами, реализуется дисциплина (LIFO) — активизируется та процедура, которая запустила завершившуюся в данный момент времени процедуру.

Чтобы результат, полученный некоторым исполнительным устройством, мог быть использован другим до записи в РгФ, исполнительные устройства связаны шинами байпаса. Благодаря этому в ряде случаев отпадает необходимость в считывании операндов из РгФ. Каждый кластер имеет отдельный блок байпаса, с помощью которого результаты передаются исполнительным устройствам не только своего, но и другого кластера. Кроме того, байпас позволяет использовать результаты считывания из кэш-памяти L1$ до их записи в регистровый файл.

**Кэш данных первого уровня L1$** выполнен в виде двух одинаковых блоков (L1$-A и L1$-B) емкостью 64 Кбайт, по одному в каждом кластере. Блоки L1$ хранят одинаковые данные, поскольку запись данных выполняется одновременно в оба блока. В блоке хранятся данные, которые используются в качестве операндов для исполнительных устройств АЛК. Но поскольку в общем случае операнды считываются из регистрового файла, они должны быть предварительно загружены в него из кэша L1$. Для этих целей предусмотрены операции загрузки, выполняемые исполнительными устройствами АЛК. В случае отсутствия требуемых данных в кэше L1$ операция загрузки продолжается поиском данных в кэш-памяти второго уровня L2$ с последующей записью их в оба блока L1$. Параллельная запись, так же как и в случае регистрового файла, вызвана необходимостью сохранения свойства когерентности данных обоих блоков L1$.

**Кэш второго уровня L2$** является общим для данных и программного кода, его объем составляет 256 Кбайт, степень ассоциативности — 4. Обращение к L2$ выполняется при отсутствии требуемых данных в L1$ или нужного программного кода в буферной памяти команд устройства БК. Если нужная информация отсутствует и в L2$, то формируется запрос к оперативной памяти. Считанная из ОП информация поступает потребителю и одновременно записывается в кэш-память вместо устаревших данных.

УОМ — Устройство обращения к массивам (AAU — Array Access Unit) предназначено для упреждающей подкачки элементов массива при выполнении векторных операций. Поскольку каждый элемент вектора обрабатывается одной и той же последовательностью операций, то обработка идет циклически. В микропроцессоре «Эльбрус» к началу очередного цикла нужный элемент вектора уже считан из памяти (кэша L2$ или ОП) и находится в буфере УОМ. Подкачка элементов массивов в буфер, использующий дисциплину очереди (FIFO), осуществляется на фоне выполнения основной (синхронной) программы параллельной ей (асинхронной) программой.

УУП — устройство управления памятью (MMU — Memory Management Unit) преобразует виртуальные адреса в физические. С целью ускорения этого процесса наиболее часто используемые строки таблицы страниц хранятся в кэше таблицы страниц (TLB) объемом 64 строки. Если нужная строка в нем отсутствует, выполняется аппаратный поиск в таблице страниц, хранящейся в памяти, и загрузка найденной строки в кэш вместо устаревшей. Кроме того, рассматриваемое устройство обеспечивает обращения в кэш L2$ и ОП.

УОП — устройство обращения в память (MAU — Memory Access Unit) предназначено для связи микропроцессора с ОП. Оно содержит буферы операций считывания и записи, позволяющих осуществить потоковое обслуживание заявок. Обмен с памятью осуществляется через 16-байтовый канал с раздельными шинами для передачи и приема данных. Обмен выполняется блоками по 32 или 64 байт.

3.2.2. Конвейер выполнения широких команд

Начальная часть конвейера имеет постоянное число стадий и является общей для всех простых операций, составляющих ШК. Затем конвейер разветвляется по исполнительным устройствам, причем ветви в общем случае содержат различное число стадий.

При оптимальном планировании на этапе компиляции обеспечивается бесконфликтное выполнение операций. Нарушения расписания, вызванные динамическими ситуациями, приводят к задержкам продвижения конвейера на время разрешения конфликтов.

Начальная часть конвейера включает восемь стадий (табл. 3.2 и 3.3), на которых осуществляется обработка программного кода.

Таблица 3.2. Обозначение стадий обработки в конвейере выполнения ШК

Такт	1	2	3	4	5	6	7	8
Станция конвейера (фаза)	АП (A)	ПК0 (F0)	ПК1 (F1)	Расп (S)	Дш (D)	БА (B)	СчО (R)	И0 (E0)

Таблица 3.3. Содержание стадий обработки в конвейере выполнения ШК

Фаза	**Содержание**
АП (A)	Ассоциативный поиск (Associative search). Адреса ШК на входных регистрах БК. Обращение в память тегов (ПТ) команд (ITAG) и формирование адресов строк по буферной памяти команд ПК (IDATA). Обращение в память тегов буфера таблицы страниц команд (БТСК) (Instruction Table Look-aside Buffer, ITLB) и формирование адреса по ответной части
ПК0 (F0)	Память команд 0 (Fetch 0). Регистры адресов в ПК (IDATA), формирование адресов обращения для банка памяти ПК. Обращение в память ответной части БТСК (ITLB) и формирование физического адреса строки
ПК1 (F1)	Память команд 1 (Fetch 1). Входные регистры в банках памяти команд ПК и выборка строки программного кода

Фаза	Содержание
Расп (S)	Распаковка (scattering). Распаковка ШК основной ветви и подготавливаемых ветвей перехода
Дш (D)	Декодирование (Decode) входных ШК всех потоков. Выборка ШК из 4 потоков для выполнения. Дешифрация операций, вычисление адресов операндов
БА (B)	Базирование (Branch and Basing). ШК на регистре B устройства управления. Базирование адресов операндов и передача их на регистры адресов РгФ. Проверка условий передачи управления и переключение на дешифрованную ШК ветви перехода
СчО (R)	Чтение (Read). Чтение операндов из РгФ
И0 (E0)	Исполнение (Execute). Операции и данные в целых исполнительных устройствах. Начало выполнения операции в исполнительном устройстве

Параллельные ветви конвейера рассматриваются в части, посвященной АЛК. Сейчас лишь отметим, что исполнение операции заканчивается записью результата в регистровый, предикатный файл или выдачей конечной информации в память.

В случае возникновения конфликтной ситуации (неготовности операндов, промах в кэш таблицы страниц и др.) устройство управления организует повторную выдачу в АЛК широкой команды, содержащей невыполненную операцию. Повтор может выполняться многократно вплоть до разрешения конфликта. Таким образом, УУ фактически восстанавливает порядок выполнения программы, нарушенный динамической ситуацией.

3.2.3. Буфер команд

Назначение

Основная функция БК — создание запаса программного кода, обеспечивающего непрерывность работы конвейера. Изначально программный код, состоящий из последовательности широких команд, поступает в БК из оперативной памяти, при этом он записывается в кэш L2$ на случай необходимости повторного использования. Из БК компоненты программного кода выдаются в устройство управления по его запросам, каждый из которых включает адрес широкой команды. Ввиду того что устройство управления кроме обработки основного потока осуществляет предварительную подготовку перехода на одну из трех ветвей, вместе с этим адресом выдается номер дополнительного конвейера, куда необходимо направить объектный код.

В общем случае при формировании адреса обращения к памяти необходимо учитывать размер адресуемого элемента или структуры данных. С этим тесно связано понятие выравнивания. Поскольку наименьшей единицей обмена с памятью является один байт, то и адресация в общем случае осуществляется с точностью до байта (или в терминах байта). Но в современных процессорах обрабатываются слова большей разрядности. В частности, в процессоре «Эльбрус» кроме 8-разрядных значений используются 16, 32 и 64 значения и 128-разрядные структуры (квадрослова). Для их хранения выделяется уже несколько смежных ячеек ОП. Так, двойное (64-разрядное) слово занимает 8 байт. Если в ОП записать только двойные слова, то их количество будет в восемь раз меньше количества однобайтовых значений, записанных в ту же память. Следовательно, и адресное пространство для двойных слов оказывается меньшим в восемь раз по сравнению с байтовым. Но для сокращения разрядности адреса обращения к двойному слову, взятого в нашем примере, необходимо размещать его в памяти так, чтобы адрес начального байта двойного слова имел нули в трех младших разрядах. Применительно к 128-разрядному формату адрес начального байта должен иметь нули в семи младших разрядах. Такое размещение в памяти информации называют выравниванием по границе двойного слова. Обнуление при выравнивании младших адресов адреса выполняется аппаратно.

Выравниваются не только данные, но и другие объекты, например страницы виртуальной памяти. Поэтому в общем случае количество N нулевых младших разрядов адреса структуры или области памяти при выравнивании определяется по формуле

$$N = \log_2 S,$$

где S — размер структуры или области в байтах.

Выравнивание, несмотря на возможные издержки, существенно повышает эффективность использования регистров, предназначенных для хранения адресных слов, поскольку сокращение поля для адреса позволяет в освободившихся разрядах разместить другие поля. Кроме того, упрощается базирование адреса, под которым понимается получение адреса элемента путем сложения базового адреса с индексом этого элемента в структуре. Адрес требуемого элемента в выровненной структуре образуется заменой нулевых разрядов значением его индекса.

Широкая команда длиной от одного до восьми двойных слов выравнивается по границе двойного слова, поэтому ее адрес (Instruction Pointer) имеет нулевой код в трех младших разрядах.

Основные узлы

Структурная схема буфера команд представлена на рис. 3.2. Он включает:

- регистры адресов;
- кэш-память I$, включающую память команд (IDATA) и память тегов (ITAG);
- устройство формирования адресов (УФА);
- буфер таблицы страниц для команд (БТСК) (ITLB);
- устройство обращения к внешней памяти (УОВП).

Регистры адресов можно разделить на входные регистры, регистры текущих адресов и регистры адресов по памяти команд.

Входные регистры предназначены для приема адресов широких команд, программного кода основного канала (начальный адрес ШК) и трех дополнительных каналов подготовки переходов $РПП_j$ ($j = 1, 2, 3$). На регистр основного канала адрес может поступить при первоначальном пуске, прерываниях, выполнении операций переходов, а также при возврате из обработчика прерывания. На регистры дополнительных каналов записываются адреса ШК, с которых начинаются ветви возможного перехода. Назначение остальных регистров адресов приведено при дальнейшем рассмотрении.

Кэш I$ частично имеет ассоциативную организацию. Память тегов представляет собой ассоциативную часть, память команд — ответную.

Память команд (IDATA) предназначена для хранения строк программного (объектного) кода размером 256 байт, называемых строками команд. В целях оптимизации работы кэша I$ каждая строка разбита на 4 части по 64 байта. Соответственно, и выходной интерфейс буфера команд, по которому передается строка программного кода в устройство управления, разбивается на 4 части. Совокупность одноименных частей всех строк хранится в запоминающем устройстве, называемом банком памяти команд.

Банки могут работать независимо друг от друга: отрабатывать запросы от четырех потоков (ветвей) по считыванию и одного потока по записи (объектного кода, поступающего из кэша L2$ и оперативной памяти). Такая автономность банков позволяет оптимизировать пропускную способность физически однопортовой памяти команд.

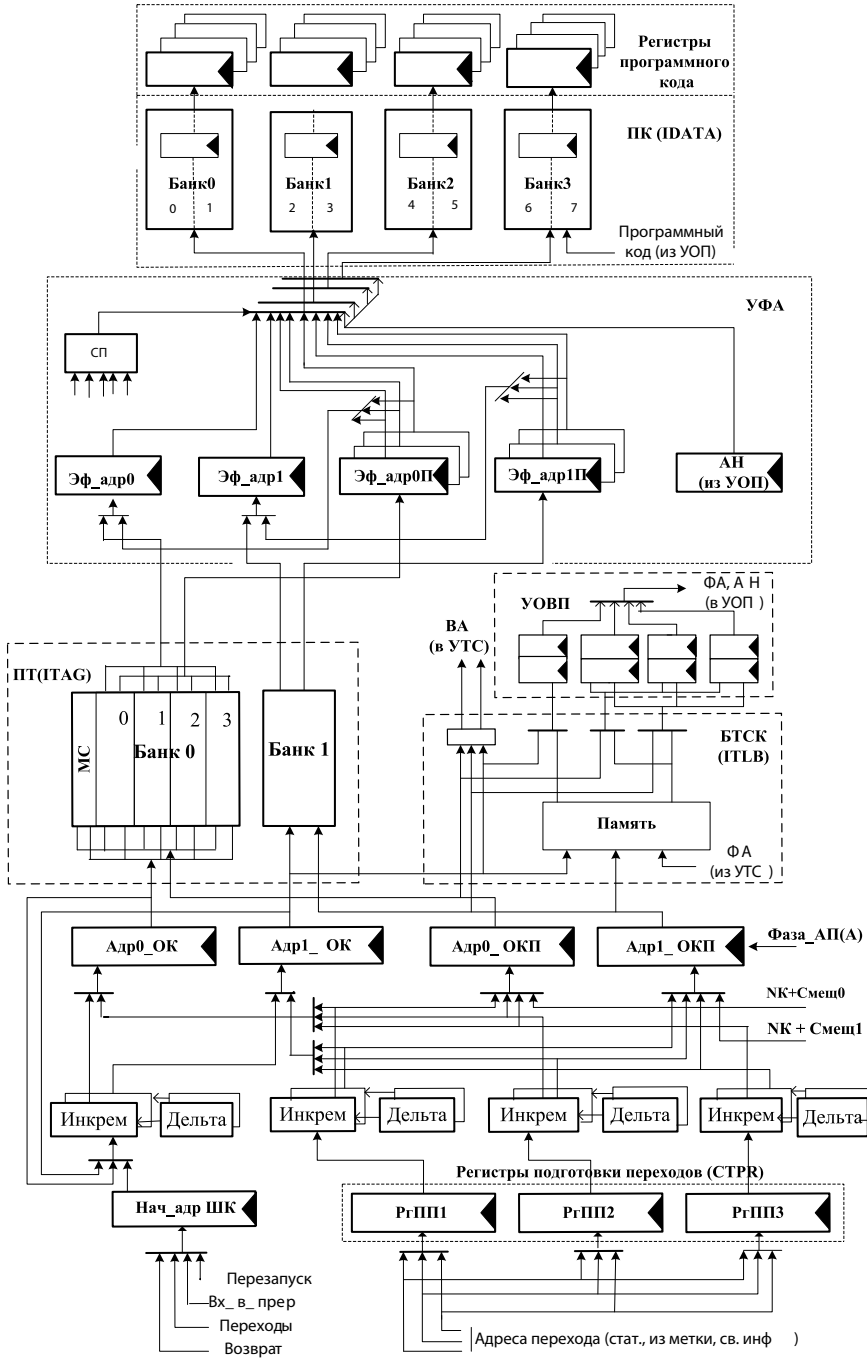


Рис. 3.2. Структурная схема буфера команд

За один такт из командной памяти по запросу УУ может быть считан фрагмент программного кода, выбранный из одной строки или двух смежных строк (при обслуживании одного конвейера) или из нескольких строк (при одновременном обслуживании нескольких конвейеров). Считанный из банка код записывается в один из четырех (по числу конвейеров устройства управления) регистров и далее выдается в УУ на регистры дешифрации.

Память тегов (ITAG) можно представить в виде таблицы, имеющей 64 строки и 4 столбца. Каждая строка содержит четыре ассоциативных признака (тега) и признак ее значимости. Таким образом, общее количество признаков равно числу строк программного кода в командной памяти, то есть каждому тегу соответствует строка объектного кода размером 256 байт.

Для полной ассоциативности необходимо иметь в памяти тегов 256 схем сравнения. Разрабатывать ЗУ с таким количеством схем сравнения нерационально, поэтому используется частично ассоциативная организация, предполагающая наличие лишь четырех (по числу столбцов) схем сравнения. На схемы сравнения поступают одновременно коды всех четырех тегов строки ПТ и соответствующая часть входного адреса, по которому происходит поиск требуемой строки объектного кода.

В качестве ассоциативного признака служат 34 старших разряда виртуального адреса (47:14) и номер контекста. В режиме физической адресации используется код 26 старших разрядов (39:14) адреса.

В целях повышения производительности буфера команд память тегов выполнена в виде двух банков по 32 строки в каждом, причем один из них хранит теги для четных строк объектного кода (ОК), другой — для нечетных. Банки памяти тегов работают одновременно, обеспечивая поиск в кэше команд двух смежных строк программного кода.

Память тегов имеет сравнительно небольшой объем, поэтому применительно к ней реализован механизм старения типа LRU.

Банк памяти тегов имеет два порта: один для обращений основного канала, второй — для одного из каналов подготовки переходов. Поскольку из каждого канала поступают два адреса, то в ПТ могут отрабатываться одновременно четыре запроса.

Устройство формирования адресов обращения к памяти команд включает регистры коротких адресов (Эф_адр), коммутаторы и схемы приоритета (СП). Кроме того, устройство имеет регистры адреса назначения (АН)

и программного кода, поступающего из устройства обмена с памятью. Короткие адреса называют также эффективными адресами. Этим термином обозначают адреса, полученные в результате какого-либо преобразования, например сложения базы и индекса. В данном случае короткий адрес получен на выходе памяти тегов вследствие подачи на его вход длинного адреса строки программного кода.

Для коротких адресов формируются признаки значимости, что позволяет заблокировать их отработку в памяти команд, а следовательно, запретить выдачу программного кода в устройство управления. Два регистра (Эф_адр0 и Эф_адр1) принимают короткие адреса (для четных и нечетных строк) основного канала, два комплекта по три регистра (по числу ветвей) принимают короткие адреса для каналов подготовки переходов. С целью повышения производительности в каждом такте запускаются все четыре банка памяти ПК.

Короткие адреса из устройства формирования выдаются в банки памяти команд через коммутаторы. При одновременном поступлении в банк обращений от нескольких запрашивающих устройств конфликты разрешаются схемами приоритета. Наивысшим приоритетом пользуется запрос по записи, поступающий от устройства обращения в память. Затем следуют запросы основного канала и каналов подготовки переходов (низший приоритет — у третьего канала подготовки).

Буфер команд может работать в режимах физической или виртуальной адресации. В режиме физической адресации строка программного кода считается найденной в кэше L1$ при совпадении кодов старших разрядов физического адреса и одного из четырех тегов значимой строки.

В режиме виртуальной адресации тег кроме упомянутого кода 34 старших разрядов виртуального адреса содержит бит глобального контекста, номер контекста текущей строки программного кода. Найденная строка объектного кода в памяти IDATA может быть считана только при наличии соответствующей строки соответствия в буфере таблицы страниц устройства БК. Связано это с защитой страниц от несанкционированного доступа, признаки и поля которой имеются в строке соответствия.

Буфер таблицы страниц для команд (Instruction Table Look-aside Buffer (ITLB)) предназначен для преобразования виртуального адреса в физический. Необходимость такого преобразования возникает в случае отсутствия требуемой строки программного кода в кэше I$ (промахе в памяти тегов).

Буфер таблицы страниц — полностью ассоциативное запоминающее устройство емкостью 64 строки. Ассоциативная часть содержит код старших (47–12) разрядов виртуального адреса, номер контекста задачи, формат страницы, признаки глобальной страницы и значимости. Ответная часть хранит физический адрес страницы (разряды 39–12 адреса), поле номера блока компиляции (Current compilation unit index (CUI) — индекса текущего модуля компиляции) и признаки (значимости, привилегированной страницы и др.). Буфер имеет два порта, один из которых предназначен для чтения по запросам основного канала и записи данных из таблицы страниц в случае подкачки новой строки соответствия, второй — для чтения по запросам каналов подготовки передачи управления.

Преобразование виртуального адреса в физический занимает два такта. В первом такте (стадия АП конвейера) происходит поиск нужной строки соответствия путем одновременного сравнения номера страницы из входного виртуального адреса и номера текущего контекста с аналогичными данными ассоциативной части.

Во втором такте (стадия ПК0) в случае успешного поиска на выходе буфера таблицы страниц появляются данные ответной части — физический адрес страницы и другие данные, указанные ранее. Физический адрес начала строки программного кода образуется дополнением физического адреса страницы кодом младших разрядов (11–3) исходного виртуального адреса. Этот код задает адрес внутри страницы. Полученный физический адрес начала программного кода выдается в устройство обращения к внешней памяти БК.

В случае отсутствия требуемой строки соответствия в буфере таблицы страниц (промаха) из буфера выдается виртуальный адрес в устройство таблицы страниц (УТС) (Table Look-aside Unit, TLU) для подкачки нужной строки. Если все строки буфера таблицы страниц устройства БК заняты, поступившая строка соответствия записывается вместо строки, вызванной в буфер ранее всех остальных. Таким образом, при замене используется дисциплина очереди.

Устройство обращения во внешнюю память предназначено для выдачи запроса в случае отсутствия требуемого программного кода в кэш-памяти I$. Запрос включает физический адрес начала строки программного кода и адрес назначения в памяти команд, по которому должен быть записан считанный объектный код. В режиме физической адресации адресом запроса является входной физический адрес. Устройство обращения в память осуществляет связь с кэшем L2$ и оперативной памятью. По одному запросу

может быть подкачана полная строка программного кода (256 байт) или часть строки.

Запросы в память могут инициироваться основным каналом и тремя каналами подготовки переходов. В конфликтных ситуациях наибольшим приоритетом обладают запросы основного канала, затем идут каналы подготовки переходов по мере возрастания их номера. Наименьшим приоритетом пользуется третий канал подготовки переходов. Возможна ситуация, когда разные источники, например основной канал и один из каналов подготовки передач управления, инициализируют запросы по одинаковым адресам. В этом случае значимость запроса с меньшим приоритетом будет заблокирована и повторного запроса за одними и теми же данными не будет.

Функционирование

Работа на линейном участке программы. На линейном участке программы отрабатываются запросы основного канала. После выполнения процедуры инициализации процессора, запускаемой при общем сбросе сигналом перезапуска (RESET), на входной регистр основного канала поступает номер ШК начала исполняемой программы. Затем адрес передается на регистры обращения к памяти тегов, причем на регистр Адр0_ОК принимается адрес четной строки объектного кода, а на регистр Адр1_ОК — адрес нечетной строки.

Адреса строк объектного кода поступают одновременно в банки тегов четных и нечетных строк с целью проверки наличия в них тегов, соответствующих адресам запроса. В случае успеха (hit) на выходе памяти тегов формируются два коротких (эффективных) адреса четной и нечетной строк объектного кода, хранящегося в банках ПК. В режиме виртуальной адресации перед считыванием программного кода из памяти команд необходимо санкционировать его использование. С этой целью адрес строки программного кода одновременно с поступлением на вход памяти тегов участвует в ассоциативном поиске в буфере таблицы строк соответствия. Убедиться в праве использования программного кода можно по признакам, имеющимся в ответной части буфера строк, то есть только при наличии в буфере требуемой строки соответствия. При положительном исходе проверки строка программного кода считывается из памяти ПК и направляется в соответствии с признаком конвейера, выданным устройством управления при запросе на подкачку строки команд. В рассматриваемом случае строка объектного кода поступит на нулевой (основной) конвейер УУ. Заметим, что признаки назначения формируются лишь при наличии признака значимости строки

программного кода и незанятости регистра дешифрации. В противном случае формирование признаков назначения блокируется и прием кода команд из БК приостанавливается.

На линейном участке программы начало строки объектного кода расположено в банке 0 ПК, поэтому для считывания полной (256-байтной) строки достаточно одного эффективного адреса, четного или нечетного. После отработки адреса запроса в память ПК выдается строб модификации адреса строки, хранящейся на регистре Адр0_ОК (Адр1_ОК). В результате с помощью узлов инкрементов образуются адреса следующих строк программного кода.

Если при просмотре памяти ITAG требуемые теги не обнаружены (ни одна из схем совпадения не сработала), отсутствующий тег записывается в колонку с номером, выданным механизмом старения (LRU). Формируется запрос в устройство обращения к памяти, в состав которого входят физический адрес отсутствующей строки команд и адрес назначения по памяти команд. Для получения физического адреса выполняется ассоциативный поиск строки соответствия в буфере таблицы страниц.

При считывании объектного кода из кэша L2$ или оперативной памяти необходимо задать объем подготавливаемого кода, выражаемого в количестве подкачиваемых строк. Такой объем называют также глубиной предварительной подкачки, и соответствующий ему код хранится в поле «ГПП» (Instruction Preliminary Depth, IPD — глубина предварительной подкачки команд) регистров конфигурации. Для основного канала глубина подкачки определяется полем регистра устройства управления памятью MMU_CR.IPD. Возможные его состояния задают два режима:

- 0 — подготовка кода команд только из кэша I$ БК (не производится обращение ни в L2$, ни во внешнюю память);
- 1 — подкачка двух строк программного кода (проверка в БК двух строк кода и в случае промаха обращение за командами в L2$ или память).

До момента поступления из памяти отсутствовавшей строки команд блокируется формирование значимостей коротких адресов на регистрах основного канала. Этим приостанавливается выдача программного кода из памяти команд по запросам основного канала на время блокировки.

В виртуальном режиме работы БК подобная блокировка может возникнуть и при наличии тега, но отсутствии строки соответствия в буфере таблицы

страниц. Она будет снята при поступлении из устройства таблицы страниц запрошенной строки соответствия.

Работа при подготовке перехода. Команды передачи управления предназначены для изменения порядка следования команд. Более подробно классификация команд переходов и принципы их выполнения рассматриваются позже. Сейчас достаточно заметить, что суть подготовки перехода заключается в заблаговременной передаче в устройство управления строки программного кода ветви, на которую возможен переход. Данные, необходимые для перехода, записываются в системные регистры подготовки переходов по специальным операциям подготовки перехода.

В регистрах подготовки передач управления РПП содержатся поля кода операции, виртуального адреса ШК (целевые адреса), которым может быть передано управление, поля тегов слов, используемых при переходе, и кодов глубины предварительной подкачки в случае промаха в памяти команд. Адреса переходов в зависимости от операций подготовки загружаются в РПП из устройства управления, АЛК и устройства подпрограмм. На схеме буфера команд (рис. 3.2) показаны только поля адресов регистров РПП. Поля операций и тегов записываются одновременно с адресами.

Важнейшей архитектурной особенностью операций подготовки переходов является их выполнение в спекулятивном режиме на фоне вычислений по основной программе. Все возникшие исключения не вызывают прерывания, а только отмечаются в регистрах РПП и диагностических словах. Разбор ситуаций откладывается на момент, следующий за фактическим переходом на данную ветвь программы.

Естественным продолжением операций подготовки переходов является работа буфера команд по считыванию ветви и загрузка ее на соответствующий конвейер устройства управления. С этой целью загруженный на РПП целевой адрес передается на регистры Адр0_ОК и Адр1_ОК четной и нечетной строк программного кода соответственно. В критических по времени случаях на эти регистры может поступить адрес перехода (по шине смещения NK + Смещ0) и адрес следующей строки (по шине NK + Смещ1) из шин байпаса. Адреса строк программного кода одновременно выдаются в устройства память тегов ITAG и буфер таблицы страниц ITLB. Поскольку для трех регистров РПП имеется только один комплект регистров адресов обращения в ITAG и ITLB, схема приоритета в случае конфликта разрешает передачу адреса наиболее приоритетного в текущий момент регистра подготовки. Наибольшим приоритетом пользуется регистр первого канала РПП1, наименьшим — РПП3. Каждый регистр РПП может обращаться

с запросами раз в три такта, что позволяет обратиться в память каналу с меньшим приоритетом. Для каждого канала подготовки переходов, как и для основного, имеются инкрементеры, модифицирующие адреса строк объектного кода после завершения обработки текущего запроса.

Дальнейшее функционирование буфера команд аналогично рассмотренному ранее процессу. Следует лишь отметить, что при подготовке переходов может не оказаться требуемой строки соответствия в буфере таблицы страниц. В этом случае, в отличие от запросов основного канала, обращение в устройство таблицы страниц с целью подкачки отсутствующей строки соответствия задерживается вплоть до перехода на данную ветвь. Этим исключается замена в буфере таблицы строки, которая может потребоваться основному каналу.

При отработке команд подготовки перехода возможно обращение в кэш L2 и оперативную память за программным кодом. Объем подготавливаемого кода, или глубина предварительной подкачки, определяется кодом поля IPD регистра подготовки перехода. В свою очередь, это поле устанавливается одноименным полем слога SS. Реализовано следующее соответствие между кодами поля и действиями:

- 00 — программный код ветви считывается только из БК (не производится обращение ни в L2$, ни в оперативную память);
- 01 — считывание первой строки объектного кода (проверяется наличие двух строк в кэш IB$, и в случае промаха формируется запрос в L2$ или ОП на считывание от начала ШК до конца строки);
- 10, 11 — подготовка двух строк кода команд (выполняется проверка хранения двух строк в кэш команд, и в случае промаха организуется считывание из кэша L2$ или ОП от начала ШК до конца строки и следующей целой строки).

В случае отсутствия слога SS в поле IPD записывается код 10 (по умолчанию), что определяет подкачку двух строк программного кода.

Работа при подкачке асинхронной программы. Начало программы асинхронной подкачки и соответствующий код операции загружаются на регистр РПП2. Далее обычным образом выполняется проверка наличия команд в буфере команд.

В отличие от обычной операции подготовки перехода отсутствие в буфере таблицы страниц строки соответствия не приводит к блокировке запроса

в L2$ и ОП. Формируется обращение в устройство таблицы страниц, и после получения строки соответствия осуществляется подкачка недостающих кодов команд в кэш I& буфера команд. Программный код, считанный из памяти IDATA, поступает в устройство управления, откуда через регистр дешифрации второго канала передается в устройство обращения к массивам УОМ. Признак конца асинхронной программы, поступающий из устройства управления, останавливает подкачку программного кода.

Таким образом, буфер команд обеспечивает выдачу в устройство управления строк программного кода основной ветви и трех ветвей предполагаемого перехода. Причем в течение одного такта могут быть выданы 256 байт объектного кода одной ветви или фрагменты кода по 64 байта, принадлежащие различным ветвям.

3.2.4. Устройство управления

Назначение и структура

Из рассмотренного ранее следует, что основная функция буфера команд заключается в хранении и выдаче программного кода в УУ, которое осуществляет его обработку. Программный код выдается в виде строк, включающих последовательность упакованных широких команд. Под упакованной ШК понимается сформированная компилятором совокупность операций, которые могут выполняться параллельно на соответствующих устройствах процессора. Количество этих операций ограничивается максимальным размером ШК, который не превышает восьми двойных слов (8×8 байт).

Длина широких команд может быть различной, поэтому строки программного кода, включающие последовательность ШК, имеют в общем случае различную длину. При этом начало ШК может оказаться в конце строки, а ее продолжение — в следующей строке.

Из-за ограничений на длину в одну широкую команду может входить лишь часть подлежащих исполнению слогов. Для дальнейшего выполнения операций она преобразуется к стандартному формату — формату исполнительной ШК (будем называть ее также дешифрованной или распакованной ШК). Дешифрованная ШК (ДШК) занимает 136 байтов и состоит из полей различной ширины (64, 32, 16 разрядов). Каждое поле закреплено за определенным слогом, причем слоги в этой команде располагаются в неизменном порядке (приложение 4).

Для преобразования упакованной ШК в исполнительную необходимо вначале выделить ее из строки программного кода. Распределение слогов по полям исполнительной широкой команды требует наличия дополнительных данных, касающихся структуры упакованной ШК и назначения ее слогов. Все они содержатся в заголовке упакованной ШК и специальном слоге.

Эти предварительные замечания позволяют перейти к рассмотрению устройства управления.

Устройство управления (Control Unit, CU) выполняет следующие функции:

- выборку упакованной широкой команды из строки программного кода, поступившей на входные регистры;
- распаковку широкой команды;
- вычисление адресов операндов;
- проверку условий для команд передачи управления и переключение процессора на соответствующую ветвь программы;
- передачу программы предварительной подкачки элементов массива в устройство обращения к массивам УОМ.

Как показано на рис. 3.3, в устройстве управления имеются четыре конвейера — основной и три дополнительных, предназначенных для подготовки передачи управления.

Основной конвейер действует на трех стадиях:

- на стадии Распак (S) на входном регистре осуществляется выделение широкой команды и ее распаковка;
- на стадии Дш (D) обеспечивается декодирование операций;
- на стадии БА (B), предназначенной для базирования (Basing) и передачи управления (Branch), выполняется вычисление адресов операндов, а также переключение основного конвейера на ветвь, для которой выполнено условие перехода.

Стадии работы дополнительных конвейеров аналогичны. Функции этих стадий реализуются следующими устройствами.

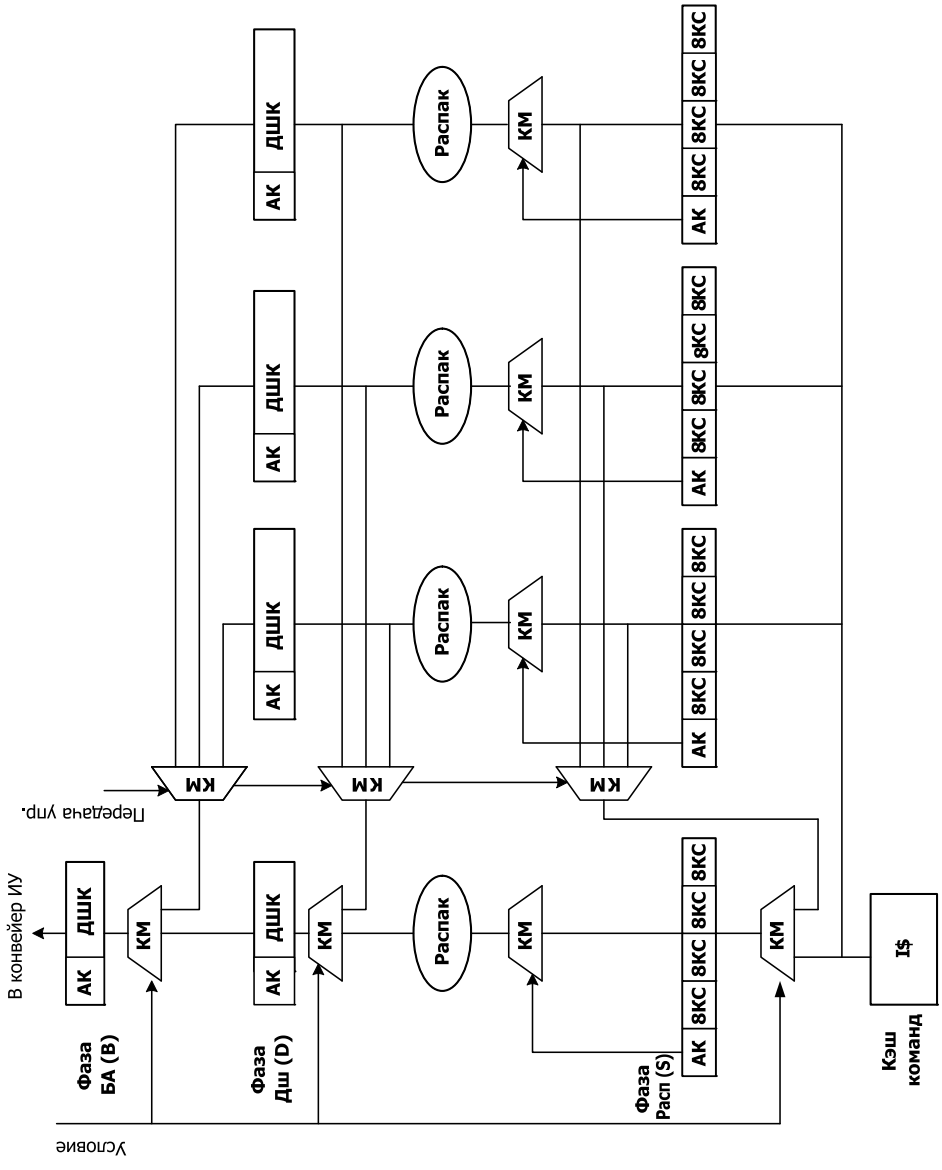


Рис. 3.3. Блок-схема устройства управления

Входные регистры предназначены для хранения 256-байтных фрагментов программного кода, считанного из кэша I$ буфера команд. На входной регистр принимаются четыре фрагмента строки объектного кода по 64 байта (8 командных слов) и код адреса широкой команды, каждое из этих полей сопровождается битом значимости.

Узел распаковки и коммутатор (КМ) предназначены для выделения широкой команды из строки программного кода. После выделения очередной команды к предыдущему значению счетчика команд прибавляется ее длина, которая содержится в заголовке команды, в результате чего образуется адрес следующей ШК в строке объектного кода. Распаковка осуществляется с помощью коммутатора, управляемого кодом счетчика команд. Если на входном регистре достаточно объектного кода для очередного цикла распаковки, то выделенная команда сопровождается битом значимости. С выхода коммутатора ШК поступает на схему рассеивания.

Схема рассеивания распределяет слоги упакованной ШК по полям исполнительной широкой команды (приложение 4). Определен следующий порядок размещения слогов в широкой команде: ALS0, ALS1, ALS2, ALS3, ALS4, ALS5, CS0, CS1, SS, ALES0, ALES1, ALES3, ALES4, AAS0, AAS1, AAS2, AAS3, AAS4, AAS5, LTS3, LTS2, LTS1, LTS0, PLS2, PLS1, PLS0, CDS2, CDS1, CDS0. Слоги могут иметь двухсловный, однословный и полусловный форматы. В дешифрованной широкой команде может быть 6 двухсловных, 17 однословных и 10 полусловных слогов.

Регистры ДШК фазы Дш (D) предназначены для хранения распакованной широкой команды. При поступлении на регистр ДШК стадии БА из основного конвейера команды условного перехода выполняются операции подготовки и передачи управления. В зависимости от значения предикатов на этот регистр коммутируется ШК с одного из дополнительных конвейеров, дешифрующего ветвь, на которую совершен переход. С регистра второго конвейера данного уровня производится передача программы предварительной подкачки элементов массива в устройство УОМ.

На стадии БА работает только один регистр основной ветви. В ней выполняются операции подготовки и передачи управления. Подключение подготовленной ветви к основному конвейеру по команде условного перехода осуществляется на этом уровне с помощью коммутаторов.

Дешифрация широкой команды

Дешифрация широкой команды выполняется за два такта (приложение 4). В первом такте производится частичная дешифрация, в результате которой заполняются поля слогов дешифрованной команды. Во втором такте на основе анализа информации арифметических слогов выполняется дополнительная дешифрация, целью которой является формирование литеральных констант из оставшихся слогов LTS (LiTeral Syllable — слог литерала). Они будут использоваться в качестве операндов в АЛК. Общая схема дешифрации широкой команды представлена на рис. 3.4.

Рис. 3.4. Схема дешифрации широкой команды

Функционирование

Формирование действительных адресов операндов. В качестве операндов для исполнительных устройств используются данные, хранящиеся в регистровом файле (РгФ), или литеральные константы. На стадиях конвейера Дш (D) и БА (B) определяются их адреса.

Операция (простая команда) распакованной широкой команды в общем случае содержит в своем составе 8-разрядные адреса источника (source, src), с помощью которых определяются действительные (эффективные) адреса операндов. Источник операндов для простых команд — РгФ или

совокупность литералов. Регистровый файл содержит несколько областей, которые подробно рассматриваются далее: стековую область (регистры текущего окна), вращающуюся область (ВО) стековых регистров, область глобальных регистров и вращающуюся область глобальных регистров. Обращение в регистровый файл ведется по индексу с точностью до одного регистра. В качестве индекса выбирается код младших разрядов адреса. Количество младших разрядов определяется максимальным размером адресуемой области РгФ. Код старших разрядов адреса задает область регистрового файла, в которую выполняется обращение. Кодировка областей приведена в табл. 3.4.

Таблица 3.4. Кодировка областей РгФ

Номера старших разрядов	**Код**	**Область регистрового файла**	**Размер области, регистр**	**Размер поля индекса, разряд**
7	0	ВО стековых регистров	128	7
7–6	10	Текущее окно	64	6
7–5	111	Область глобальных регистров	32	5
7–3	11111	ВО глобальных регистров	8	3
7–5	110	—	—	—

Действительный адрес по РгФ определяется в результате индексирования базы области. Номер первого регистра области задан в дескрипторе, который описывает эту область. В частности, эффективный адрес регистра текущего окна определяется путем сложения кода базы, хранящейся в составе дескриптора текущего окна на регистре WD (window descriptor), и кода индекса (шести младших разрядов адреса из простой команды). Если действительный адрес регистра по регистровому файлу выходит за пределы текущего окна, вырабатывается соответствующий сигнал прерывания.

Аналогичным образом вычисляются действительные адреса по РгФ и для записи результатов операции. В качестве исходных данных берутся адреса назначения (dst — адрес назначения (destination) простой команды).

Если код старших разрядов адреса 110, то вычисление адреса по регистровому файлу не производится, поскольку в поле адреса находится короткий

литерал. Код 1101 является ссылкой на литерал длиной 16, 32 или 64 разряда.

Управление конвейером. Простые операции широкой команды выполняются всеми исполнительными устройствами микропроцессора одновременно, то есть конвейер после стадий работы УУ разделяется на параллельные ветви. При этом каждая ветвь конвейера содержит свою часть ШК. Устройство управления контролирует выполнение операций на первых трех станциях исполняющей части, чтобы определить возможность корректного завершения отработки текущей широкой команды. Контроль заключается в проверке наличия готовых операндов для всех операций текущей широкой команды, определении возможности выдачи на выполнение операций с темпом один раз и более за два такта, доступности регистров состояния микропроцессора, модифицируемых предшествующими широкими командами, и соблюдения других условий. Нарушение нормального выполнения ШК является конфликтной ситуацией. Следует отметить, что конфликтная ситуация носит временный характер и приводит лишь к задержке вычислительного процесса. Но одновременный останов всех параллельных ветвей такого широкого конвейера, какой применен в рассматриваемом процессоре, физически невозможен (хотя бы по причине разного времени распространения сигналов), поэтому принято другое решение — ввести так называемую вертушку, представляющую собой буфер с четырьмя станциями запоминания. По мере продвижения широкой команды по исполнительным стадиям И0 (Е0), И1 (Е1) и И2 (Е2) происходит одновременное перемещение содержимого вертушки с одной станции на другую. При возникновении конфликтной ситуации начальная часть конвейера блокируется (останавливается), а в остальную, исполнительную часть конвейера последовательно выдаются широкие команды из вертушки с целью повторной отработки. В общем случае широкие команды повторяются многократно вплоть до разрешения конфликтной ситуации.

Таким образом, любая конфликтная ситуация приводит к останову конвейера, причем часть его, находящаяся в буфере команд и УУ, замирает, а фазы остальной части циклически повторяются, пока не будут ликвидированы причины задержки. Организуя повторную отработку команд, УУ фактически восстанавливает расписание выполнения программы.

Работа на линейном участке программы. На линейном участке программы, который размещается на основном конвейере, работа УУ по распаковке широких команд и их дешифрации обеспечивается своевременной подкачкой программного кода из буфера команд. На выходных станциях

осуществляется вычисление адресов операндов и их считывание с регистрового файла. Простые операции широкой команды выдаются одновременно на исполнение в арифметико-логические каналы.

Для согласования взаимодействия буфера команд, способного выдать за один такт строку программного кода длиной 256 байт, и устройства управления, обладающего гораздо меньшей производительностью, на входе УУ введен буфер на четыре позиции. Каждая позиция буфера хранит отрезок программного кода, равный емкости банка этого буфера (64 байта), и имеет бит значимости, который доступен буферу команд. В зависимости от его значения разрешается или запрещается выдача программного кода в позицию входного регистра УУ, соответствующую определенному банку.

Работа при выполнении ветвлений. Выполнение команд переходов требует определенных временных затрат, которые негативно отражаются на общей производительности процессора. Для их сокращения используется заблаговременная подготовка программного кода, которая может задействовать до трех направлений предполагаемого перехода. Предусмотрены два варианта передачи управления:

- немедленная (instant);
- подготовленная (prepared).

Немедленная передача осуществляется одной операцией СТ (Control Transfer), содержащей всю необходимую информацию. Для подготовленной передачи управления требуются операции двух типов: подготовки передачи управления (Control Transfer Preparation, CTP) и фактической передачи управления (СТ).

Операции типа CTP предназначены для выполнения всей подготовительной работы на фоне выполнения команд прямой ветви. Каждая из них содержит код операции, адрес целевой команды, которой передается управление, тип адресного слова, глубину подкачки, определяющую количество строк программного кода, и подкачиваемых в буфер команд.

Исполнение операции подготовки перехода заключается в следующем. По адресу ШК, записанному в заданный входной регистр БК (регистр РПП), производится поиск требуемой строки программного кода. При ее наличии в кэше команд строка выдается на входной регистр УУ дополнительного конвейера, номер которого соответствует номеру регистра подготовки CTPR. Выполняются распаковка первой широкой команды строки и ее дешифрация.

В случае отсутствия в буфере команд строки программного кода, содержащей целевую широкую команду, происходит подкачки строки из кэша L2$ или ОП. При этом может быть задана глубина подкачки. По умолчанию подкачиваются отсутствующая в БК и следующая за ней строки.

Переход по операции СТ может быть условным или безусловным. Условие передачи управления содержит 4-разрядное поле типа условия и адрес предиката по предикатному файлу. Такое широкое поле типа условия позволяет сформировать команду перехода применительно к большому числу вариантов. Например, 0 соответствует блокировке переходов (never — никогда), 1 — безусловному переходу (always — всегда), 2 — переходу по предикату из предикатного файла, 3 — по инверсному значению предиката и т. д.

При совпадении условия перехода, имеющегося в операции, с выработанным условием производится переключение процессора на подготовленную ветвь. Переход заключается в переписывании содержимого задействованного дополнительного конвейера в основной конвейер.

Работа по сигналам прерывания. В результате прерывания вызывается процедура операционной системы (ОС). Контекст пользователя включает интерфейс ОС, который организован как массив, содержащий 32 элемента (сегментов кода фиксированной длины). Массив размещается в начале интерфейсного модуля компиляции ОС и описывается регистром дескриптора модуля компиляции операционной системы (Operation System Compilation Unit Descriptor, OSCUD).

Каждый элемент массива служит в качестве входа операционной системы. ОС может ограничить интерфейс пользователя путем маскирования того или иного входа. Для конкретного пользователя формируется битовая маска, определяющая разрешенные (запрещенные) входы.

Различаются аппаратурные и программные прерывания. Аппаратурные прерывания возникают вследствие возникновения исключительных ситуаций, например деления на нуль, выход за границы окна стека и т. п. Программные или внешние прерывания вырабатываются по сигналам от устройств, входящих в вычислительную систему, или инициируются из самого кода программы.

По отношению к основному командному потоку аппаратурные и программные прерывания делятся на синхронные и асинхронные. Синхронные прерывания возникают при отработке операций основного командного

потока, асинхронные прерывания имеют внешний характер, как, например, прерывание по таймеру.

Устройство управления вырабатывает около десяти синхронных сигналов прерываний, формируемых при выполнении запрещенных операций с плавающей точкой, попытках в непривилегированном режиме выполнить привилегированные действия, неправильной работе с текущим окном регистрового файла и в других ситуациях. Все сигналы вырабатываются до уровня И1 (Е1) конвейера.

При возникновении сигнала прерывания в общем случае производится сохранение состояния процессора, переход к процедуре обработки прерывания (обработчику), затем восстановление состояния и возврат к выполнению основной программы в том же режиме, в каком находился процессор в момент прерывания. Эта общая схема требует детализации в части продолжения выполнения основной программы после обработчика, которая определяется типом прерывания.

Если сигнал прерывания является следствием исключительной ситуации, возникшей при отработке команды, то необходимо повторить ее после возврата. В случае программного прерывания основная программа продолжается с невыполненной команды. Повторение команды, вызвавшей нештатную ситуацию, возможно только в случае, когда не возникло отрицательных последствий и текущее состояние процессора то же, что при нахождении команды в стадии И2 (Е2). Но, из-за того что наряду с простой командой в вертушке запоминается и состояние регистров процессора, которые могут быть изменены при продвижении команды по конвейеру, вертушка обеспечивает повторение отработки команды и при возникновении многих исключительных ситуаций. В ряде случаев при отработке прерываний требуются поддержка компилятора и вспомогательные аппаратные средства.

По сигналу прерывания, выработанному в результате незамаскированной исключительной ситуации, в блоке управления фиксируется начало входа в прерывание. В момент перехода на фазу И2 широкой команды, вызвавшей нештатную ситуацию, происходит отмена всех последующих команд, находящихся в конвейере, — очистка конвейера, flush. Одновременно по данным, хранящимся в вертушке, восстанавливается состояние процессора, которое было в момент нахождения рассматриваемой широкой команды на этой фазе. Затем устройство управления конвейером формирует временную диаграмму прерывания, результатом выполнения которой является переход к системной процедуре обработки возникшего исключения.

3.2.5. Регистровый файл

Регистровый файл (Register File, RF) предназначен для хранения операндов и результатов выполненных операций, которые в общем случае используются в последующих командах. По уровню иерархии запоминающих устройств РгФ является сверхоперативным запоминающим устройством (СОЗУ) процессора, соответственно, в системе команд предусмотрены операции, реализующие его связь с оперативной памятью вычислительной системы, которые выполняются в арифметико-логических каналах.

Считывание операндов осуществляется на стадии СчО (R) конвейера. Возможность произвести считывание по нескольким адресам реализована за счет разработки многопортового запоминающего устройства. Порт — это узел, обеспечивающий доступ абонента к требуемому регистру файла, который содержит управляющее оборудование и усилители чтения/записи. Все порты жестко закреплены за устройствами: один из них назначается для считывания операнда по первому адресу, к примеру нулевого АЛК, по другому порту доставляется второй операнд для этого же канала и т. д. В широкой команде в общем случае может быть до 20 адресов операндов, поэтому требуется такое же количество портов чтения. Очевидно, что увеличение количества портов приводит к возрастанию объема оборудования, причем появляется необходимость в дополнительных устройствах, исключающих конфликты при отработке в разных портах одинаковых адресов. Напротив, снижение числа портов позволяет уменьшить площадь кристалла и увеличить тактовую частоту работы регистрового файла. По этой причине РгФ содержит два одинаковых блока RF-A и RF-B для кластеров A и B соответственно. Для поддержания когерентности каждый блок доступен АЛК другого кластера по записи, он имеет 10 портов чтения и 10 портов записи.

Порты чтения обеспечивают считывание операндов для АЛК, причем каждому каналу выделено три порта, что обеспечивает возможность доставки данных для комбинированных операций, требующих трех операндов. Один из портов чтения предназначен для считывания из РгФ в АЛК2 (АЛК5) операнда для команды записи в память.

Через шесть портов записи в РгФ направляются результаты операций АЛК и данные из кэша первого уровня L1$. Четыре порта выделены для записи в РгФ данных, поступающих по шине кэша L2$, и элементов массива из буфера предварительной подкачки. Запись данных в целях их когерентности осуществляется одновременно в оба блока РгФ (таким образом, оба блока РгФ содержат одинаковые данные).

Следует отметить, что уменьшение числа портов в блоке РгФ с 20 до 10 позволило увеличить вдвое тактовую частоту его работы. В результате появилась возможность совместить порты записи и чтения: полтакта процессора занимает режим считывания из РгФ, в другой половине такта выполняется запись.

Память блока РгФ включает 256 регистров по 84 разряда каждый, имеющих три поля. Первые два поля предназначены для хранения 32-разрядных слов с 2-разрядными тегами, двойного слова Ф64, мантиссы вещественного числа, а третье поле отводится для хранения 16-разрядного значения порядка.

В управление регистровым файлом введено три механизма, обеспечивающих корректное и опережающее использование результатов операций в качестве операндов для последующих операций. Все механизмы задействуют адреса, по которым записываются результаты выполненных операций. Эти адреса сохраняются в АЛК и сопровождают операции по мере продвижения их по конвейеру вплоть до записи результата в РгФ.

Первый механизм (Scoreboarding — схема анализа блокировок выдачи ШК из УУ) блокирует очередную широкую команду, если она содержит обращения за еще не готовыми операндами, которые являются результатами предшествующих операций.

Второй механизм обеспечивает управление шинами байпаса, ускоряющими доставку операндов. Его работа основана на сравнении адресов операндов из ШК с адресами результатов выполненных операций, доступных по шинам байпаса.

Третий механизм предназначен для разрешения так называемой проблемы зависимости по выходу (output dependence). Она возникает, когда две последовательные операции имеют один и тот же адрес результата, но разное время выполнения и первая операция формирует свой результат уже после завершения выполнения второй операции.

Принцип использования. Окно, совокупность смежных регистров РгФ, выделяется для каждой запускаемой процедуры. База окна и его размер задаются компилятором. Окно описывается дескриптором, который хранится на регистре WD (current Window Descriptor).

Все регистры РгФ разбиты на две группы (рис. 3.5). Первая группа регистров (0–223) используется для окон процедур и составляет область стековых регистров, вторая группа (224–255) представляет собой область

глобальных регистров. Выделение окон осуществляется по кольцевому принципу. После достижения адреса 223 формируется очередной адрес 0. Вновь выделенное окно может включать несколько последних регистров предыдущего окна, они рассматриваются как выходные регистры окна и служат для передачи фактических параметров запускаемой процедуре.

Рис. 3.5. Логическая структура регистрового файла

В каждом окне может быть организована «вращающаяся область», которая предназначена для циклической обработки элементов массива, записываемых из буфера предварительной подкачки в РгФ. При ее организации задаются база относительно окна, текущая база и размер. Вращающаяся область представляет собой последовательность нескольких смежных стековых регистров, обращение к которым происходит по кольцевому принципу: после записи в последний регистр области очередной элемент массива будет записан в первый регистр.

Вращающаяся область может быть организована и в глобальных регистрах. Под нее отводятся последние восемь регистров.

Основной особенностью регистрового файла является возможность обслуживания в течение такта всех его абонентов. Его объем позволяет обеспечить необходимым количеством регистров несколько процедур.

3.2.6. Устройство логических предикатов

Логический предикат (ЛП) — результат операции сравнения двух чисел (целых или вещественных), выполняемой в арифметико-логических каналах. Он представляется двухразрядным составным значением, содержащим тег (0 — обычный предикат, 1 — диагностический предикат) и булево значение (0 — false, 1 — true).

Устройство логических предикатов (УЛП) (Predicate Logic Unit (PLU)), разработанное с целью разгрузки арифметических устройств, выполняет различные операции над малоформатными значениями предикатов и выдает результаты в арифметико-логический канал для управления его действиями. В число операций АЛК входят считывание из предикатного файла, вычисление вторичных предикатов по двум первичным, задание предикатного (условного) выполнения операций в арифметико-логическом канале и др. Кроме этого, УЛП выполняет запись предикатов в предикатный файл.

В состав УЛП входят:

- блок вычисления логических предикатов (ВЛП) (PLU);
- блок распределения логических предикатов (РЛП) (Rout Logic Predicate Unit, RLPU);
- блок предикатного файла (ПФ) (Predicat File, PF) с байпасами.

Блок вычисления логических предикатов предназначен для получения вторичного предиката — результата вычисления булева выражения. Оно выполняется за три этапа.

1. Считывание первичных операндов из предикатного файла отдельными операциями.
2. Логические операции над первичными операндами.
3. Запись результата в ПФ.

Блок распределения логических предикатов задает предикатный режим выполнения операции в арифметико-логическом канале (операция разрешается или запрещается) и передает ему предикат для управления.

Предикатный файл и байпасы. Предикатный файл — это 64-разрядный регистр, предназначенный для хранения предикатов. Четный разряд $j \times 2$ хранит значения j-го предиката, нечетный разряд $j \times 2 + 1$ отведен для тега j-го предиката. Таким образом, в предикатном файле может храниться до 32 предикатов. Обращение к ПФ за предикатом производится по номеру предиката, соответствующему номеру разряда регистра. Поскольку предикат имеет два разряда, ПФ рассматривают как массив двух разрядных регистров.

Для циклических участков программ организуется вращающаяся область. Принцип работы и параметры вращающейся области не отличаются от рассмотренных ранее для регистрового файла.

С целью ускоренной передачи логических предикатов от их формирователей к потребителям используются байпасы, через которые ЛП пересылаются вместе с их номерами.

3.2.7. Устройство подпрограмм

Назначение и структура

Устройство подпрограмм (УПП) (SubRoutine Unit, SRU) предназначено для выполнения операций, связанных с процедурными переходами и возвратами к прерванным процедурам.

Каждая процедура имеет свой контекст и выполняется на своем стеке. Оперативная (наиболее часто используемая) часть стека располагается в выделенном окне регистрового файла, а продолжение стека находится в общей памяти. Отсюда следует, что процедурный переход связан в общем

случае с выполнением значительного числа операций по организации смены контекста, стека для новой процедуры, запоминания режима, в котором она работала до перехода к новой процедуре, и восстановления состояния процессора при возврате к прерванной процедуре.

Основу устройства подпрограмм составляют программно доступные регистры, часть из которых выполнены в виде отдельных СОЗУ. Регистры УПП доступны на считывание в арифметико-логическом канале АЛК0 через шины длинного литерала, контекстная информация может быть считана в 0, 2, 3 и 5-й каналы.

Из структуры УПП целесообразно рассмотреть только назначение регистров, поскольку функции коммутаторов и сумматоров, формирующих базовые адреса из дескрипторов и индексов, не требуют пояснений. Основные регистры УПП можно разделить на следующие группы:

- стек пользователя;
- модуль компиляции;
- связующая информация;
- кэш таблицы.

Группа регистров стека пользователя образована регистрами базы стека пользователя USBR (User Stack Base Register) и дескриптора стека пользователя USD (User Stack Descriptor).

Дескриптор базы стека содержит виртуальный адрес дна текущего стека пользователя с нумерацией разрядов в терминах байтовой адресации.

Дескриптор стека пользователя описывает свободное пространство памяти, предназначенное для локальных данных пользователя, которое содержит базовый виртуальный адрес и размер стека (для незащищенного стека). В случае использования защищенного стека, признак которого указывается в дескрипторе, содержимое регистров стека пользователя несколько отличается от рассмотренного.

Группу регистров модуля компиляции (МК) составляют регистры его спецификаторов (описателей).

Спецификаторами модуля компиляции являются дескрипторы:

- текущего модуля компиляции CUD (Current Compilation Unit Descriptor);

- глобальных данных GD (Current compilation unit globals descriptor);
- типов TSD (Current compilation unit types descriptor).

Дескриптор модуля компиляции описывает память, содержащую программный код текущего МК, и включает базовый виртуальный адрес модуля, его размер и флаг системной проверки межмодульной защиты.

Дескриптор глобальных данных GD описывает память глобальных переменных и содержит данные, аналогичные содержимому дескриптора CUD.

Дескриптор типов текущего модуля описывает диапазон абсолютных индексов типов (в массиве типов), принадлежащих текущему МК.

Спецификаторы МК хранятся в одноименных регистрах УПП (CUD, GD, TSD) и упаковываются в четыре двойных слова. Такие упаковки собираются в таблицу модулей компиляции CUT (Compilation Units Table), образуя ее строки.

Таблица модулей компиляции, состоящая из 32-байтовых строк, описывается своим дескриптором CUTD, который хранится на соответствующем регистре устройства подпрограмм. Строка таблицы МК содержит все данные о модуле компиляции, номером которого является индекс его строки, определяющий контекст модуля компиляции. В момент исполнения он хранится на регистре индекса текущего модуля компиляции CUIR (Current Compilation Unit Index Register), а замена содержимого CUIR происходит при процедурных переходах со сменой контекста.

Большинство регистров рассмотренных групп снабжено теневыми регистрами. Они применяются в том случае, если основные регистры модифицируются по мере прохождения команд в стадиях Сч0—И2. В этом случае первоначальные значения запоминаются в теневых регистрах и подобно регистрам вертушки, передаются с одного теневого регистра в другой, сопровождая свои команды. В случае необходимости повтора команды происходит восстановление основных регистров данными, взятыми из теневых регистров.

Группа регистров связующей информации образована регистрами, с помощью которых организуется стек, содержащий историю процедурных переходов для текущего процесса. Частично он состоит из регистров процессора CR (chain registers), в состав которых входят CWD (current Chain Window Descriptor — дескриптор текущего окна регистров связующей информации), PCSHTP (Procedure Stack Hardware Top Pointer — указатель аппаратурной вершины стека процедур), PCSP (Procedure Chain Stack

Pointer — указатель стека связующей информации). Остальная часть стека находится в оперативной памяти.

Регистры CR выполнены в виде памяти емкостью 128 двойных слов, состоящей физически из четырех банков двухпортовой памяти 32×64. Они модифицируются операциями записи в регистр при вызове новой процедуры и входе в аппаратурный обработчик прерывания.

Кроме рассмотренных регистров в состав УПП для обеспечения совместимости с машинами фирмы Intel введены сегментные регистры (CS, DS, ES, FS, GS, SS), описывающие адресное пространство, используемое данной фирмой. Они не связаны с остальным оборудованием УПП.

Типы стеков

В архитектуре «Эльбрус» различают следующие разновидности стеков:

- стек процедур;
- стек пользователя;
- стек связующей информации.

Стек процедур. Стек процедур представляет собой непрерывную область, предназначенную для временного хранения тех фактических параметров и локальных данных еще не завершенных процедур, которые размещены компилятором в операционных регистрах (регистровых окнах). К этой категории данных относятся также результаты выполненных операций, записываемых в регистровый файл.

Для каждой запускаемой процедуры в регистровом файле отводится окно (фрейм) требуемого размера, в пределах которого разрешена ее работа. Окно новой процедуры и предыдущее окно могут иметь общую область, в которой содержатся параметры, передаваемые запускающей процедурой, и возвращаемые значения — результаты ее работы. Завершение процедуры приводит к ликвидации ее окна.

По мере запуска новых процедур ресурс свободных регистров РгФ может быть недостаточным для очередного окна. В этом случае нижняя часть стека автоматически откачивается в память. И наоборот, после завершения запущенной процедуры и возврата к прерванной процедуре необходимые данные автоматически подкачиваются из памяти в РгФ. В силу этого стек процедур состоит из двух частей, одна из которых располагается в регистровом файле, а для другой отводится участок памяти на случай переполнения РгФ.

Стек пользователя. Стек пользователя предназначен для данных, не относящихся к рассмотренной ранее категории, например различного рода динамических массивов или объектов программ на объектно-ориентированных языках.

В принципе вопросы выделения памяти под различные структуры данных решаются с помощью стандартных процедур управления динамической памятью типа `new/delete`. Но обращение к ним связано с дополнительными затратами времени при вызове процедур операционной системы. Поэтому в микропроцессоре реализован более эффективный способ управления динамической памятью, основанный на организации стека для этих целей. В систему команд введена специальная операция, формирующая дескриптор на участок памяти, выделенный из стека пользователя, вследствие чего процедура может эффективно заказывать себе память. Освобождение памяти выполняется автоматически при возврате из процедуры.

Стек связующей информации. Стек связующей информации предназначен для временного хранения данных о запускающей процедуре, обеспечивающих возврат к ней после отработки запускаемой процедуры. При защищенном программировании пользователь не должен иметь возможности изменять эту информацию, поэтому стек доступен только операционной системе и аппаратуре. Принцип организации стека связующей информации аналогичен стеку процедур.

Организация стеков рассматривается в приложении 5.

Выполнение процедурных переходов

Процедурный переход заключается в запуске новой процедуры из некоторой точки текущей процедуры, который может произойти по сигналам прерывания или по командам программы. В первом случае осуществляется вход в аппаратурный обработчик, программные операции позволяют осуществить переход к любой процедуре.

Процедурный переход в общем случае включает следующие действия:

- вызов в буфер команд и устройство управления строки программного кода новой процедуры;
- запоминание всей информации о текущей процедуре, необходимой для восстановления состояния процессора на момент вызова процедуры;

- организацию в РгФ окна для запускаемой процедуры, включающего регистры, содержащие фактические параметры (для процедуры с параметрами);
- переключение окна стека пользователя; для вызванной процедуры в памяти организуется новый стек пользователя, удаляемый при возврате из процедуры;
- переключение контекста процедуры.

Возврат из процедуры также является разновидностью процедурного перехода. Все необходимые данные для возврата к продолжению прерванной процедуры считываются из стека связующей информации.

Процедурный переход по командам программы делится на два этапа: подготовка перехода и фактический переход. Подготовка перехода рассмотрена ранее, она реализуется одной из нескольких операций в зависимости от способа получения информации о запускаемой процедуре.

Фактический процедурный переход выполняется операцией вызова CALL.

Операция вызова процедуры (CALL). Вызов задается в слоге CS1 кодом операции и кодом wbs — расстоянием в квадрословах от базы текущей процедуры до базы запускаемой процедуры, которое в дальнейшем обозначается символом Δ.

При выполнении операции в регистровой части стека связующей информации организуется новое окно путем выделения очередных двух регистров CR. Одновременно модифицируется указатель аппаратурной вершины стека связующей информации УВСС (PCSHTP), содержащий количество занятых регистров в файле CF. Запоминается информация о текущей процедуре: указатель команды продолжения процедуры, ее контекст (индекс дескрипторов модуля компиляции (CUIR), в который входит текущая процедура), база окна в РгФ (код wbs), флаг наличия в ее области вещественных 80, состояние предикатного файла, дескриптор стека пользователя, база вращающейся области и регистр состояния процессора. Вся эта информация упаковывается в два квадрослова.

В случае исчерпания файла связующей информации инициируется процесс откачки, после чего выполняются указанные действия. После запоминания информации для возврата происходят переключение указателя команды на адрес начала новой процедуры, формирование дескрипторов окна

связующей информации, переключение модуля компиляции (обновление при необходимости спецификаторов) и другие действия.

База окна в РгФ для запускаемой процедуры определяется прибавлением значения Δ к базе текущего окна. Суммирование осуществляется по модулю, равному количеству отводимых на АВС пар регистров (квадрослов). С выделением окна происходит приращение указателя УВС (PSHTP) на величину Δ, чем фиксируется положение дна регистровой части стека процедур.

Следует отметить, что размер окна определяется количеством фактических параметров, передаваемых запускаемой процедуре, то есть новое окно — это часть окна запускающей процедуры, содержащая передаваемые параметры (рис. 3.6). Но кроме параметров процедура использует локальные данные и промежуточные результаты, поэтому необходимо окно большего размера. Расширение окна выполняется операцией SETWD, содержащей требуемый его размер. При переключении фрейма в защищенном режиме на единицу увеличивается уровень процедуры в стеке пользователя (psl).

Операция возврата из процедуры (RETURN). Операция выполняет подготовленную передачу управления запустившей процедуре. Порядок возврата подчиняется стековой дисциплине, то есть переход осуществляется на последнюю неоконченную процедуру.

Действия, которые выполняются в ходе операции возврата, идентичны рассмотренным ранее, но все данные считываются из регистров файла связующей информации. Так, на регистр указателя команды записывается адрес команды, с которой необходимо продолжить выполнение прерванной процедуры, регистры состояния процессора также восстанавливаются данными, находящимися в регистрах CR, регистры группы модуля компиляции, если номер модуля меняется, считываются из таблицы модулей компиляции.

База окна в РгФ процедуры, в которую выполняется переход (назовем ее для краткости процедурой возврата), определяется вычитанием значения Δ из текущей базы, при этом на такую же величину уменьшается УВС (PSHTP.ind).

Переключение фрейма обеспечивает восстановление дескриптора свободной области стека пользователя на момент, предшествующий запуску процедуры, из которой осуществляется возврат. При выполнении возврата восстанавливаются предикатный файл, регистр состояния процессора и флаги, определяющие режим работы прерванной процедуры.

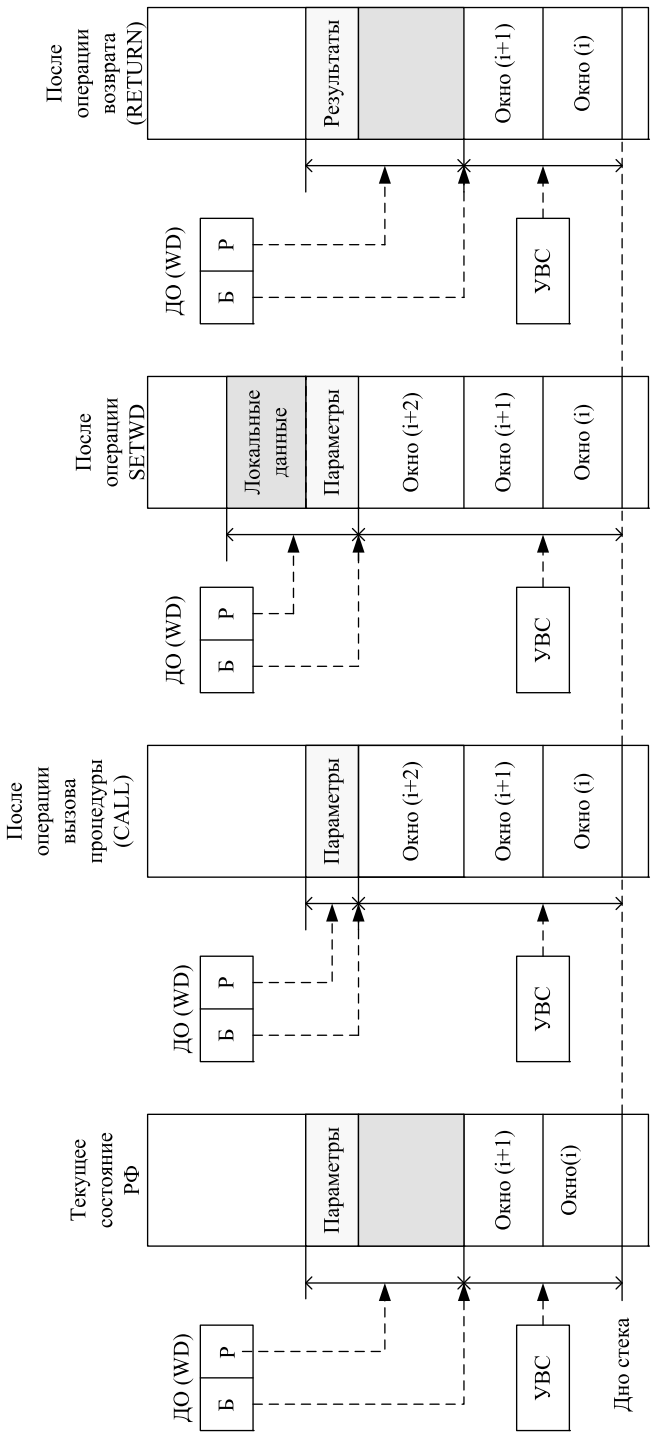


Рис. 3.6. Принцип переключения окон при процедурных переходах

Кроме рассмотренной операции возврата системой команд предусмотрена операция возврата из аппаратурного обработчика прерываний (операция DONE).

Аппаратные операции откачки и подкачки применительно к верхушке стека

Аппаратурные операции формируются устройством управления и выдаются в основной конвейер на стадии базирования адресов операндов (стадия БА). Рассмотрение аппаратурных операций в этом месте обусловлено методическими соображениями — они отсутствуют в программном коде и формируются при возникновении соответствующей ситуации.

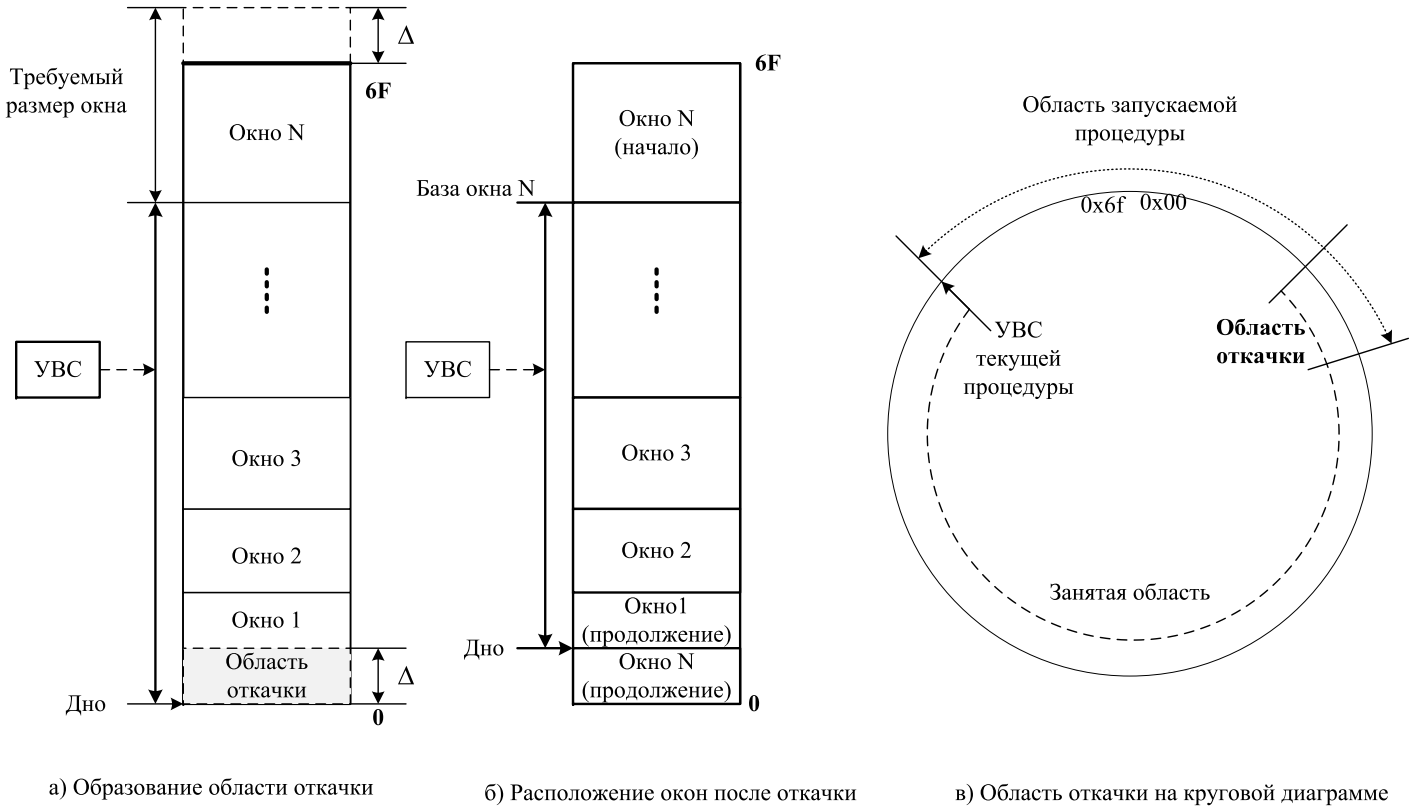
Системой команд предусмотрены три такие операции. Одна из них имитирует вставку в широкую команду «пустых» простых команд (операция BUBBLE), две другие осуществляют откачку (SPILL) данных из аппаратурной вершины стека (стека процедур или стека связующей информации) и подкачку (FILL) в нее данных из части стека, находящейся в памяти.

Операция откачки. Инициируется операцией очистки стека процедуры, стека связующей информации или возникает при выполнении операции ввода в новую процедуру в случае нехватки ресурса свободных регистров при организации окна файла (RF или CF).

Поскольку структуры стеков процедур и связующей информации одинаковы, принцип выполнения операции откачки рассмотрим применительно к стеку процедур. Образование области откачки показано на рис. 3.7, где она показана на круговой диаграмме. Применение ее основано на циклическом выделении ресурса регистрового файла при образовании окон: после использования регистра с номером 0x6f следующим номером будет 0x00 (границы окон выровнены по квадрослову, поэтому количество пар регистров равно 112).

Для откачки используются указатели аппаратурной вершины стека УВС (PSHTP) и УС (PSR). Первый из них указывает на дно стека процедур — регистр, начиная с которого происходит откачка. В указателе стека имеется поле, где хранится индекс начала свободной области. Откачка производится с точностью до двух регистров РгФ. Детали процесса откачки связаны с особенностями хранения данных в АВС и памяти (приложение 6).

Операция подкачки. Операция подкачки формируется в случае отрицательного значения указателя АВС (PSHTP), полученного в результате



а) Образование области откачки

б) Расположение окон после откачки

в) Область откачки на круговой диаграмме

Рис. 3.7. Принцип образования области откачки регистрового файла:
а — образование области откачки; *б* — расположение окон после откачки;
в — область откачки на круговой диаграмме

переключения с окна текущей процедуры на окно процедуры возврата. При этом вычисляется новое значение указателя:

$$УВС = УВС - \Delta,$$

где УВС (PSHTP) — число занятых регистров под окном завершившейся процедуры; Δ (wbs) — расстояние между базой окна завершающейся процедуры и базой окна процедуры возврата. Отрицательное значение указателя можно интерпретировать как попадание его в область незанятых, пустых регистров РгФ.

Состояние РгФ и условное положение указателей УВС (PSHTP) при выполнении команды возврата, обеспечивающей переключение процессора с выполненной (текущей) процедуры на процедуру, в которую осуществляется возврат, иллюстрирует рис. 3.8. Точками показано окно текущей процедуры. Указатель УВС для процедуры возврата на рисунке находится за пределами занятой области РгФ в окне выполненной процедуры. После перехода к команде продолжения прерванной процедуры это окно будет ликвидировано.

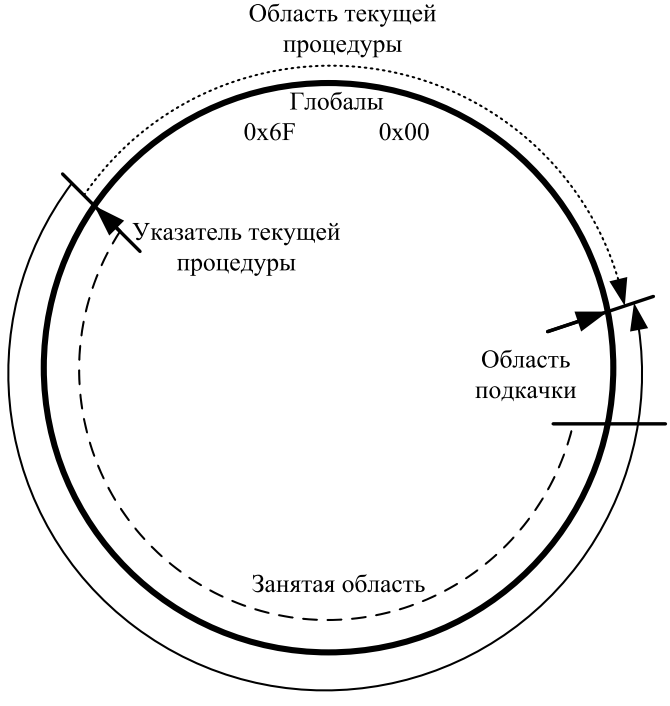


Рис. 3.8. Принцип образования области подкачки

Выход указателя за границу занятой области РгФ явился следствием предыдущей откачки данных из окна процедуры, в которую осуществляется возврат, потому должна быть предварительно выполнена подкачка по указателю АВС этих данных. Область подкачки располагается между нижней границей занятой области для текущей процедуры (дном) и указателем АВС для процедуры возврата. Отрицательное значение УВС и является количеством данных, которое необходимо подкачать.

Процесс подкачки начинается после поступления команды, следующей за операцией возврата, на стадию БА. Адрес по памяти подкачиваемого значения определяется уменьшением УС (PSR.ind) на размер одного двойного слова (8 байт). Во время подкачки данные из вершины области памяти передаются в дно регистровой части стека процедур. Характер процесса подкачки определяется типом подкачиваемых данных (приложение 6).

3.2.8. Арифметико-логическое устройство

Назначение и структура

Арифметико-логическое устройство микропроцессора предназначено для выполнения арифметических и логических операций обработки данных различных форматов и типов. Помимо этих традиционных операций на устройство возложены задачи преобразования адресной информации, а также отработки команд обращения к памяти на считывание и записи данных. Одной из особенностей АЛУ является выполнение им комбинированных операций с тремя аргументами.

Комбинированная операция является парой связанных обычных операций, закодированных как одна операция с тремя операндами. В паре различаются операции первой и второй стадий. Обе они являются двухместными, при этом первый и второй операнды комбинированной операции являются соответственно первым и вторым операндами операции первой стадии, третий операнд является первым операндом операции второй стадии, а результат первой стадии является вторым операндом операции второй стадии.

Кроме того, в состав АЛУ включены устройства для обработки визуальной и звуковой информации. Их операнды представляют собой упаковки значений байтового, полусловного или словного форматов. Поскольку разрядность операнда равна 64, упакованный формат может содержать 8, 4 или 2 значения. Операция выполняется одновременно над всеми значениями. Поэтому введены понятия скалярной операции (операции над

одиночными значениями) и операции с упакованными числами — целыми или вещественными.

Одновременное выполнение простых операций широкой команды обеспечивается шестью АЛК, каждый из которых имеет определенный набор исполнительных устройств. Поскольку операции по частоте использования отличаются друг от друга (некоторые весьма популярны, другие используются сравнительно редко), нет смысла делать каналы одинаковыми по набору исполнительных блоков. Поэтому, например, операция деления, довольно редко используемая, но требующая значительного объема оборудования, выполняется только в одном канале. Напротив, операции сложения целых чисел, сдвига, логические операции выполняют все каналы АЛУ.

В состав каждого АЛК входят два типа исполнительных устройств: обработки целых чисел (целочисленные устройства) и вещественных чисел (устройства с плавающей точкой, или вещественные устройства). Комбинированные целочисленные операции выполняются с помощью двух последовательно включенных целочисленных устройств. Подобные операции с плавающей запятой выполняются путем коммутации двух устройств (например, умножения и сложения).

Канал получает операнды из РгФ, в него же помещаются результаты операций. Для того чтобы иметь возможность одновременно отрабатывать запросы к регистровому файлу, каждый АЛК имеет отдельные порты по чтению операндов и записи результатов. Поскольку организация порта требует значительного объема оборудования, его сокращение достигнуто разделением каналов АЛУ (и регистрового файла) на два кластера по три арифметико-логических канала в каждом. На рис. 3.9 представлена структура кластера АЛУ. Все каналы кластера объединены общей шиной байпаса. Кроме этого, каждый канал может получить в качестве операндов результаты, полученные в соседнем кластере.

В отличие от других архитектур, здесь для выполнения операций над целыми и вещественными числами используется общий регистровый файл, что позволяет более эффективно использовать его объем и устранить дополнительные пересылки между различными регистровыми файлами. Однако при таком объединении возможны конфликты между целочисленным и вещественным устройствами канала при записи результатов в регистровый файл. При конфликте приоритет отдается целочисленным операциям. Результаты вещественных операций помещаются в выходной буфер (Бф2), где хранятся до момента освобождения шины.

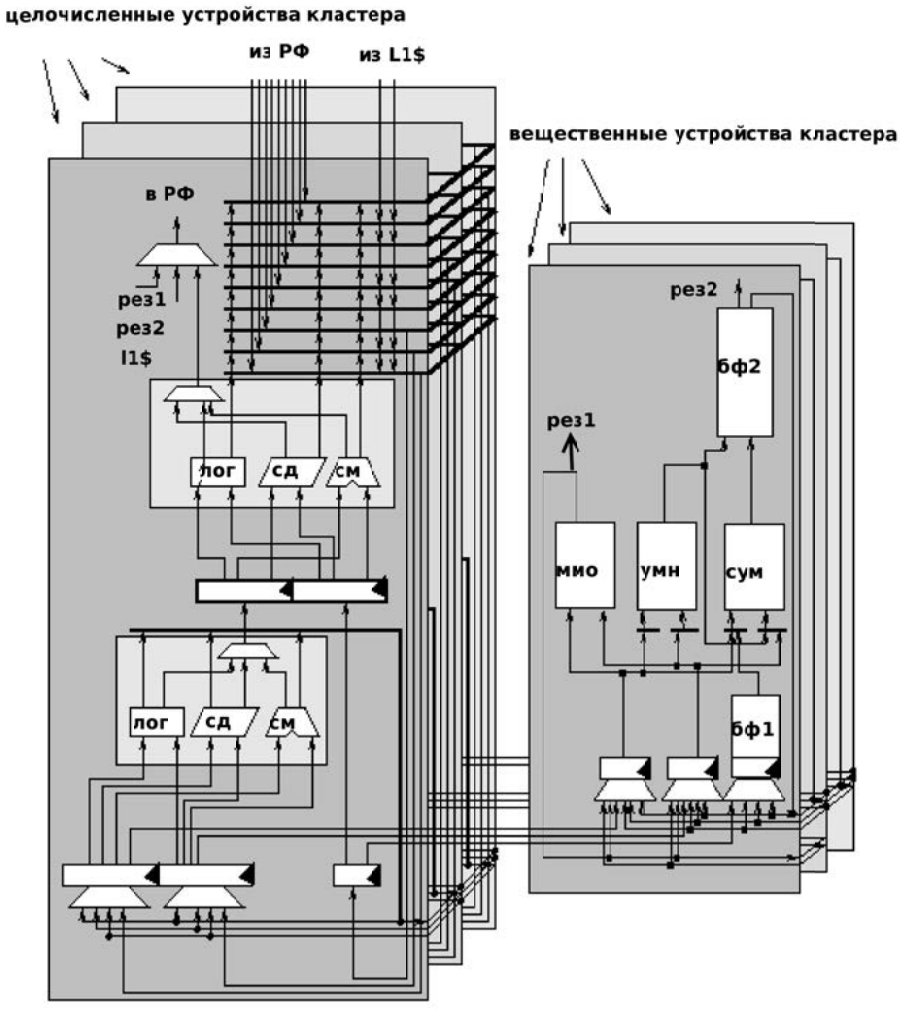


Рис. 3.9. Кластер арифметико-логического устройства

Каналы АЛУ имеют конвейерную организацию. Большинство операций выполняются за один такт. Остальные операции требуют прохождения нескольких стадий конвейера, и время их отработки занимает соответствующее число тактов.

Устройства целочисленной арифметики

На рис. 3.9 показана структурная схема целочисленных устройств (исполнительных блоков) одного канала.

К устройствам целочисленной арифметики относятся устройства, выполняющие операции над целыми числами (Лог, Сд, СМ), и устройство преобразования адресных типов (ПАТ), которое на схеме не показано. Исполнительные блоки этой группы отрабатывают логические команды (Лог), сдвигов (Сд), сложения, вычитания и сравнения (СМ). Операндами для них могут быть целые с форматом одинарного и двойного слова. Операции целочисленного умножения и деления выполняются в устройствах вещественной арифметики.

Устройство преобразования адресных типов оперирует адресными словами. Результат выполнения — также адресное слово, например дескриптор на подмассив исходного массива, метка процедуры и др. Дескриптор формата квадрослова формируется одновременно двумя каналами. Один из них формирует младшую половину дескриптора, второй — старшую. На блок ПАТ возложено выполнение операций чтения регистра статуса и записи в него, а также операций с кэш-памятью таблиц, чтение/запись и вычеркивание строк соответствия.

Блоки Лог, Сд, СМ имеются в каждом канале, устройство ПАТ — в четырех каналах. В любом канале за такт запускается только одно устройство — то, которое выполняет поступившую в канал операцию.

В каналы, выполняющие целочисленные комбинированные операции, введен второй этаж целочисленных устройств (Лог, Сд, СМ). Промежуточные результаты устройств первого этажа поступают на входы блоков второго этажа. Вследствие этого результаты комбинированных команд доступны только с уровня второго этажа.

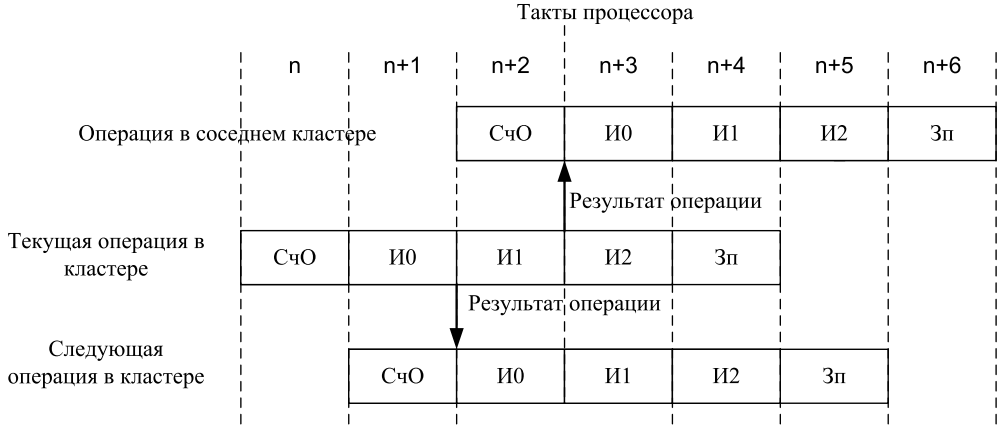
В двух каналах кластера имеются блоки обращения к памяти на считывание и записи Сч/Зп. В этих устройствах с помощью сумматоров формируются виртуальные адреса, которые выдаются в устройство управления памятью для окончательного выполнения операции считывания или записи.

В операции записи подлежащее записи число читается непосредственно или передается через шины байпаса целочисленных и вещественных устройств в устройство управления памятью. Операции чтения и записи с форматом квадрослова выполняются одновременно в двух каналах.

Временная диаграмма работы исполнительных блоков целочисленной арифметики приведена на рис. 3.10. Конвейер (рис. 3.10, *а*) состоит из четырех стадий обработки (стадия СчО, в которой организуется считывание операндов, принадлежит общей части конвейера). В первой исполнительной стадии И0 осуществляется прием операндов на входные регистры

каналов. Заметим, что, согласно схеме, изображенной на рис. 3.9, с входных регистров данные подаются не только в исполнительные блоки целочисленной арифметики, но и в блоки обработки вещественных чисел и устройства мультимедийной обработки. Данные в качестве операндов могут поступить из шин байпаса, а также из отдельной шины байпаса, связывающей выходы исполнительных блоков Лог, Сд, СМ с их входами. Благодаря этой шине результат однотактной предыдущей операции может быть использован в качестве операнда уже следующей операцией, то есть для нее обеспечено нулевое время считывания этого операнда (рис. 3.10, *б*).

а) Конвейер выполнения целочисленных операций



б) Доступность результата текущей однотактной операции

Рис. 3.10. Временная диаграмма выполнения целочисленных операций: *а* — конвейер; *б* — доступность результата текущей однотактной операции

Почти все операции целочисленной арифметики выполняются за один такт, операции преобразования адресных типов (блок ПАТ) и комбинированные операции — за два такта. Результат однотактных операций доступен для «своего» кластера в стадии И0 (благодаря отдельной шине байпаса), для другого кластера — в стадии И1 после записи его в регистр И1. В стадии записи (Зп) результаты операций устройств целочисленной арифметики

поступают на входные регистры регистрового файла и записываются в него. Через такт их можно получить, лишь прочитав из регистрового файла.

Временные диаграммы двухтактных операций, в число которых входят и комбинированные операции, по существу не отличаются от рассмотренных ранее. Следует иметь в виду, что их результаты записываются на выходные регистры в стадии И2 конвейера.

Таким образом, устройства целочисленной арифметики выполняют весь набор операций обработки целых чисел, кроме умножения и деления. Почти все команды отрабатываются за один такт. Кроме того, на них возложено выполнение комбинированных (сцепленных), представляющих собой звено из двух операций, операций преобразования адресной информации и обращения к памяти. Время их выполнения (за исключением последних операций) составляет два такта.

Устройства вещественной арифметики

Для выполнения операций над данными вещественного типа предназначены устройства вещественной арифметики. В микропроцессоре обрабатываются вещественные формата 32 (Floating point Word, FW — вещественное слово), 64 (Floating point Double Word, FDW — вещественное двойное слово), 80 (Floating point eXtended, FX — вещественное расширенного формата). В дальнейшем для удобства будем пользоваться обозначениями вещественных форматов B32, B64 и B80.

Помимо стандартных операций с вещественными числами в микропроцессоре выполняются операции выдачи большего (меньшего) числа из двух, а также значения квадратного корня. Предусмотрены операции сравнения с выдачей в качестве результата предиката, записываемого в предикатный файл, или с установкой флагов. Флаги помещаются в результат операции, и результат записывается в регистровый файл.

Все вещественные операции соответствуют международному стандарту IEEE-754 на двоичную арифметику с плавающей точкой. Особо следует отметить, что в отличие от других микропроцессоров в микропроцессоре «Эльбрус» этот стандарт реализован полностью аппаратно (например, в микропроцессорах фирмы Intel значительная доля его требований реализована с помощью микропрограммы). Поэтому для выполнения некоторых вещественных операций (с денормализованными операндами и/или результатом, вызывающим переполнение и др.) в микропроцессоре «Эльбрус» требуется одна команда с плавающей точкой, в то время как в микропроцессоре фирмы Intel требуется несколько десятков или даже сотен операций.

В операциях с 80-разрядными вещественными числами первый операнд всегда имеет формат B80, а второй операнд и результат могут иметь любой из вещественных форматов. Приведение формата второго операнда к формату B80, а формата результата — в соответствие с выполняемой операцией производится в процессе отработки команды.

Кроме рассмотренных операций имеется целый ряд других, например операции преобразования форматов и типов чисел.

Что касается обработки мультимедийных данных, то более подробно принцип выполнения таких операций рассматривается в дальнейшем. Здесь лишь отметим, что в системе команд имеется довольно большой перечень операций обработки упакованных значений.

Блок-схема вещественных устройств одного канала приведена на рис. 3.9. Устройство с плавающей запятой содержит:

- устройство сложения (Сум);
- устройство умножения (Умн);
- устройство мультимедийной обработки (ММО);
- устройство деления (не показано);
- три входных регистра;
- входной буфер (Бф1);
- выходные буферы канала (Бф2).

Устройство сложения выполняет операции сложения (вычитания), сравнения, преобразования различных форматов и типов и др. Например, предусмотрены операции преобразования формата FX в целое 32/64, а целого 32/64 — в формат FX и др.

Устройство умножения предназначено для выполнения операций умножения вещественных и целых чисел.

Устройство мультимедийной обработки реализует операции с упакованными значениями и состоит из четырех типов устройств, которые выполняют:

- команды сложения/вычитания, сравнения и логические операции над упакованными целыми значениями;
- логические операции, операции сдвига и операции с полями;

- операции умножения упакованных полуслов;
- операции сравнения вещественных чисел форматов B32 и B64, как одиночных, так и упакованных.

Также оно формирует результат, записываемый в регистровый файл.

Устройство деления выполняет операции деления и извлечения квадратного корня целых и вещественных чисел.

Следует заметить, что операции с упакованными вещественными значениями B32 и B64 предусмотрены во всех арифметических вещественных устройствах — сложения, умножения и деления.

Операции с упакованными вещественными принято использовать в так называемых макрооперациях, содержащих две или четыре операции с упакованными данными. Операнды и результат макрооперации могут (но не обязательно) рассматриваться как формат квадрослова.

При выполнении операций деления и извлечения квадратного корня с операндами B32 макрооперация содержит четыре операции с упакованными данными. Они запускаются последовательно раз в два такта. В случае операции деления чисел B64 макрооперация содержит две операции с упакованными данными. Они запускаются последовательно раз в два такта.

Если операция не выполняет деление или извлечение квадратного корня, то макрооперация содержит две операции с упакованными данными, которые размещаются в одной широкой команде и запускаются одновременно. Если операция с упакованными данными имеет дело с двумя значениями B32 в каждом операнде и выдает два результата B32, она может рассматриваться как две подоперации. Макрооперация выполняется в двух каналах одного и того же кластера. Она считается завершенной, когда завершается последняя составляющая операция.

Выходной буфер (Бф2) необходим для хранения результатов операции при конфликтах по выходной шине АЛК, при этом целочисленное устройство и ММО имеют наивысший приоритет. Как и целочисленные устройства, устройства вещественной арифметики одного кластера объединены собственной шиной байпаса. Результаты их операций с упакованными операндами по шине байпаса передаются с выходов АЛК на их входы. Число шин и метод их коммутации обеспечивают бесконфликтную коммутацию всех выходов на все входы.

3.2.9. Подсистема памяти
Основные устройства

Проектирование подсистемы памяти микропроцессора выполнялось из расчета на то, что параллельное исполнение, являющееся базовым принципом архитектуры «Эльбрус», многократно увеличивает поток обмена данными с памятью. Это требует использования ряда решений, существенно увеличивающих ее пропускную способность, в первую очередь за счет локализации данных многоуровневой иерархии памяти и увеличения числа параллельных каналов доступа к различным уровням [23]. Основными компонентами подсистемы памяти микропроцессора «Эльбрус» являются кэш-память первого и второго уровня (L1$ и L2$), устройство управления памятью (УУП) (MMU) и устройство обращения к массивам (УОМ) (AAU).

Кэш данных первого уровня L1$

Кэш данных L1$ первого уровня состоит из двух одинаковых устройств, которые размещены в разных кластерах микропроцессора. Каждое из них обслуживает запросы от исполнительных устройств своего кластера, но при этом оба устройства содержат одинаковые данные. Это свойство обеспечивается дублированием записи данных в кэш своего и соседнего кластеров. Устройство в кластере 0 обслуживает арифметико-логические каналы 0 и 2, а устройство в кластере 1 — каналы 3 и 5. Физически кэш данных L1$ реализован как два 2-портовых блока (по одному в каждом кластере) размером 64 Кбайт каждый.

На рис. 3.11 представлена структурная схема кэша L1$ (для одного канала) и показано использование отдельных разрядов виртуального адреса при обращении в кэш. Устройство содержит память данных и память тегов, имеющих частично ассоциативную организацию. Память тегов представляет собой ассоциативную часть, память данных — ответную. Кроме того, в устройстве имеется память битов LRU, которые используются алгоритмом вытеснения. Все три памяти содержат по 512 строк.

Память данных и память тегов имеют по два универсальных порта чтения и записи. Это позволяет одновременно выполнять два обращения по чтению или одно обращение по чтению и одно обращение по записи от своего или соседнего кластера.

В каждой строке памяти тегов содержится по 4 тега для соответствующих столбцов данных.

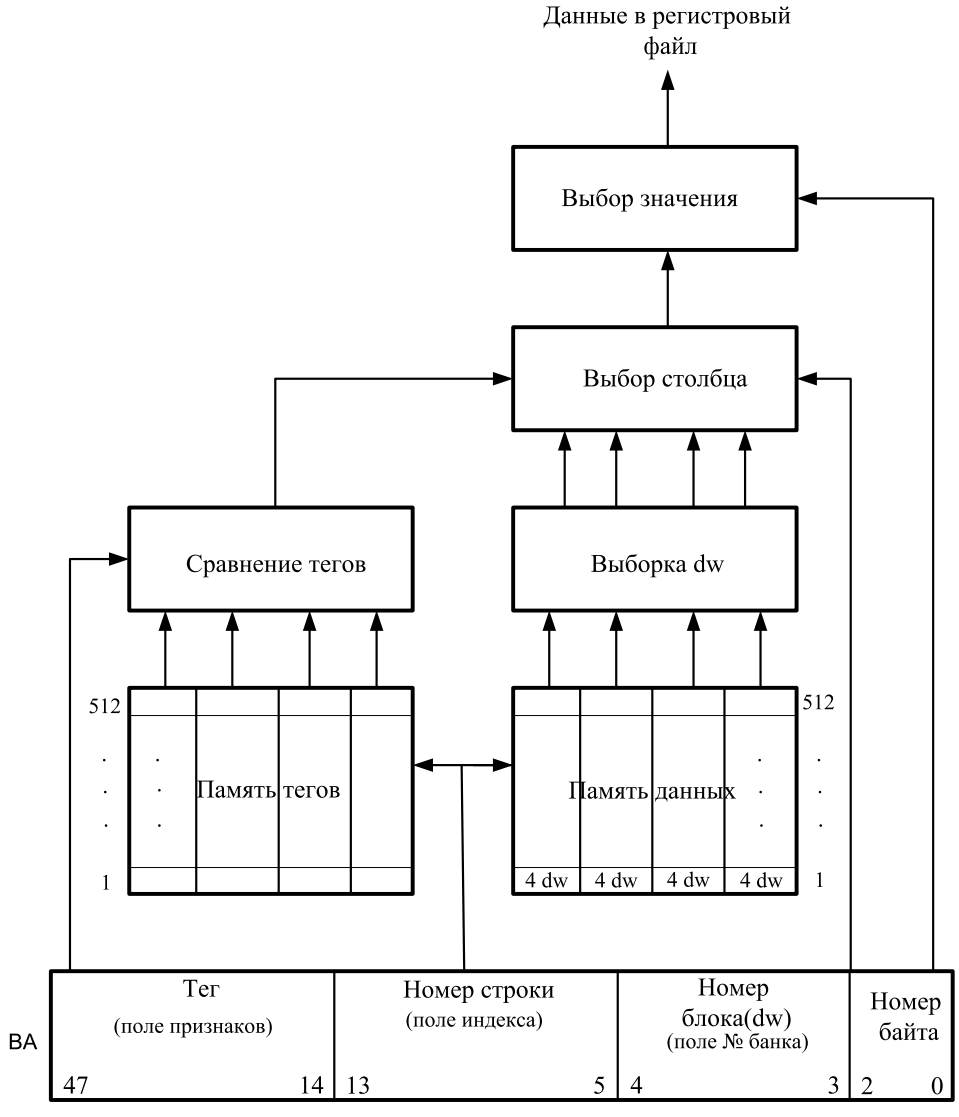


Рис. 3.11. Структурная схема кэша данных первого уровня L1$

Память LRU с длиной строки 3 разряда хранит историю обращений к блокам данных каждой строки (возраст строки).

Память данных разделена на 4 столбца по 512 строк каждый. Размер строки в одном столбце — 4 двойных слова (dw), что составляет 32 байта, или один блок. Поэтому 48-разрядный виртуальный адрес BA[47:0] обращения к кэшу представлен в виде четырех полей:

- 34-битного поля признака (тега), кодирующего номер области основной памяти с отображаемым блоком;
- 9-битного поля индекса (номера строки), определяющего строку (набор) в памяти данных и тегов ($2^9 = 512$);
- 2-битного поля с номером двойного слова в блоке ($2^2 = 4$);
- 3-битного поля с номером байта в двойном слове ($2^3 = 8$).

Алгоритм функционирования кэш-памяти L1$ при поиске данных в ней сводится к следующему.

- При обращении по индексу ВА[13:5] выбирается строка в памяти данных и тегов. Затем в четырех строках данных параллельно выбираются заданные двойные слова.
- Входной тег сравнивается с тегами выбранной строки на четырех схемах сравнения. Номер схемы сравнения соответствует номеру столбца в памяти тегов и памяти данных.
- При наличии в кэше требуемых данных одна из схем сравнения вырабатывает сигнал совпадения. Этот сигнал, управляя выходным коммутатором, обеспечивает выдачу данных из соответствующего столбца строки памяти данных. Таким образом среди двойных слов выбирается искомое двойное слово, из которого в соответствии с форматом обращения берется искомое значение. В противном случае фиксируется промах в кэш-памяти данных L1$ и блок читается из кэша второго уровня или оперативной памяти.

Физическая структура памяти данных L1$ показана на рис. 3.12. Память данных в одном кластере представляет собой четыре двухпортовых блока статической памяти (m0, m1, m2, m3), разбитых на 4 области каждый (b0, b1, b2, b3). Одна область хранит данные определенных сета и банка. Сет представляет собой 4 одноименных блока разных банков.

Если пронумеровать сеты s0, s1, s2, s3, а банки — b0, b1, b2, b3, то можно составить таблицу соответствия (табл. 3.5). В данном случае прямой адрес двойного слова в кэше — это совокупность индекса по физическому носителю и номера этого носителя (m[1:0], index_m[10:0]). Напомним, что с точки зрения системы команд оперируют другим набором значений: set, index, bank (s[1:0], index[8:0], b[1:0]). Bank — это номер банка — разряды ВА[4:3], index — разряды ВА[13:5]. Таблица позволяет переводить одно представление в другое.

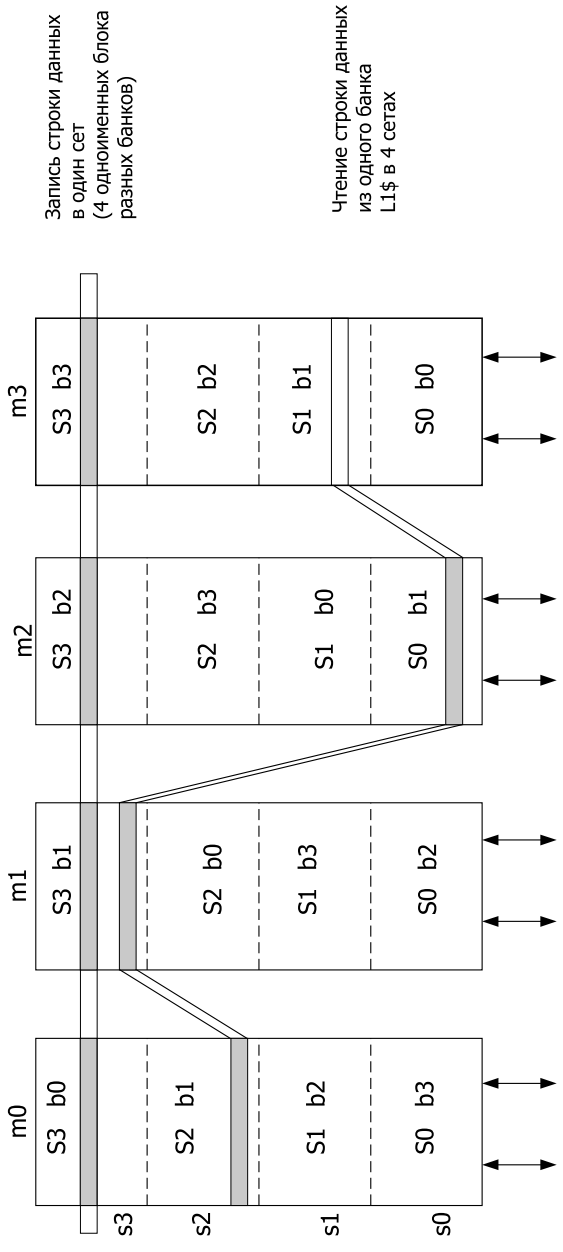


Рис. 3.12. Физическая структура памяти данных L1$

Таблица 3.5. Соответствие адреса в физической и программной модели L1$

Сет	Блок статической памяти			
	m0	**m1**	**m2**	**m3**
s3	b0	b1	b2	b3
s2	b1	b0	b3	b2
s1	b2	b3	b0	b1
s0	b3	b2	b1	b0

Поскольку обращение в кэш возможно как по виртуальному, так и по физическому адресу, тег содержит дополнительные поля, показанные на рис. 3.13:

- addr — адрес тега;
- бит addr_is_va указывает, что адрес тега является виртуальным (иначе — физическим) адресом;
- root — признак принадлежности адресов и данных виртуальному пространству Intel (root = 0 соответствует виртуальному пространству Elbrus);
- context — 12-разрядное поле контекста, которое описывает область памяти, непосредственно доступную текущей задаче;
- бит G (global) отменяет сравнение контекстов, то есть данная страница считается общей для всех задач;
- бит pagesize_small указывает, что страница, которой принадлежит этот виртуальный адрес, — малая (4 Кбайт), иначе — большая (4 Мбайт). Для физических адресов этот бит должен быть всегда равен 1;
- dontuse (не использовать) — признак дефекта в строке. Если установлен, строка не используется;
- vtag — бит значимости — указывает, значима или не значима данная строка.

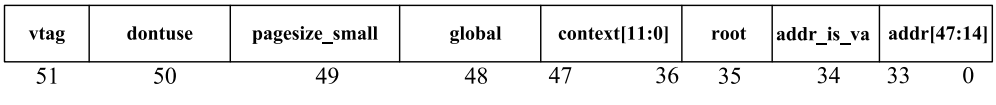


Рис. 3.13. Структура тега L1$

Дополнительные поля тега хранятся в специальном массиве достоверности. Физические адреса поступают из таблицы страниц (TLB).

Считается, что запрос попал в кэш, если выполняются следующие условия:

- совпадает тип обращения (физический адрес/виртуальный адрес);
- совпадает тип обращения («Эльбрус»/Intel);
- виртуальный адрес совпадает с точностью до строки (VA[47:5]).

Если происходит обращение типа «Эльбрус» по виртуальным адресам и у данной страницы нет признака global, то должен совпадать контекст.

Механизм старения определяет, какие блоки в памяти данных могут быть замещены. Алгоритмом замещения в L1$ является REDUCED LRU. Каждой строке L1$ приписаны три признака замещения (биты lru). Один из них указывает, обращение к какой паре столбцов (0 и 1 или 2 и 3) было последним. Каждый бит из двух остальных указывает, обращение к какому столбцу из пары (0 или 1, 2 или 3) было последним. Пример работы механизма старения рассмотрен в главе 2.

Значения разрядов «пара блоков» и «блок в паре» меняются на противоположные после назначения блока для замещения. Биты lru хранятся в отдельной памяти — считываются при каждом запросе и модифицируются при попадании (hit) и заведении. Память битов lru имеет четыре порта по считыванию и четыре порта по записи, то есть нет ограничений по доступу, за исключением случая, когда заявки попадают в одну строку L1$. В этом случае модификация производится только для одного канала — в соответствии с приоритетом каналов. По сбросу все биты lru устанавливаются в 1.

В L1$ реализован программно-аппаратный механизм сокрытия физических дефектов, возникающих при изготовлении кристалла. Он позволяет маскировать дефекты в следующих структурах:

- памяти данных L1$;
- памяти тегов L1$ (кроме бита значимости vtag);
- память физической копии тегов L1$.

Формально можно представить, что строка L1$ состоит из данных, виртуального и физического тегов. Если хотя бы в одной из этих частей

обнаружена ошибка, строка помечается битом dontuse (не используется) в теге. На основании этого признака данная строка не будет использоваться при работе кэш-памяти — физические дефекты в этой строке будут скрыты. Биты dontuse заполняются в начале работы процессора (до включения L1 кэша). Предусмотрены два механизма заполнения битов dontuse — программно-аппаратный и полностью программный. После прописывания битов dontuse можно включать L1$ и работать с ним.

Кэш второго уровня L2$ для хранения команд и данных

Кэш второго уровня L2$ предназначен для хранения команд и данных. Он имеет объем 256 Кбайт, время доступа для скалярных данных (при промахе в L1$) 9 тактов, частично ассоциативную организацию (4 колонки по 256 строк, размер блока 64 байта) и может обслуживать до четырех запросов каждый такт.

Структурная схема кэша L2$ представлена на рис. 3.14. Он состоит из четырех банков и имеет четырехканальную организацию. Каждый из банков содержит 4 столбца памяти данных, 4 столбца памяти тегов, буфер записи STB (STore Buffer), буфер запросов в MAU (Memory Access Unit — устройство работы с памятью) на чтение — MRQ (Memory Read Queue — очередь запросов в память на чтение), буфер запросов в MAU для операций откачки WBQ (Write-Back Queue — очередь запросов откачки данных). Выходные регистры запросов на чтение и запись, а также выходные регистры данных являются общими для всех банков. Обращение в кэш возможно только по физическому адресу.

Теги, данные и адреса назначения запросов принимаются на входные буферы каналов, откуда выбираются после запуска заявки на обслуживание.

Коды операций и индексы коммутируются на 4 банка в соответствии с разрядами [7:6] физического адреса (номер банка) и записываются в регистры очередей заявок, откуда затем выбираются схемами приоритета для запуска заявки на обслуживание. Переполнение буферов и очередей вырабатывает блокировку в MMU.

При отсутствии заявок в очереди банка запросы поступают на схему приоритета сразу и могут быть переданы на обслуживание с минимальной задержкой 1 такт. Схема приоритета запускает запросы на считывание таким образом, что суммарная ширина считываемых данных не превышает четырех двойных слов.

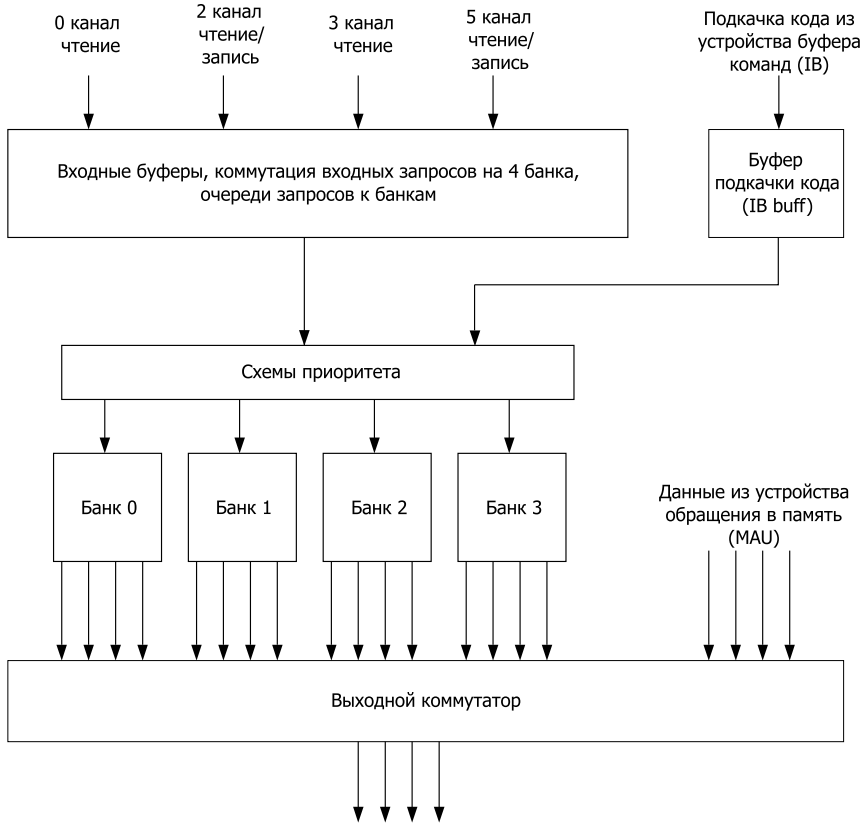


Рис. 3.14. Структурная схема кэша L2$

Кроме запросов от процессора на схемы приоритета банков поступают:

- собственные запросы (запись нового тега, изменение битов состояния, запрос операции откачки write-back, повторение запроса);
- внешние запросы поддержки когерентности (Coherent Read, Coherent Invalidate и др.);
- Refill — запись блока данных, пришедших из памяти;
- запросы от устройства буфера команд IB — чтение блоков данных (от одного до четырех блоков).

Таким образом, кэш второго уровня L2$ можно разделить на две основные части: арбитр L2 и банки L2. Арбитр L2 принимает запросы процессора и распределяет их по 4 банкам. При невозможности немедленного

обслуживания запросы буферизуются в очередях запросов перед банками L2. В каждом такте процессора на вход арбитра L2 может поступать до четырех запросов, из них до двух запросов по записи.

Структурная схема одного банка L2 представлена на рис. 3.15. Алгоритм функционирования банка L2$ при поиске данных в нем аналогичен рассмотренному ранее для L1$ и сводится к следующему.

1. При обращении по индексу (разряды [18:15] физического адреса (ФА)) выбирается строка в памяти данных и тегов.

2. Входной тег сравнивается с тегами выбранной строки на четырех схемах сравнения. Номер схемы сравнения соответствует номеру столбца в памяти тегов и памяти данных.

3. При наличии в банке требуемых данных одна из схем сравнения вырабатывает сигнал совпадения. Этот сигнал, управляя выходным коммутатором, обеспечивает выдачу данных из соответствующего столбца строки памяти данных.

Таким образом, в соответствии с параметрами операции обращения выбирается искомое значение. В противном случае фиксируется промах в L2$ и блок читается из оперативной памяти.

Формы операций обращения к памяти приведены в приложении 7.

Кэш второго уровня состоит из двух частей. Первая из них — L2-арбитр. Она включает компоненты «Входные буферы» и «Схемы приоритета» (см. рис. 3.14), осуществляет арбитраж запросов в кэш L2$ от исполнительных устройств процессора (CPU), устройства предварительной подкачки массивов (AAU), устройства доступа в память (MAU), банков L2 кэша, буфера команд (IB) и когерентных запросов. К функциям арбитра относятся и такие собственные операции, как откачка модифицированных строк в ОП и повтор запроса при возникновении блокировок.

Поступившие запросы принимаются на соответствующие входные регистры и с них согласно номеру банка попадают либо на выходной регистр арбитра соответствующего банка по байпасу (входной регистр — выходной регистр), либо в очередь запросов и буферы тегов и данных.

Очередь состоит из общего буфера на 8 запросов, четырех буферов на 7 запросов каждый, соответствующих банкам кэш-памяти, и четырех регистров между общим и банковыми буферами для организации байпаса.

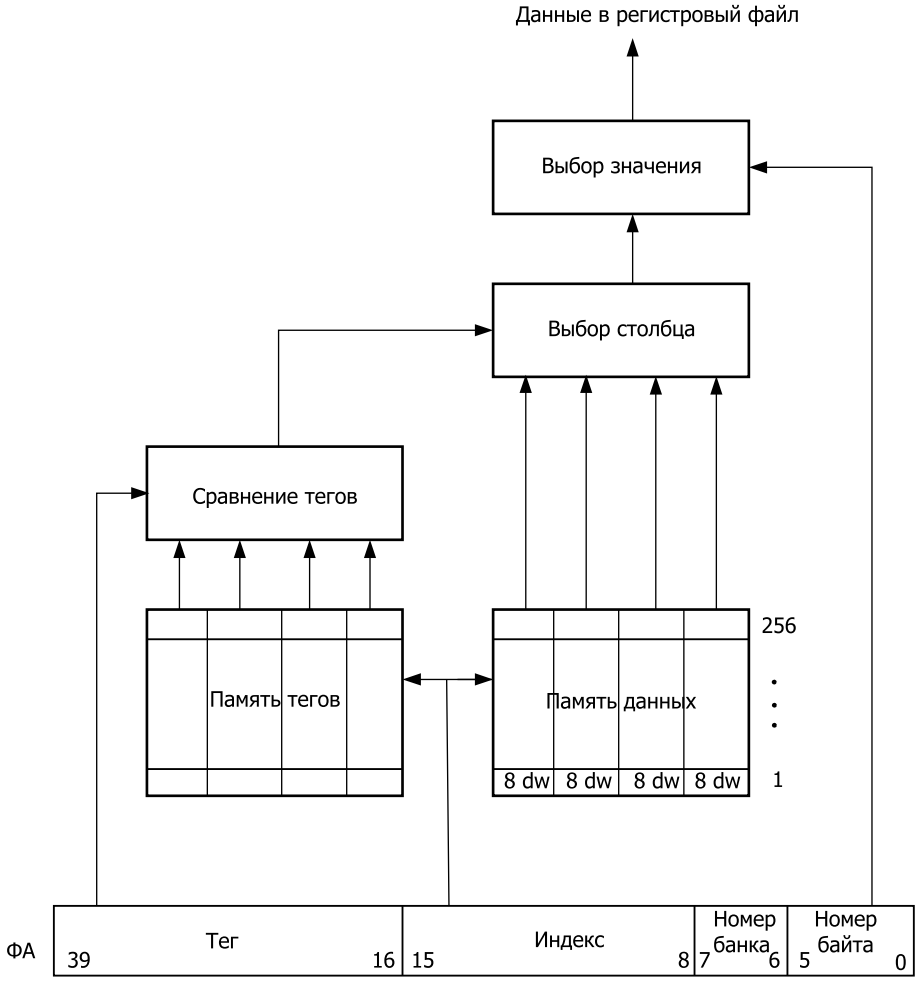


Рис. 3.15. Структурная схема одного банка кэш-памяти L2$

Поля тегов, адресов назначения и данных хранятся в отдельных буферах глубиной 16 и 12 регистров соответственно. Для тегов и адресов назначения выделены 4 буфера, соответствующие каналам 0, 2, 3 и 5, для данных — 2 буфера, относящиеся к каналам 2 и 5. Соответствующие индексы буферов тегов и данных хранятся с основной частью запроса в очереди.

Установлены следующие приоритеты обслуживания запросов (табл. 3.6).

Таблица 3.6. Приоритеты операций в L2-арбитре

Приоритет	Запрос	Источник
1	Заполнение строки Запись тегов и/или битов состояния	MAU L2arb
2	Запрос откачки блока (write-back)	L2arb
3	Когерентные запросы	MAU
4	Повтор запроса	L2arb
5	Запрос диагностического считывания данных (Invalidate & WbInvalidate)	Генерируются L2arb по счетчику после получения запроса от 2-го канала
6	Чтение данных Запись данных	CPU/AAU CPU/AAU
7	Чтение для IB	IB

За один такт из арбитра может быть выдано до четырех запросов записи или считывания (по одному на банк).

Из очереди на выходной регистр арбитра банка может быть принят один из трех ближайших (в порядке попадания в очередь) запросов при выполнении условий приема по байпасу и условий приема запросов из очереди. В остальных случаях первый запрос принимается безусловно, последующие — при несовпадении индексов с предыдущим.

Обеспечиваются следующие ограничения на порядок выполнения операций.

1. Чтения выполняются по отношению друг к другу в произвольном порядке, если нет конфликта по адресам.

2. Записи выполняются по отношению друг к другу в том порядке, в котором они находились в потоке инструкций. По возможности несколько записей выполняются за один такт.

3. Запись выполняется только после того, как были выполнены все предшествующие ей чтения.

4. Запросы по чтению или записи, в которых есть конфликт по адресам, выполняются в том порядке, в котором они находились в потоке инструкций.

5. Запросы с признаком ввода-вывода данных (IO) выполняются по одному за такт, то есть строго упорядочены по отношению друг к другу и не обгоняют остальные запросы.

Для того чтобы выдержать «интеловскую» модель памяти, все запросы на чтение с признаком регистрового файла и на запись нумеруются восьмибитовым числом. Запросы на чтение дополнительно отмечаются в 256-разрядной шкале. Далее запоминается номер последнего невыполненного чтения, и все последующие записи останавливаются в буфере записей до тех пор, пока оно не будет выполнено.

Аппаратура, управляющая порядком завершения записей, состоит из буферов FIFO емкостью 32 строки, содержащих номера запросов в шкале, номера соответствующих им строк в STB, номер двойного слова и значимости всех байтов в DW. Далее, если нет препятствий в виде более ранних запросов на запись, дается разрешение на запись. В идеальном случае можно выполнять по четыре записи за один такт. При переполнении буфера формируется управляющий сигнал, который блокирует выдачу записей в этот банк.

Другая часть кэш-памяти второго уровня, L2-кэш, хранит, считывает и записывает данные. Она состоит (см. рис. 3.14) из банков 0–3, выходного коммутатора данных и очередей запросов к MAU. Каждый банк содержит память данных — 64 Кбайт, разбитую на 4 столбца с частично-ассоциативным доступом. Каждый столбец состоит из 256 блоков размером по 64 Кбайт. Формат, в котором данные хранятся в L2-кэше, описан в приложении 8.

Наряду с рассмотренными составными частями важное место в кэше второго уровня занимают выходной коммутатор и буфер записи STB. Выходной коммутатор пропускает за один такт 4 двойных слова из одного банка, либо по 2 двойных слова из банков 0, 1, 2 или 3, либо по одному двойному слову из каждого банка. Согласно этому на выходной регистр арбитра пропускаются запросы на считывание в случае освобождения выходного регистра банка кэш-памяти.

Коммутатор обслуживает запросы от банков в соответствии с установленными для них приоритетами, которые автоматически изменяются в соответствии с круговой дисциплиной и директивно — по сигналам управления.

Буфер записи STB (Store Buffer) состоит из четырех «строк» — регистров данных с форматом двойного слова каждый. Каждому регистру данных соответствует регистр адреса, кода операции и атрибутов заявки, таким

образом, буфер может хранить данные, относящиеся к четырем кэш-блокам. Запись в STB производится с точностью до байта. Кроме значимости каждый байт имеет признак завершенности записи. Когда все значимые байты в строке имеют признаки завершения, строка становится кандидатом на перезапись в кэш или в ОП (для записей без размещения в L2$).

Запросы на считывание проверяются в кэше и в STB. При наличии данных в STB считанные из него данные накладываются на данные, считанные из кэша. При отсутствии данных в кэше и STB вырабатывается запрос в MAU для подкачки запрашиваемой строки из оперативной памяти. При отсутствии данных в кэше и наличии данных в STB чтение производится из STB.

Устройство управления памятью

Устройство управления памятью MMU обеспечивает следующие функции:

- трансляцию 48-разрядного виртуального адреса в 40-разрядный физический адрес, выполняемую устройством «кэш таблица страниц данных» (DTLB);
- поддержку различных контекстов;
- поддержку двух размеров станиц — 4 Кбайт и 4 Мбайт;
- аппаратный поиск в таблице страниц, выполняемый устройством обращения в таблицу страниц (Table Look-Up, TLU);
- контроль корректности выполнения заявок, чтобы защитить адресное пространство каждого процесса, запущенного операционной системой в одно и то же время.

Архитектура «Эльбрус» предоставляет пользователю два различных виртуальных пространства — первичное «Эльбрус» и вторичное Intel, поддерживаемые двумя различными таблицами страниц. Тип адресуемого виртуального пространства определяется кодом операции. Большинство операций предназначены для обращения в первичное виртуальное пространство.

Контекст, описывающий область памяти, непосредственно доступный текущей задаче и позволяющий осуществить защиту данных между различными пользователями, хранится на специальном 12-разрядном регистре контекста процессора (Context register) в устройстве обращения в таблицу

страниц TLU. Поле «контекст» используется для расширения виртуального пространства «Эльбрус». Из TLU поле контекста поступает в TLB и участвует в трансляции адреса.

Запросы на трансляцию в MMU поступают по 4 каналам параллельно. Приоритет синхронных каналов и их нумерация определяются в соответствии с расположением операций в широкой команде — 0, 2, 3, 5. При этом 2-й и 5-й каналы обслуживают заявки как по считыванию, так и по записи, а 0-й и 3-й каналы обслуживают только запросы по считыванию.

Кроме того:

- 0-й и 3-й каналы предназначены для обслуживания скалярных синхронных запросов и асинхронных запросов устройства обращения к массивам AAU и запросов буферов невыполненных асинхронных заявок по считыванию;
- 2-й канал предназначен для обслуживания скалярных и векторных синхронных запросов, а также асинхронных запросов устройства обращения в таблицу страниц TLU и устройства подпрограмм SRU;
- 5-й канал предназначен для обслуживания скалярных и векторных синхронных и асинхронных запросов по чтению и записи.

Какой именно синхронный запрос — скалярный от арифметико-логических каналов или векторный от устройства обращения к массивам AAU — коммутируется на вход DTLB, определяется компилятором.

Виртуальные адреса состоят из номера виртуальной страницы VPN (Virtual Page Number) и смещения внутри страницы (offset) (рис. 3.16).

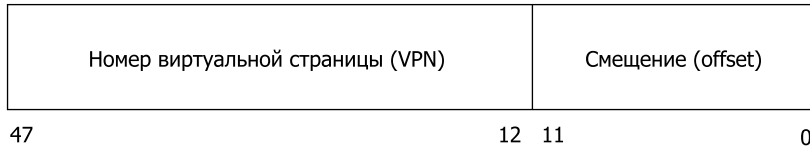


Рис. 3.16. Структура виртуального адреса

Устройство управления памятью MMU выполняет трансляцию адресов путем обращения в древовидную таблицу страниц в оперативной памяти PT (Page Table), которая может содержать указатели таблицы страниц PTP (Page Table Pointer) с физическим адресом таблицы страниц следующего уровня или строки таблицы страниц PTE (Page Table Entry) с физическим

адресом страницы и ее правами доступа (приложение 8). Когда трансляция установлена, MMU загружает строку таблицы страниц в свой кэш таблицы страниц TLB, с тем чтобы сократить использование таблицы страниц при последующих обращениях.

Устройство обращения в таблицу страниц TLU запускается только при отсутствии информации в TLB. Использован линейный вариант организации таблиц с загрузкой их в виртуальную память. Причем все уровни таблицы загружены в ту же виртуальную память задачи, которую они сами описывают. То есть трансляция виртуальных адресов таблицы выполняется в точности по тому же правилу, которое применяется для любого другого адреса задачи. Адресация к элементам таблицы осуществляется через указатель, индексируемый виртуальным номером страницы. Формат строки таблицы (PTE) для всех уровней — 64 разряда. Принципы трансляции адреса рассмотрены в приложении 8.

Особенностью архитектуры «Эльбрус» является то, что для эмуляции работы с виртуальными адресами Intel используется структура таблиц Pentium Pro. При этом поддержаны размеры страниц 4 Кбайт и 4 Мбайт, а также работа с 32-битовым физическим адресом.

Кэш таблицы страниц DTLB состоит из двух полностью идентичных устройств, каждое из которых обслуживает запросы своего кластера. Одно устройство имеет объем 512 строк таблицы страниц, частично ассоциативную организацию (4 столбца по 128 строк) и 2 порта обращения. Обращение к L1$ и к DTLB по четырем синхронным каналам осуществляется параллельно. Кроме двух портов по считыванию, DTLB имеет порт по записи, причем запись элемента в устройства DTLB обоих кластеров выполняется одновременно.

Распределение столбцов DTLB для размещения страниц обеих размеров задается установкой специальных битов в регистре управления MMU. Алгоритм замещения элементов в строке DTLB приведен в приложении 9.

Устройство обращения к массивам

Значительное время доступа во внешнюю память является одним из основных препятствий для обеспечения непрерывной работы и полной загрузки вычислительных устройств микропроцессора. Кэширование данных во многом решает эту проблему. Тем не менее оно имеет недостатки, которые не позволяют преодолевать блокировки по ожиданию данных из памяти во многих современных приложениях. К ним относятся ограниченный размер кэша и сами его алгоритмы, которые сохраняют только часто используемые данные.

Для оптимизации работы с подсистемой памяти используют методы предварительной подкачки данных. Они позволяют прогнозировать обращения в память и производить подкачку необходимых данных в кэш или другое специальное устройство за некоторое время до их использования.

Существующие методы подразделяются на программные и аппаратные. В первых применяются специальные инструкции предварительной подкачки, которые планируются с использованием информации о программе, сформированной на этапе компиляции. Они обрабатываются аналогично обычным обращениям в память, с тем исключением, что полученные данные оседают в кэше, а не передаются в регистровый файл микропроцессора.

В отличие от них аппаратные методы работают во время исполнения программы, используя для прогнозирования и предварительной подкачки динамическую информацию об обращениях в память. Они предполагают наличие в составе микропроцессора дополнительных модулей, например модуля предсказания обращений, опережающего счетчик команд, используемого в алгоритме подкачки, на один блок (one-block look ahead). К тому же аппаратные методы не требуют специальных инструкций подкачки и работают асинхронно, то есть могут функционировать даже во время блокировок микропроцессора. В то же время они дороги, недостаточно гибки в реализации и могут упустить необходимость подкачки, поскольку действуют на основе истории обращения в память.

Достоинствами обоих методов обладают комбинированные программно-аппаратные методы, особенность которых состоит в том, что при наличии в микропроцессоре дополнительных специальных модулей необходимость предварительной подкачки определяется компилятором, который вставляет в код специальные инструкции [24], [25]. Микропроцессор «Эльбрус» является одним из немногих микропроцессоров, в котором реализован именно такой метод. Для реализации его аппаратной части в архитектуру включено специальное устройство обращения к массивам (Array Access Unit, AAU). Кроме того, разработаны специальные инструкции микропроцессора: инициализация AAU, запуск (останов) программы предварительной подкачки, асинхронные инструкции предварительной подкачки (FAPB) и синхронные инструкции пересылки данных из буфера подкачки массивов (Array Prefetch Buffer, APB) в регистровый файл (MOVA).

Решение о введении специализированного устройства предварительной подкачки массивов обусловлено тем, что элементы массивов в циклах обрабатываются строго последовательно и редко используются более

одного раза, поэтому предварительное считывание в буфер значительно эффективнее, чем помещение элементов массивов в кэш, которому может сопутствовать выталкивание действительно полезных данных. При использовании AAU элементы массивов асинхронно и параллельно с исполнением основной программы считываются из оперативной памяти или кэша L2$ в буфер предварительной подкачки элементов массивов. Предварительно загруженные данные хранятся в этом буфере и затем перемещаются в регистровый файл общего назначения RF специальными командами (MOVA).

Структура AAU представлена на рис. 3.17.

Устройство содержит:

- две секции буфера предварительной подкачки массивов (APB) по 2 Кбайт каждая;
- два блока буфера команд асинхронной программы (PIB) по 32 64-разрядных регистра в каждом;
- два канала асинхронной подкачки массивов (Prefetch);
- четыре канала синхронной пересылки элементов массивов (Move) из APB в РгФ;
- два канала синхронного обращения к массивам (Store/Load).

Кроме того, в состав AAU входит регистровая память (8 регистров инкрементов, 16 регистров начальных индексов и 32 регистра дескрипторов массивов).

Буфер APB организован как циклический FIFO-буфер и представляет собой память объемом 4 Кбайт, разделенную на две секции по 2 Кбайт. Каждая секция имеет 2 порта шириной по 8 байт (2 двойных слова) для считывания, адресуемых из каналов синхронной пересылки элементов массивов (AAU_MOVE-каналов), и 1 порт шириной 4 двойных слова по записи, адресуемый из канала асинхронной подкачки массивов (AAU_Prefetch-канала). Каждый порт адресуется независимо, что позволяет одновременно записывать в секцию памяти и читать из нее по двум портам.

Слово порта записи буфера APB коммутируется на 4 банка памяти, каждый из которых содержит 64 строки размером в двойное слово (8 байт вместе с тегами), причем память адресуется с точностью до строки.

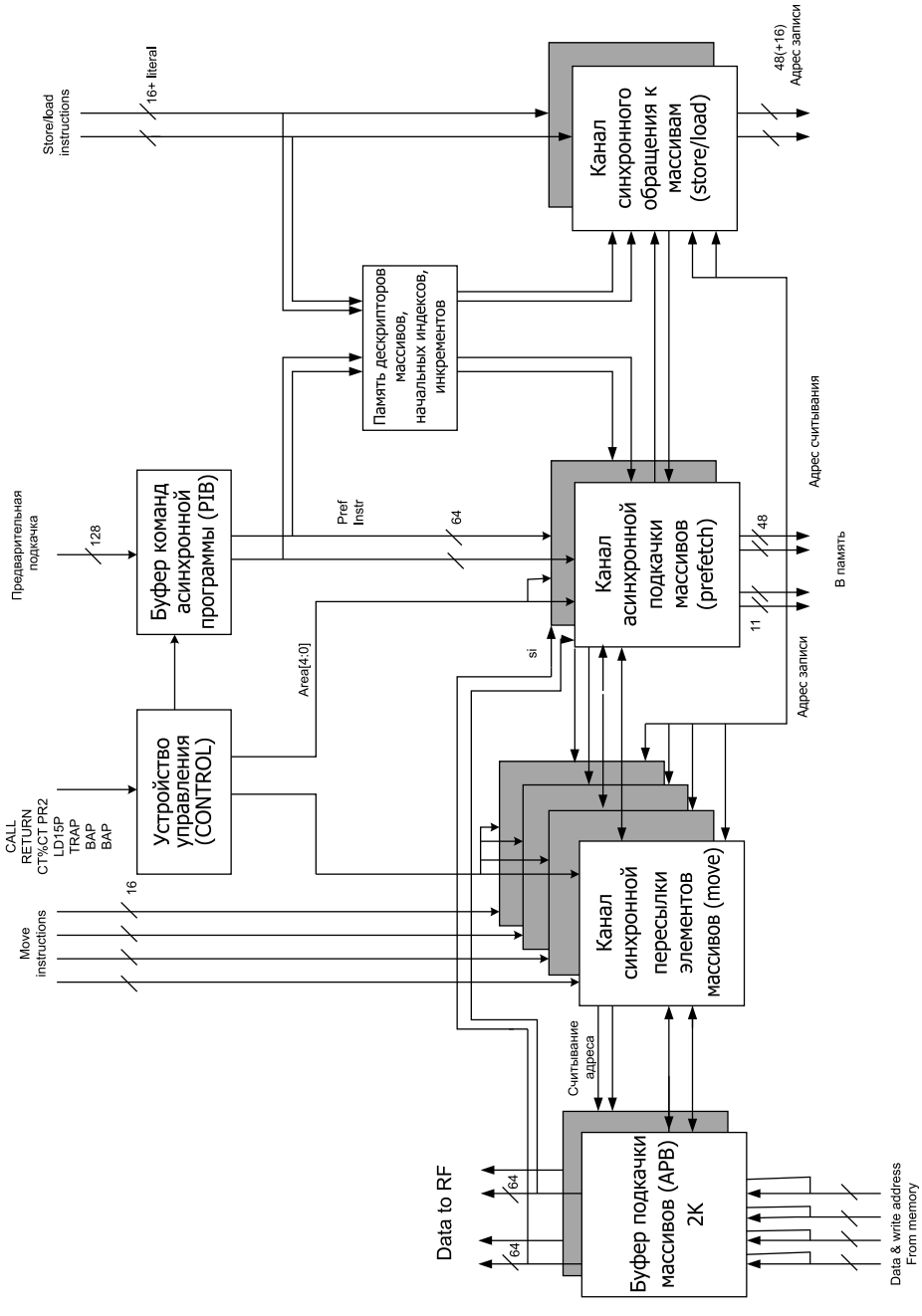


Рис. 3.17. Структурная схема устройства обращения к массивам

Пересылка массивов из памяти в буфер APB, асинхронная по отношению к выполнению основной программы, выполняется двумя Prefetch-каналами, причем каждый из них загружает свою секцию буфера. С точки зрения команд Prefetch-каналов секции APB разбиты на независимые области, количество которых соответствует числу массивов, подкачиваемых асинхронной программой и считываемых синхронной программой (одна команда определяет и наполняет свою область). Одна секция APB может содержать до 32 областей, минимальный размер которых — 8 двойных слов (строк одного банка). Каждая из них представляет непрерывный фрагмент памяти, включая разряды тегов. В командах асинхронной программы задаются начальные адреса областей, их размер и формат подкачиваемых элементов массива. Кроме адресов считывания данных из памяти асинхронные Prefetch-каналы генерируют адреса назначения, используя память текущих индексов и память указателей записи в APB.

Prefetch-каналы работают под управлением программы, которая загружается и хранится в буфере PIB. Он состоит из двух секций, каждая из которых содержит до 32 команд для обслуживания одного канала Prefetch.

Чтение данных из APB и пересылку их в RF осуществляют четыре синхронных MOVE-канала (по 2 канала для каждой секции APB) с помощью операций MOVA из основного программного потока (синхронной программы).

Два синхронных STORE/LOAD-канала генерируют адреса для синхронной записи данных из RF в память. При этом используются дескрипторы массивов, хранящиеся в устройстве AAU. Эти же каналы применяются и для обращения в память для считывания из синхронной программы. Упрощенная схема функционирования устройства обращения к массивам (AAU) представлена на рис. 3.18.

Дескрипторы массивов, начальные индексы, инкременты и программа предварительной подкачки массивов загружаются в устройство AAU перед выполнением циклического фрагмента. Данные квантами в 1, 2 или 4 двойных слова заблаговременно вызываются из памяти в буфер предварительной подкачки APB и затем поэлементно пересылаются в регистровый файл (RF) для использования в вычислениях. Результаты вычислений отсылаются обратно в заданные массивы в память. Предварительная подкачка данных в буфер APB производится отдельной программой, которая выполняется в устройстве AAU асинхронно по отношению к основной синхронной программе. Пересылка данных в регистровый файл (RF) и запись в память результатов вычислений производятся основной синхронной программой.

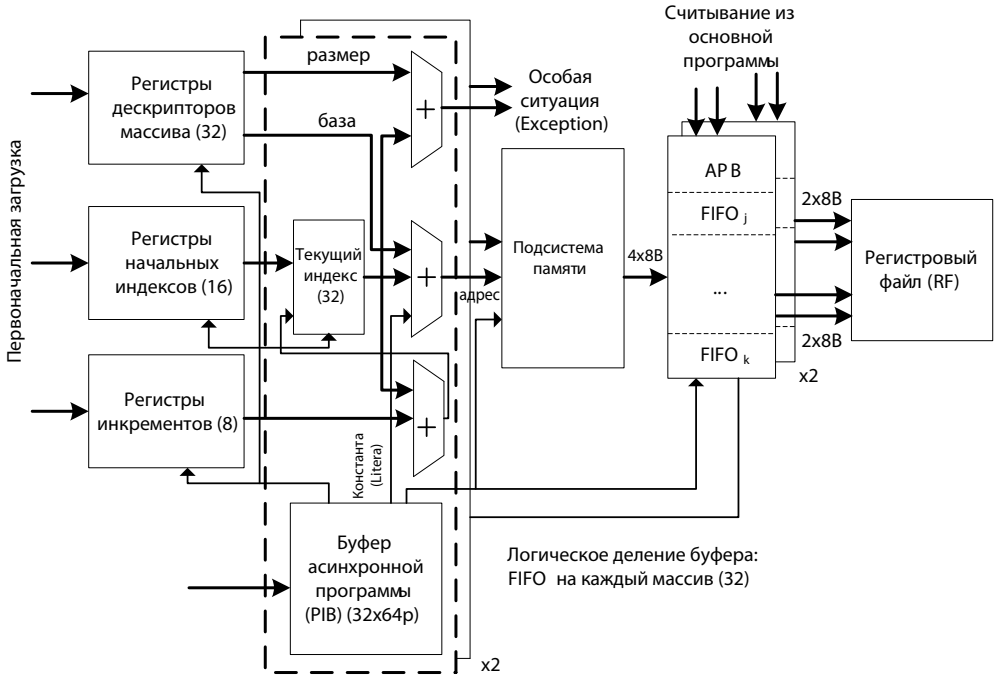


Рис. 3.18. Схема функционирования устройства обращения к массивам (AAU)

Программа предварительной подкачки загружается как отдельная ветвь общей программы в специальный буфер команд подкачки PIB, используя регистр подготовки CTPR2 устройства управления CU, и далее выполняется циклически асинхронным образом по отношению к основной (синхронной) программе. Асинхронная программа содержит по одной команде предварительной подкачки FAPB для каждого вызываемого массива (в общей сложности могут быть заданы до 64 команд), а в конце — команду перехода на начало асинхронной программы. В команде FAPB заданы начальный индекс и шаг продвижения по массиву подкачки в памяти, а также база и размер области буфера APB.

Пересылка элементов массивов из буфера APB в регистровый файл RF выполняется операциями MOVA из синхронной программы. Может быть задано до четырех операций MOVA одновременно.

Запись результатов вычислений также выполняется операциями из синхронной программы. При этом используются дескрипторы массивов и инкременты из той же памяти, что и при выполнении предварительной подкачки массивов, но отдельный набор начальных индексов, который

также загружается перед исполнением циклического фрагмента программы. Операции чтения из синхронной программы, подобно операциям записи, используют предварительно загруженные дескрипторы и индексы.

3.3. Технические характеристики микропроцессора «Эльбрус»

Приведенные в предыдущих разделах параметры устройств микропроцессора и некоторые из его общих показателей сведены в табл. 3.7.

Таблица 3.7. Технические характеристики микропроцессора «Эльбрус»

Параметр	**Значение**
Технологический процесс	КМОП 0,13 мкм, 8 слоев металла
Рабочая тактовая частота, МГц	300
Производительность: — 64 разряда, GIPS/GFLOPS; — 32 разряда, GIPS/GFLOPS; — 16–8 разрядов, GIPS	 6,67/2,4 9,5/4,8 12,2–22,6
Разрядность данных, бит: — целые; — вещественные	 8, 16, 32, 64 32, 64, 80
Кэш-память команд первого уровня, Кбайт	64
Кэш-память данных первого уровня, Кбайт	64
Кэш-память второго уровня, Кбайт	256
Кэш-таблица страниц, число входов	512
Пропускная способность шин связи с кэш-памятью, Гбайт/с	9,6
Пропускная способность шин связи с оперативной памятью, Гбайт/с	4,8
Напряжение питания — ядро/периферия, В	1,05/3,3
Рассеиваемая мощность, Вт	6,0
Размеры кристалла, мм	15,0×12,6
Количество транзисторов, млн шт.	75,8
Тип корпуса/количество выводов	HFCBGA/900
Размеры корпуса, мм	31,0×31,0×2,4

3.4. Вычислительный комплекс «Эльбрус-3М1»

3.4.1. Назначение

ВК «Эльбрус-3М1» стал машиной, в составе которой были отработаны принципы формирования общей структуры универсального высокопроизводительного вычислительного комплекса, построенного на базе микропроцессора «Эльбрус» [7]. Фактически в процессе его государственных испытаний, которые проводились совместно с испытаниями микропроцессора, прошла проверку продуктивность решений, описанных в предыдущих разделах. В этом смысле особое значение имели сравнительные оценки (рис. 3.19), показавшие, что существенно большие возможности распараллеливания вычислений, заложенные в аппаратуру и поддержанные оптимизирующим компилятором и операционной системой ВК, позволили получить производительность, сравнимую с показателями процессоров Pentium, работающих на более высоких тактовых частотах, или заметно превосходящую их.

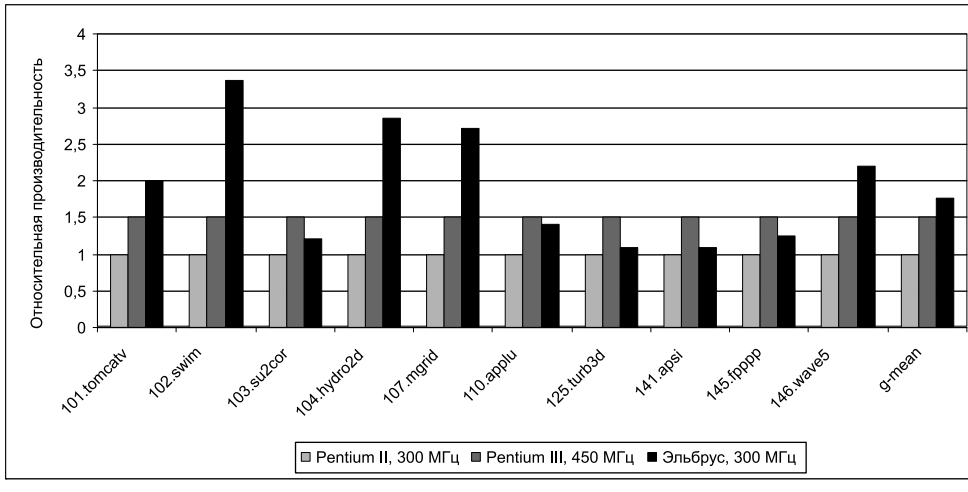


Рис. 3.19. Сравнительная производительность ВК «Эльбрус-3М1» и Pentium II и III на пакете задач SPECfp95

Наряду с эффективным обеспечением совместимости с широко распространенной платформой Intel x86 и возможностью защищенного исполнения программ эти показатели вычислительного комплекса «Эльбрус-3М1» позволили рекомендовать его к использованию для решения сложных вычислительных задач с большим объемом данных в крупномасштабных информационно-вычислительных и управляющих системах.

3.4.2. Структура и технические характеристики

Согласно общей классификации, вычислительный комплекс «Эльбрус-3М1» имеет структуру, типичную для современного многопроцессорного компьютера (рис. 3.20).

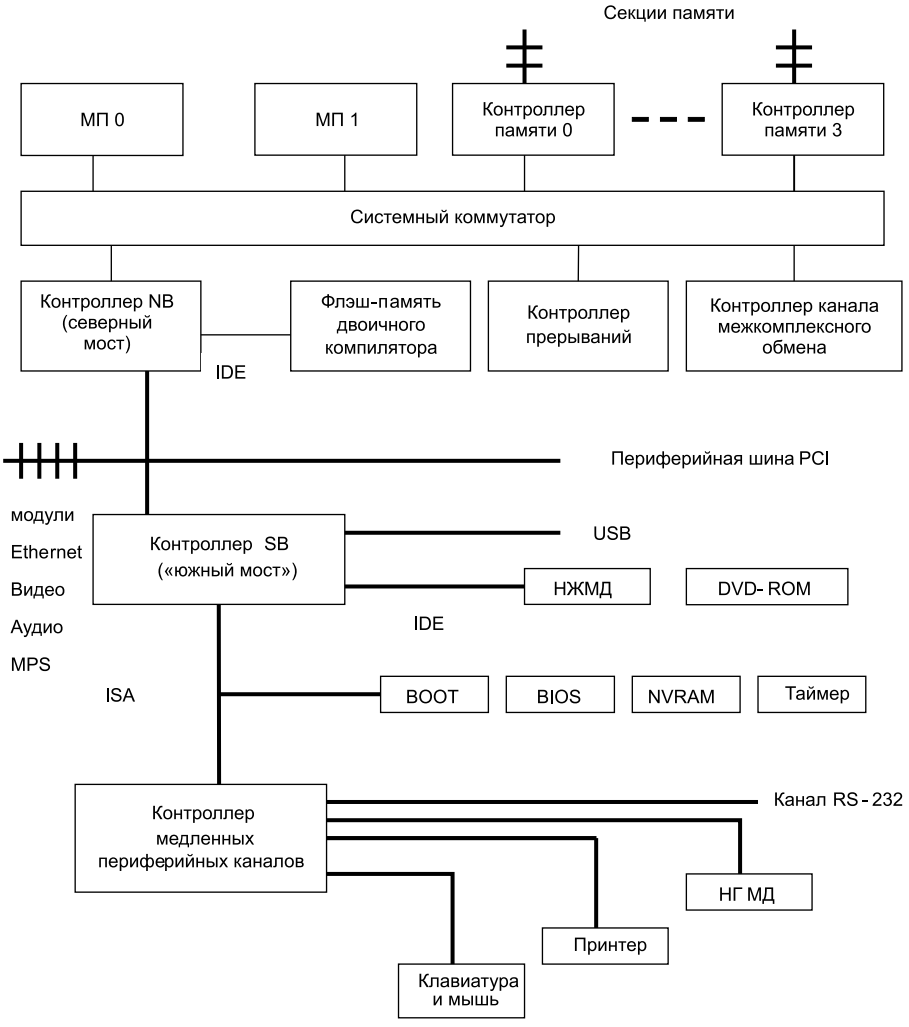


Рис. 3.20. Структура вычислительного комплекса «Эльбрус-3М1»

Его процессорную часть образуют два микропроцессора «Эльбрус» (МП 0 и МП 1), соединенные с оперативной памятью через системный коммутатор. Для обеспечения высокой пропускной способности и большого объема

ОП разработан системный коммутатор с 4 каналами обращения в память. Соответственно, память содержит четыре секции, в каждой из которых размещены по два модуля объемом до 4 Гбайт каждый, соединенных с системным коммутатором через контроллеры памяти 0–3. При обмене с памятью используется код коррекции ошибок (error correcting code, ECC), рассчитанный на исправление одиночных и обнаружение двойных ошибок.

Абоненты системного коммутатора (микропроцессоры, модули памяти и контроллер NB) подключены с использованием метода передачи данных DDR (Double Data Rate), обеспечивающего две посылки данных в течение одного такта синхронизации. В системном коммутаторе реализованы схема поддержки когерентного состояния памяти и внутренних кэшей микропроцессоров, а также схема синхронизации при обращениях к семафорам в двухпроцессорной системе.

Дуплексный высокоскоростной канал межмашинного обмена, реализованный с использованием LVDS-интерфейса, подключен к оперативной памяти непосредственно через системный коммутатор, минуя подсистему ввода/вывода, что позволяет вести интенсивный обмен между комплексами и исключает негативное влияние на обмен с внешними устройствами. Также в системный коммутатор перенесены из процессоров локальные контроллеры прерываний.

Базовой магистралью подсистемы ввода/вывода комплекса является периферийная шина PCI (32 разряда, 33 МГц), соединенная с процессорной частью и памятью через контроллер NB («северный мост»). Четыре слота (разъема) на этой шине предназначены для непосредственного подключения мезонинных ячеек контроллера сети Ethernet 10/100, видеокарты, аудиокарты, контроллера MPS (канала сопряжения с устройством внешних абонентов ВК «Эльбрус-90микро»). Остальная периферия связана с шиной PCI через контроллер SB («южный мост»).

«Южный мост» содержит контроллеры шин USB, IDE и ISA. К шине USB могут подключаться внешние периферийные устройства, например переносная флеш-память, с шиной IDE связаны дисковые накопители на жестких магнитных дисках (НЖМД) и DVD-ROM. Шина ISA соединяет «южный мост» с контроллером медленных периферийных каналов, перезаписываемой памятью программы начального старта BOOT, энергонезависимой памятью системных параметров NVRAM и системным таймером. Контроллер медленных периферийных каналов содержит контроллеры двух последовательных каналов RS-232, контроллер параллельного канала принтера IEEE 1284, контроллер последовательного канала Floppy для подключения накопителя

на гибких магнитных дисках (НГМД), контроллеры двух последовательных каналов PS/2 для подключения клавиатуры и мыши.

Для обеспечения работы битового компилятора кодов платформы Intel x86 в состав вычислительного комплекса введены перезаписываемая память начального старта BIOS и флеш-память двоичного компилятора для хранения оттранслированных кодов. Показатели вычислительного комплекса сведены в табл. 3.8.

Таблица 3.8. Технические характеристики ВК «Эльбрус-3М1»

Параметр	**Значение**
Количество процессоров	2
Объем оперативной памяти, Гбайт	16
Тактовая частота, МГц	300
Пиковая пропускная способность каналов обмена с памятью, Гбайт/с	9,6
Объем дисковой памяти, Гбайт	160
Канал межкомплексного обмена	Дуплекс, 400 Мбайт/с
Периферийные шины	PCI, SBus
Каналы обмена, количество типов	Более 10
Пиковая производительность: — полноформатные вычисления 64/32 разряда, GIPS; — полноформатные вычисления с плавающей запятой 64/32 разряда, GFLOPS; — малоформатные вычисления 16/8 разрядов, GIPS	 13,3/19,1 4,8/9,6 24,4/45,2
Количество каналов приема внешних прерываний	Не менее 20
Время реакции на внешнее прерывание, мкс	Не более 15
Разрядность представления чисел: — с плавающей запятой; — целых	 32, 64, 80 32, 64
Средняя наработка на отказ, ч	Не менее 10 000
Первичная сеть, В/Гц	220±10 %/50
Потребляемая мощность (в зависимости от комплектации), Вт	120...250

Вычислительный комплекс «Эльбрус-3М1» поставляется в конструктивах «Евромеханика» и «серверный корпус» (Rockmount 19) (рис. 3.21).

а

б

Рис. 3.21. Варианты конструктивного исполнения ВК «Эльбрус-3М1»:
а — «Евромеханика», настольное исполнение;
б — Rockmount 19, установка в серверной стойке

3.5. Система на кристалле «Эльбрус-S»

Разработка микросхемы «Эльбрус-S», выполненная непосредственно после выпуска микропроцессора «Эльбрус» и на его базе, преследовала следующие цели:

- создать систему на кристалле, существенно расширяющую состав и параметры устройств базовой микросхемы «Эльбрус»;

- повысить производительность путем перехода на технологические нормы 90 нм и тактовую частоту 500 МГц;
- организовать обмен с оперативной памятью и межпроцессорный обмен на основе распределенного коммутатора с поддержкой когерентности данных в общей распределенной оперативной памяти и кэш-памяти процессоров.

Структурная схема системы на кристалле (рис. 3.22) включает:

- ядро микропроцессора «Эльбрус»;
- системный контроллер SIC, содержащий:
  - два контроллера оперативной памяти MC 0 и MC 1;
  - три контроллера каналов межпроцессорного обмена IPCC 0, IPCC 1, IPCC 2;
  - контроллер канала ввода/вывода IOCC.

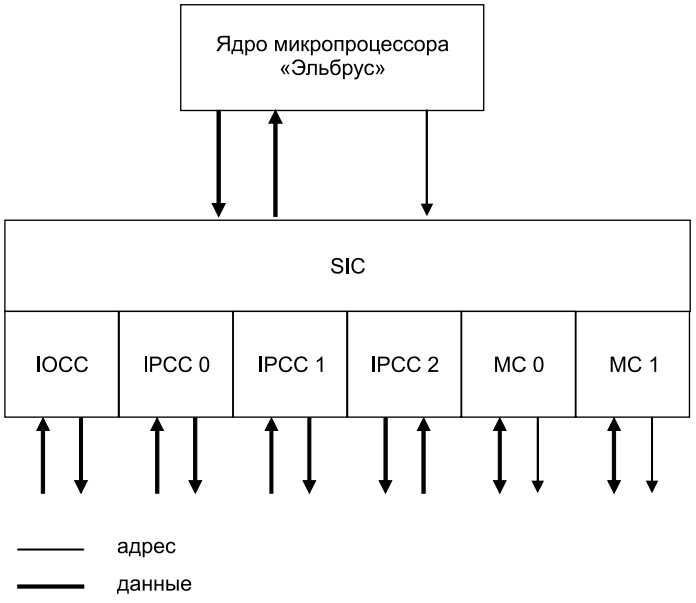


Рис. 3.22. Структурная схема системы на кристалле «Эльбрус-S»

Ядро микропроцессора «Эльбрус» является высокопроизводительным универсальным микропроцессором с архитектурой широкого командного слова. Его тактовая частота повышена до 500 МГц.

Контроллер системного обмена SIC обеспечивает:

- доступ через контроллеры MC 0 и MC 1 к локальной памяти системы на кристалле от ядра микропроцессора, контроллеров связи с тремя внешними системами на кристалле и контроллера связи с каналом ввода/вывода;
- доступ ядра микропроцессора к памяти других систем на кристалле «Эльбрус-S» через каналы межпроцессорного обмена IPCC;
- доступ ядра микропроцессора к подсистеме ввода/вывода через контроллер канала ввода/вывода IOCC.

Каждый из контроллеров MC 0 и MC 1 обеспечивает обмен с соответствующей ему секцией локальной памяти объемом 4 Гбайт, то есть суммарный доступный объем локальной памяти составляет 8 Гбайт. Контроллер реализует полудуплексный канал шириной 8 байт с частотой обмена 250 МГц и передачей данных в две посылки за один такт. Соответственно, максимальный темп обмена данными с памятью по двум каналам составляет $2 \times 8 \times 2 \times 250 = 8$ Гбайт/с, или 2 двойных слова за один такт ядра микропроцессора «Эльбрус».

С помощью контроллеров IPCC 0, 1, 2 может быть выполнено обращение в локальные памяти других систем на кристалле и к другим абонентам через их внутренние системные коммутаторы. Доступный объем удаленной памяти составляет $3 \times 8 = 24$ Гбайт. Каждый контроллер IPCC реализует канал полнодуплексного обмена шириной 2 байта с частотой обмена 500 МГц и передачей данных в две посылки за один такт. Таким образом, один канал обмена обеспечивает пропускную способность 4 Гбайт/с (по 2 Гбайт/с на прием и выдачу данных соответственно), а максимальный суммарный темп обмена данными по всем трем каналам составляет 12 Гбайт/с. По каналам межпроцессорного обмена передаются в числе других операции поддержки когерентного состояния общей распределенной памяти.

Контроллер канала IOCC предназначен для связи с подсистемой ввода/вывода. Канал позволяет ядру микропроцессора «Эльбрус» обмениваться данными непосредственно с внешними устройствами, а также реализует режим прямого доступа внешних устройств к памяти системы на кристалле (режим DMA). Он обеспечивает полнодуплексный обмен шириной 1 байт с частотой 500 МГц и передачей данных в две посылки за один такт. Суммарная пропускная способность канала 2 Гбайт/с (1 Гбайт — прием, 1 Гбайт — выдача). В итоге работы над проектом была выпущена микросхема с характеристиками, приведенными в табл. 3.9.

Таблица 3.9. Технические характеристики микросхемы «Эльбрус-S»

Параметр	Значение
Рабочая тактовая частота, МГц	500
Разрядность данных: — целые; — вещественные	 8, 16, 32, 64 32, 64, 80
Производительность: — 64 разряда, GIPS/GFLOPS; — 32 разряда, GIPS/GFLOPS; — 16–8 разрядов, GIPS	 10,0/4,0 15,9/8,0 21,5–39,5
Кэш-память данных 1-го уровня, Кбайт	64
Кэш-память команд 1-го уровня, Кбайт	64
Кэш-память 2-го уровня (универсальная), Мбайт	2
Кэш-память страниц данных, входов	1024
Кэш-память страниц команд, входов	64
Пропускная способность кэш-памяти, Гбайт/с	16
Пропускная способность каналов оперативной памяти, Гбайт/с	8
Пропускная способность каналов межпроцессорного обмена, Гбайт/с	12
Пропускная способность канала ввода/вывода, Гбайт/с	2
Технологический процесс КМОП, нм	90
Количество транзисторов, млн шт.	218
Напряжение питания, В	1,1; 1,8; 2,5
Потребляемая мощность, Вт	13
Тип корпуса/количество выводов, шт.	HFCBGA/1156
Размеры корпуса, мм	35×35×3,2

3.6. Модуль MB3S/C

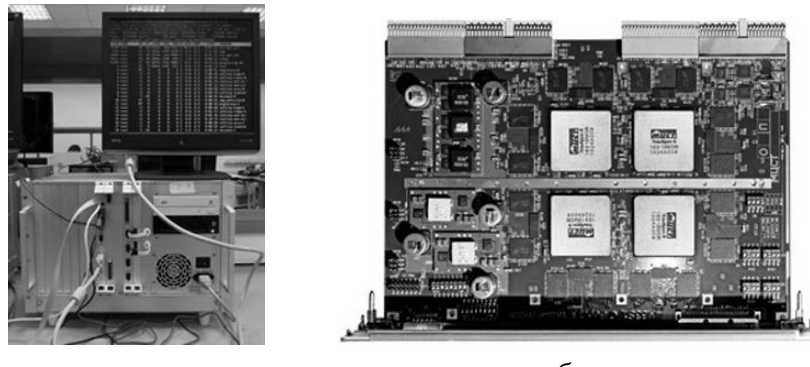
Модуль MB3S/C, спроектированный на базе системы на кристалле «Эльбрус-S», представляет собой универсальную 4-процессорную высокопроизводительную вычислительную систему, которая обеспечивает:

- выполнение многопроцессорных, многопользовательских и многопрограммных вычислений в различных режимах, включая режим жесткого реального времени;

- возможность создания больших программных комплексов с аппаратно-программной защитой программ и данных;
- полную двоичную совместимость с архитектурной платформой Intel IA-32 (x86), что позволяет использовать его вместо импортной техники;
- высокую скорость исполнения программ.

Входящий в состав модуля набор специальных и стандартных интерфейсов, поддержанных средствами операционной системы, позволяет создавать на их основе вычислительные системы различной производительности и функциональности. Возможность резервирования и использование технологии защищенного исполнения программ позволяют увеличить надежность работы многомашинных комплексов и их программного обеспечения при эксплуатации в жестких условиях.

Модуль MB3S/C включает процессорный модуль MB3S1/C и системный модуль MB3S2/C в конструктиве «Евромеханика-6U» Compact PCI, устанавливаемые в объединительную панель крейта Compact PCI. Модуль MB3S1/C представляет собой соответствующую одноплатную конфигурацию, состоящую из четырех микросхем «Эльбрус-S», объединяемых высокоскоростными каналами межпроцессорного обмена, подключенных к ним секций общей оперативной памяти и контроллеров ввода/вывода. Пропускные способности этих связей соответствуют параметрам соответствующих каналов микросхемы «Эльбрус-S». Внешний вид модулей на базе микросхемы «Эльбрус-S» представлен на рис. 3.23, основные показатели модуля MB3S/C — в табл. 3.10.



а *б*

Рис. 3.23. Модули на базе микросхемы «Эльбрус-S»: *а* — модуль MB3S/C; *б* — процессорный модуль MB3S1/C

Таблица 3.10. Технические характеристики модуля MB3S/C

Параметр	Значение
Производительность, GIPS/GFLOPS: — 64 разряда; — 32 разряда	 34,3/15,4 53,0/29,5
Объем оперативной памяти, Гбайт	8
Суммарная пропускная способность памяти, Гбайт/с	32
Набор интерфейсов: PCI Express 8x, PCI 64/66, SATA, IDE, Ethernet 10/100/1000, USB 2.0, IEEE 1284, Video, AC-97, RS-232, SPI, I2C, GPIO	

3.7. Двухъядерная гетерогенная система на кристалле «Эльбрус-2С+»

Разработки, рассмотренные в предыдущих разделах, соответствуют определяющим тенденциям в проектировании современных микропроцессорных систем — повышению производительности за счет добавления процессорных ядер и интеграции процессорной части, устройств доступа к памяти и контроллеров ввода/вывода в составе системы на кристалле, которая позволила существенно повысить пропускную способность и уменьшить задержки при межмодульном обмене. Наряду с этим все более очевидные преимущества демонстрирует создание гетерогенных процессоров, которые, так же как и универсальные, имеют специализированные ядра, предназначенные для быстрой обработки данных в определенных применениях [26].

Именно эти ресурсы повышения производительности использовались в проекте, предполагающем применение универсальных высокопроизводительных микропроцессоров с архитектурой «Эльбрус» в крупных радиолокационных системах. При создании микросхемы «Эльбрус-2С+» впервые была реализована двухъядерная процессорная часть на базе архитектуры «Эльбрус», введен кластер обработки цифровой сигнальной информации, состоящий из четырех DSP-процессоров с архитектурой ElCore (разработка ФГУП НПЦ «Элвис»), а обе процессорные части объединены в составе системы на кристалле с двухканальным контроллером оперативной памяти DDR2-800, контроллером ввода/вывода и контроллером межпроцессорных линков.

В интересах оперативной реализации проекта в качестве его IP-блоков была использована часть оборудования микросхемы «Эльбрус-S»: процессорные ядра, включая незначительно измененный L2-кэш, и системный контроллер SIC с контроллером памяти и контроллером межпроцессорных линков

и контроллером ввода/вывода. Все изменения, связанные с увеличением частоты оперативной памяти, выполнены в контроллере памяти и не коснулись остальной системы. DSP-кластер также выступал как отдельный IP-блок, который изготавливал ФГУП НПЦ «Элвис».

Общая схема системы на кристалле «Эльбрус-2С+» приведена на рис. 3.24. Объединение процессорных ядер выполнено на основе контроллера межъядерных взаимодействий (Core Integration Controller (CIC)). В его состав введен контроллер MRC (MAU Request Controller), который осуществляет круговой арбитраж, поочередно выдавая в системный коммутатор SC запросы от каждого из ядер.

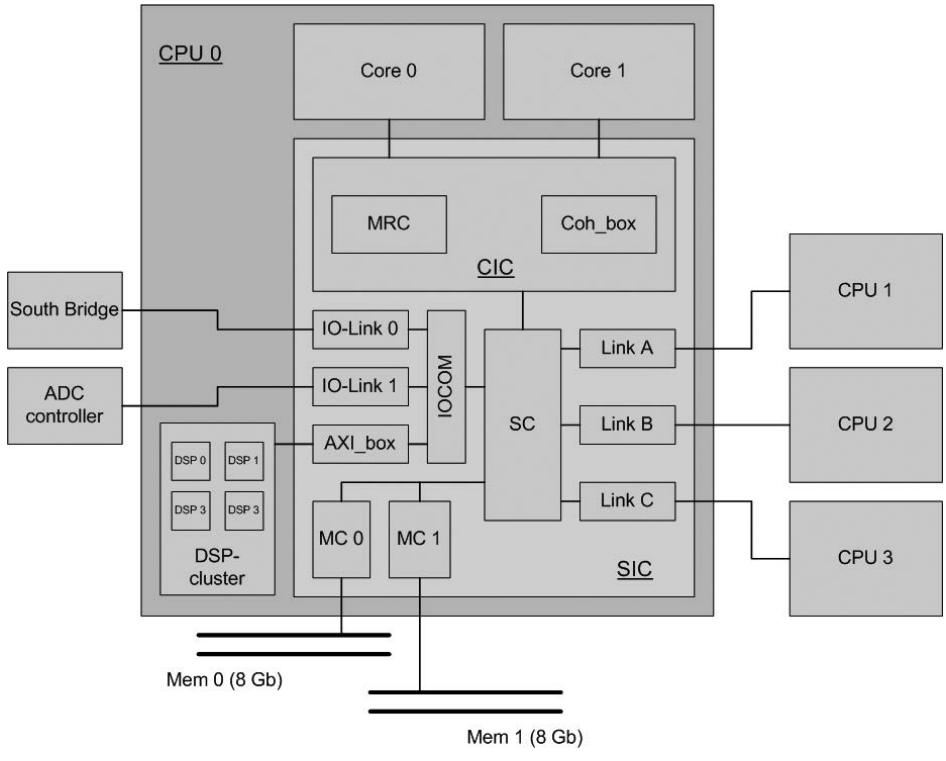


Рис. 3.24. Структурная схема системы на кристалле «Эльбрус-2С+»

Другой проблемой, решенной при проектировании контроллера CIC, является разработка в его составе контроллера Coh_box, который анализирует когерентные запросы от SC, выборочно передавая их нужным ядрам, и принимает когерентные ответы от ядер, при необходимости формируя обобщенный когерентный ответ в формате одноядерной системы.

Интегрированный в систему на кристалле четырехъядерный DSP-кластер работает под управлением универсальных ядер «Эльбрус», которым доступны его регистры и память. Во внутреннюю память кластера записывается программа для DSP-ядер, запуск и управление которыми осуществляется посредством операций универсальных ядер с регистрами кластера, организованных аналогично обменам с внешними устройствами. В свою очередь, DSP-кластеру доступна не только собственная внутренняя память, но и память всей системы.

Взаимодействие универсальных ядер с ядрами DSP осуществляется через разработанный в проекте интерфейсный контроллер AXI-box. Он связан с кластером через интерфейс, основанный на протоколе AMBA AXI 3.0 фирмы ARM [26], который выбран из-за его универсальности и возможности построения на его основе быстродействующего внутреннего интерфейса. Данные передаются AXI-контроллером по четырем раздельным каналам (DMA-чтение, DMA-запись, I/O-чтение, I/O-запись), каждый из которых имеет пропускную способность 4 Гбайт/с и работает на частоте ядер «Эльбрус» и DSP (500 МГц).

Оперативная память системы на кристалле используется, помимо прочего, в качестве буфера данных, пересылаемых между аналогово-цифровыми преобразователями и DSP-кластером. С учетом этого обстоятельства, исходя из анализа соотношений пропускной способности соответствующих интерфейсов, в состав микросхемы помимо основного канала ввода/вывода (IO-Link0), подключенного через «южный мост» (South Bridge), введен дополнительный канал (IO-Link1) для ввода цифровой сигнальной информации, подключенный через ADC-контроллер.

Еще одной важной задачей, которая была решена при проектировании микросхемы «Эльбрус-2С+», стало увеличение частоты оперативной памяти для повышения ее пропускной способности. Из-за сложности конвейеризации и физического дизайна контроллер памяти MC был разделен на два частотных домена (домен ядра контроллера и домен интерфейса с памятью) с введением новой схемы передачи данных с некратным соотношением частот между доменами [27]. Она обеспечивает достаточно гибкую настройку рабочих частот, благодаря чему контроллер памяти микросхемы «Эльбрус-2С+» при фиксированной частоте процессора 500 МГц поддерживает несколько режимов работы с оперативной памятью: DDR2-800, DDR2-666, DDR2-600 и DDR2-500.

В результате рассмотренных решений была разработана уникальная в отечественной практике гетерогенная система на кристалле с показателями, приведенными в табл. 3.11.

Таблица 3.11. Технические характеристики микросхемы «Эльбрус-2С+»

Параметр	**Значение**
Технологический процесс	90 нм
Тактовая частота, МГц	500
Число ядер архитектуры «Эльбрус»	2
Число ядер DSP (Elcore-09)	4
Производительность микросхемы (ядра CPU + ядра DSP): — 64 разряда, GFLOPS; — 32 разряда, GIPS/GFLOPS	 8 44/28
Кэш-память команд (на ядро), Кбайт	64
Кэш-память данных (на ядро), Кбайт	64
Кэш-память второго уровня (на ядро), Мбайт	1
Встроенная память DSP (на ядро DSP), Кбайт	128
Пропускная способность шины связи с кэш-памятью, Гбайт/с	16
Пропускная способность шин связи с оперативной памятью (два канала), Гбайт/с	12,8
Количество каналов межпроцессорного обмена	3
Пропускная способность канала межпроцессорного обмена, Гбайт/с	4
Количество каналов ввода/вывода	2
Пропускная способность канала ввода/вывода, Гбайт/с	2
Площадь кристалла, мм^2	289
Количество транзисторов, млн	368
Количество слоев металла	9
Тип корпуса/количество выводов	HFCBGA/1296
Размеры корпуса, мм	37,5×37,5
Напряжение питания, В	1,0/1,8/2,5
Рассеиваемая мощность, Вт	~25

Контрольные вопросы и задания

1. Каковы состав и структура широкой команды?
2. Дайте характеристику типов слогов в составе широкой команды.
3. Дайте характеристику спекулятивного режима выполнения команд.
4. В чем заключается назначение дескриптора объекта?

5. Чем определяются типы и форматы данных?
6. Что такое контекст программы?
7. Поясните сущность двоичной трансляции программы.
8. Дайте характеристику стандартных средств архитектуры, обеспечивающих совместимость.
9. Дайте характеристику специальных средств архитектуры, обеспечивающих совместимость и производительность.
10. Дайте характеристику стандартных средств архитектуры, повышающих эффективность системы двоичной трансляции.
11. Какие устройства включает микропроцессор?
12. Каковы особенности устройства управления микропроцессора?
13. Каковы особенности построения арифметико-логического устройства микропроцессора?
14. Для чего в структуру микропроцессора введено устройство логических предикатов?
15. Какие устройства памяти содержит микропроцессор?
16. Какие ступени выполнения команд содержит конвейер микропроцессора? В чем их смысл?
17. В каких устройствах микропроцессора происходит обработка программного кода?
18. С какой целью производится выравнивание различных структур при их размещении в памяти?
19. В чем заключается основная функция буфера команд?
20. По какому принципу построен кэш команд устройства буфера команд?
21. С какой целью ассоциативная и ответная части кэша команд выполнены в виде отдельных устройств?
22. Каков принцип построения ассоциативной и ответных частей кэша команд?
23. Что является ассоциативным признаком строки программного кода?

24. Как осуществляется поиск требуемой строки программного кода в кэше команд устройства БК (буфер команд)?
25. Каким образом происходит считывание строки программного кода из кэша команд и выдача ее в устройство управления?
26. Каковы особенности работы буфера команд в режиме виртуальной адресации?
27. Чем различаются принципы построения буфера страниц и кэша команд устройства БК?
28. Как осуществляется подкачка строк программного кода в кэш команд устройства БК?
29. Каким образом в устройстве БК выполняется подготовка перехода при выполнении команд ветвления?
30. Какие функции выполняет устройство управления?
31. Как определяются в устройстве управления действительные адреса операндов?
32. Как реализован в микропроцессоре динамический останов конвейера?
33. Каков принцип выполнения команд переходов?
34. Каковы особенности структуры регистрового файла, отличающие его от классических СОЗУ?
35. По каким причинам регистровый файл выполнен в виде двух отдельных устройств?
36. Как хранятся данные различных форматов и типов в регистровом файле?
37. Что представляют собой области регистрового файла?
38. Для какой цели предназначена и каким образом организуется вращающаяся область?
39. Как организована в микропроцессоре проверка готовности операндов для очередных простых операций широкой команды?
40. В чем заключается сущность метода ускорения доставки операндов и как он реализован в микропроцессоре?

41. Каков принцип управления шинами байпаса?
42. В чем заключается проблема зависимости по выходу и как она решена в микропроцессоре?
43. Какие типы стеков используются в микропроцессоре?
44. Какие части включает процедурный стек и как они организуются?
45. Когда и каким образом выполняется откачка данных из регистрового файла?
46. Когда возникает необходимость в подкачке данных в регистровый файл?
47. Какие действия выполняются в микропроцессоре при запуске новой процедуры?
48. Какие действия выполняются при возврате из завершившейся процедуры?
49. Какие исполнительные блоки включают устройства целочисленной арифметики?
50. Каков принцип выполнения комбинированных целочисленных операций?
51. Какие этапы включает конвейер целочисленных операций?
52. Каков состав устройства вещественной арифметики?
53. Какие типы данных обрабатываются в устройствах вещественной арифметики?
54. Чем различаются скалярные и упакованные значения?
55. Каким образом поступают операнды в устройства вещественной арифметики?
56. Как выполняются комбинированные вещественные операции?
57. Для каких целей в структуру устройства вещественной арифметики введены входные и выходные буферы?
58. Что представляют собой операнды для команд мультимедийной обработки?
59. Каков принцип выполнения операций мультимедийной обработки?

60. Перечислите состав системы памяти ВК «Эльбрус-3М1» и кратко охарактеризуйте назначение ее составных частей.
61. Назовите основные технические характеристики системы памяти ВК «Эльбрус-3М1».
62. Охарактеризуйте кэш данных первого уровня. Поясните, как происходит обращение к нему.
63. Поясните назначение и объясните принципы работы механизма старения кэша данных первого уровня.
64. Перечислите функции, выполняемые устройством управления памятью.
65. Поясните принципы преобразования виртуальных адресов в физические.
66. Охарактеризуйте кэш второго уровня. Поясните, как происходит обращение к нему.
67. Поясните назначение и организацию кэша таблицы страниц.
68. Поясните назначение и объясните принципы работы устройства обращения к массивам AAU.
69. Поясните назначение и объясните принципы работы контроллера памяти.
70. В чем заключается назначение вычислительного комплекса «Эльбрус-3М1»?
71. Назовите состав вычислительного комплекса «Эльбрус -3М1».
72. Характеристика структуры системы на кристалле «Эльбрус-S».
73. Каково назначение основных элементов системы на кристалле «Эльбрус-S»?
74. Каково назначение модуля MB3S/C?
75. Расскажите о структуре микросхемы «Эльбрус-2С+».
76. Каковы основные характеристики микросхемы «Эльбрус-2С+»?

ГЛАВА 4

Общее программное обеспечение вычислительных комплексов семейства «Эльбрус»

Базовым компонентом программного обеспечения рассмотренных компьютерных платформ является общесистемное, или общее, программное обеспечение (ОПО), организующее функционирование и поддерживающее возможности аппаратных средств, управляющее вычислительными процессами и ресурсами системы. ОПО для архитектур SPARC и «Эльбрус» собирают из единой базы исходных текстов. Операционные системы и оптимизирующие компиляторы в его составе обеспечивают реализацию основных факторов повышения производительности, свойственных обеим архитектурным линиям. Наряду с этим в составе архитектуры «Эльбрус» определяющее значение имеют система исполнения двоичных кодов архитектуры x86 с помощью динамической двоичной трансляции (для краткости двоичный компилятор) и система защищенного исполнения программ на языках C и C++.

4.1. Операционные системы

4.1.1. Номенклатура поставляемых ОС

В составе ОПО для вычислительных комплексов с архитектурой SPARC и «Эльбрус» создана, сопровождается и постоянно развивается операционная система ОС «Эльбрус» на базе ядра Linux 2.6.14, обеспечивающая многозадачный и многопользовательский режимы работы. Для нее потребовалось разработать особые механизмы управления процессами, виртуальной памятью, прерываниями, сигналами, синхронизацией, тегированными вычислениями. Чтобы обеспечить использование вычислительных средств серии «Эльбрус» в ряде стратегических систем, была проделана фундаментальная работа по преобразованию ОС Linux в операционную систему, поддерживающую режим работы в жестком реальном времени [28], для которой реализована стандартная библиотека управления потоками вычислений и синхронизацией libpthread и создана собственная библиотека elpthread [29].

Помимо этой операционной системы поставляются и поддерживаются ОС МСВС с ядром OSL_90 (на базе Linux 2.4.25) для ВК «Эльбрус-90микро» и ОС МСВС с ядром OSL_3M1 (на базе Linux 2.6.14) для ВК «Эльбрус-3М1». В составе серии ВК «Эльбрус-90микро» эксплуатируется также операционная система OS_E90 на базе Solaris 2.5.1.

В дистрибутив ОПО «Эльбрус» входит уже откомпилированное ядро ОС. Если возникает необходимость внести изменения в исходные тексты ОС, ввести в операционную систему дополнительный функционал, то ядро следует перекомпилировать с добавлением требуемых функций. Этот принцип действует и для формирования ядра ОС «Эльбрус» с поддержкой реального времени.

В ходе работы в реальном времени можно устанавливать различные режимы обработки внешних прерываний, планирования вычислений, обменов с дисковыми накопителями и некоторые другие. Возможности настройки режима работы в реальном времени приведены в приложении 10.

4.1.2. Средства поддержки интерфейса пользователя в составе ОС «Эльбрус»

Базовые средства включают следующие составляющие:

- средства поддержки интерфейса командной строки, обычно ассоциируемые с понятием «консоль», — обеспечивают оператору возможность работы с вычислительным комплексом в текстовом режиме с помощью набора команд и получения текстовых сообщений от операционной системы и запускаемых приложений;
- средства архивации — предназначены для объединения ряда файлов в единый архив или серию архивов, что обеспечивает удобство переноса (передачи через каналы связи) или хранения. В ходе архивации может использоваться сжатие данных;
- средства разработки программного обеспечения — необходимы для организации процесса разработки и поддержки программного обеспечения. Это ассемблеры, трансляторы, компиляторы, компоновщики (редакторы связей), сборщики, препроцессоры, отладчики, текстовые редакторы, библиотеки подпрограмм, средства управления версиями, средства документирования;
- средства планирования заданий — позволяют указать операционной системе, какие действия, в какое время и с какой периодичностью необходимо выполнить.

Помимо базовых в интерфейс пользователя введен ряд средств, поддерживающих создание функционального программного обеспечения. Наиболее часто используемые из них рассматриваются далее.

Средства поддержки графического пользовательского интерфейса содержат базовые компоненты графической системы Xorg, а также набор различных вспомогательных библиотек, в том числе GTK+ и Qt.

Средства работы с сетью включают базовые утилиты, сервисы и библиотеки для обеспечения работы в сетевом окружении:

- сервер гипертекстовой обработки данных apache;

- библиотеки анализа сетевых пакетов libpcap;
- клиенты доступа к гипертекстовой информации curl и lynx;
- клиент доступа к сообщениям mailx;
- клиент сетевой службы времени ntp;
- средства защищенного удаленного доступа openssh;
- библиотеки криптографии openssl;
- сервис назначения портов portmap;
- клиент удаленного терминала rdesktop;
- сервис сетевой файловой системы rpc.rstatd;
- средства и сервисы работы с сетевым справочником yp-tools и ypbind-mt;
- сервисы удаленной работы xinetd;
- различные средства работы с сетью, такие как inetutils, iproute2, net-snmp, net-tools, nfs-utils, tcpdump, xproto.

Программное средство Sylpheed — это почтовый клиент, разработанный на базе программной библиотеки GTK+. Он поддерживает:

- работу с множественными учетными записями (с возможностью размещения входящей почты как в отдельные папки для каждой записи, так и в общие папки);
- множество почтовых протоколов, таких как POP3/APOP, IMAP4, SMTP, NNTP, работу с протоколом IPv6, легко справляется с большими объемами почты;
- разнообразные и функциональные средства фильтрации как входящей, так и исходящей корреспонденции (по заголовку, теме, тексту тела, размеру, дате, флагам и т. д.) и поиска нужного письма;
- средства многоязыковой поддержки пользовательского интерфейса (в данный момент переведен на 30 языков), хорошо справляется с кодировками и их автоопределением.

Sylpheed обладает модулем защиты от спама с возможностью выбора внешнего обработчика, а также позволяет получать и отправлять почту с использованием множества кодировок.

Средство отображения гипертекстовой информации Links полезно, когда нужно обеспечить максимально возможный функционал оператора с минимальными затратами системных ресурсов. Это быстрый и простой веб-браузер, который может функционировать как в терминальном, так и в текстовом режимах и поддерживать работу с фреймами, вкладками, таблицами и программами на языке JavaScript. Он удобен при работе на серверах, лишенных собственного монитора, к которым можно подключаться только в терминальном режиме, например по протоколу TELNET через каналы связи с низкой пропускной способностью или в условиях воздействия помех.

Links работает в текстовом и графическом режимах и может выдавать подробную информацию об используемых ресурсах, а также о конфигурации сервера, на котором расположен просматриваемый вами сайт.

Сервер работы с сообщениями Sendmail — программное средство рассылки электронной почты через локальную сеть или Интернет. Оно обеспечивает работу модульной системы рассылки, которая предназначена для получения и отправки корреспонденции, а также управления программами подготовки и просмотра почтовых сообщений. Sendmail позволяет организовать почтовую службу локальной сети и обмениваться почтой с другими серверами почтовых служб через специальные шлюзы.

Программные средства доступа к удаленному рабочему столу

В состав комплекса сервисных и пользовательских программ, идущего в комплекте с ОС «Эльбрус», входят средство доступа к удаленному рабочему столу vncviewer и программа «Клиент терминального сервера» (Terminal Server Client). Оба приложения позволяют администратору подключаться к рабочей сессии пользователя таким образом, чтобы и пользователь, непосредственно работающий на ВК, и администратор одновременно видели один и тот же экран, включая все активные приложения и действия (например, перемещение мыши и т. п.). При этом vncviewer в основном используется для подключения к Linux-системам, в то время как «Клиент терминального сервера» может быть использован для доступа и к Windows-, и к Linux-системам.

Сервер доступа к удаленному рабочему столу VNCserver — программный сервер, позволяющий организовать на конкретной вычислительной машине возможность доступа и удаленного управления с помощью специальных средств. Это особенно удобно, когда непосредственный физический доступ к настраиваемой ЭВМ затруднен или невозможен, например, она установлена в промышленном помещении с особыми требованиями к стерильности

атмосферы, или автоматизированном архиве документов, оборудованном автоматической системой пожаротушения, да и просто вследствие физической удаленности объекта.

Средства отладки включают компоненты для отладки и трассировки программного обеспечения — отладчик Gdb и трассировщик Strace. Gdb — консольный отладчик, входящий в состав проекта открытого программного обеспечения, который способен производить отладку программ, написанных на многих языках программирования — C, C++, Free Pascal, FreeBASIC, «Ада», «Фортран» и многих других. Strace — утилита, позволяющая выполнять трассировку сообщений от псевдоустройств (драйверов).

Средства обработки текстов содержат базовый набор компонентов для потокового и интерактивного редактирования текстовой информации, таких как текстовые редакторы ed, ede, vim, а также утилиты работы с переносимыми документами espgs.

Средства работы с периферийными устройствами

Для обеспечения работы с периферийными устройствами введены утилиты работы со звуком, поддержки работы консоли, сервиса консольной поддержки мыши.

Графическая рабочая среда Xfce

Xfce — графическая рабочая среда, которая обычно позиционируется как графический пользовательский интерфейс для компьютеров с ограниченным объемом оперативной памяти. Xfce предоставляет полный набор необходимых оконных приложений и инструменты для администрирования системы, позволяющие обеспечить максимальное удобство работы оператора при минимальном расходе системных ресурсов.

При работе с Xfce предоставляются следующие возможности:

- менеджер окон, всплывающие подсказки, меню, темы изображений;
- поддержка нескольких виртуальных рабочих столов;
- поддержка наиболее распространенных приложений — текстового редактора, органайзера, файл-менеджера, консоли и др.;
- русифицированный интерфейс.

Менеджер Xfce обеспечивает модульность, прозрачность работы с сетью и централизованную систему конфигурации.

4.1.3. Система тестирования ОС «Эльбрус»

Система действует на всех вычислительных комплексах и модулях, работающих под управлением ОС «Эльбрус», и моделирующих их инструментальных комплексах.

Основной состав тестов выполняет:

- моделирование работы многопроцессорной системы в режиме реального времени;
- проверку того, что работа ОС и функционального программного обеспечения основывается на приоритетном планировании всех процессов с возможностью управления приоритетами;
- проверку режимов трассировки ОС и пользовательских программ, сброса, анализа и дампа процесса пользователя;
- проверку выполнения требований технических заданий, включая учет режима жесткого реального времени, контроль доступа субъектов к защищаемым ресурсам по дискреционному и мандатному принципам и контроль управления по потокам информации с помощью меток конфиденциальности;
- проверку режимов перезапуска ВК.

4.2. Особенности оптимизирующего компилятора вычислительных комплексов, построенных на базе микропроцессоров с архитектурой «Эльбрус»

4.2.1. Этапы преобразования исходного кода

Оптимизирующий компилятор с языков C, C++, «Фортран» в составе ОПО «Эльбрус» в полной мере использует возможности распараллеливания, реализованные в аппаратуре каждой из двух архитектурных линий. Для архитектуры SPARC они в основном включают конвейерное параллельное исполнение операций, набор целочисленных и вещественных операций над короткими векторами, а также многоядерность и многопроцессорность на общей памяти с когерентным доступом, свойственные обеим линиям. В то же время для архитектуры «Эльбрус», в которой аппаратная поддержка параллелизма развита гораздо сильнее, спектр вводимых компилятором

оптимизаций и их эффект существенно выше. Именно этот вариант рассматривается в данном разделе.

По отношению к последовательности, характерной для работы типичных компиляторов [30], в оптимизирующий компилятор добавлены несколько этапов, существенно повышающих эффективность результирующего кода.

В качестве первого этапа, предшествующего лексическому анализу, введено препроцессирование. Оно обеспечивает корректное подключение исходной программы и заголовочных файлов, в которых может содержаться информация о программном окружении. На выходе формируется преобразованный текст исходной программы, в котором раскрыты все макроопределения, удалены комментарии, а составные операторы разделены на простые.

Наряду с этим после семантического анализа введены три последовательных этапа, непосредственно связанных с процессом оптимизации.

На этапе глобального межпроцедурного анализа и оптимизации проверяется наличие в синтаксическом дереве рекурсивных функций, которые при обнаружении преобразуются в обычные циклы. Это позволяет уменьшить риск возникновения ошибок переполнения стека. На выходе этапа формируется машинное промежуточное представление исходной программы с минимально оптимизированным кодом.

На этапе глобального попроцедурного анализа и оптимизации контролируются свойства параметров и результатов функций, а также зависимости между операциями в исходной программе. По результатам анализа осуществляется оптимизация машинного промежуточного представления путем упрощения индексных выражений, устранения ненужных вычислений, удаления ненужного копирования данных, лишних обращений в память и ненужного кода. В результате формируется машинное промежуточное представление исходной программы с более оптимизированным кодом.

На следующем этапе — при планировании и распределении регистров — осуществляется поиск независимых друг от друга операций исходной программы. В случае их обнаружения реализуется оптимальное отображение независимых операций исходной программы на аппаратные регистры микропроцессора. На выходе планирования и распределения регистров формируется машинное промежуточное представление исходной программы с максимально оптимизированным кодом [31].

4.2.2. Методы распараллеливания программ

Основным способом повышения производительности компилируемых программ является распараллеливание вычислений на уровне операций. Наиболее сложно реализовать оптимизации этого рода применительно к универсальным целочисленным задачам, которым свойственна сложность управления. Для преодоления связанных с ней проблем применяются различные приемы, эффективность которых зависит от параллельных возможностей архитектуры, таких как спекулятивное выполнение нескольких ветвей программы или спекулятивные обращения в память. Из-за того что подобные операции могут приводить к взаимным помехам, используются методы профилирования (статические и динамические), которые позволяют ограничить спекулятивный параллелизм и рост кода программы [32].

Наиболее полный эффект от распараллеливания на уровне операций достигается при программной конвейеризации циклов, имеющей мощную аппаратную поддержку в виде предварительной подкачки данных и других решений. Она позволяет в несколько раз повысить скорость по сравнению с последовательным выполнением итераций.

Хотя оптимизирующий компилятор может создавать параллельный код в стандартном режиме, предельная производительность достигается при тонкой настройке, которая подбирается для отдельной программы. С этой целью применительно к каждой категории задач, целочисленных и вещественных, были введены наборы оптимизаций, позволяющих существенно увеличить количество выполняемых за такт операций.

За счет распараллеливания на уровне операций логическая скорость выполнения на одном процессоре с архитектурой «Эльбрус» целочисленных и вещественных тестовых пакетов SPEC 2000 возрастает соответственно в 3,01 и 7,66 раза по сравнению с показателями эталонной суперскалярной машины Ultra 10.

Увеличение производительности задач, в которых присутствует параллелизм на уровне данных, достигается автоматической конвейеризацией, базирующейся на использовании коротких векторных инструкций (параллельным исполнением нескольких операций над векторами упакованных данных) [33]. Процесс автоматической векторизации включает последовательность фаз, в совокупности упрощающих программу для последующего анализа, выявляющих участки программы, в которых присутствует параллелизм на уровне данных, позволяющих избежать высоких накладных расходов при векторизации обращений к памяти. Если анализ определил,

что цикл может быть векторизован, компилятор далее раскручивает его и заменяет в нем группы скалярных инструкций соответствующими векторными инструкциями. Наряду со статическим анализом оптимизирующий компилятор для архитектуры «Эльбрус» позволяет получать информацию о компилируемой задаче динамически, во время ее исполнения. Важность динамических проверок весьма велика, поскольку без них невозможно получение высокопроизводительного векторного кода для большинства реальных задач.

Экспериментальная проверка эффективности автоматической векторизации, исследованная на тестовых пакетах и функциях высокопроизводительной библиотеки векторных вычислений EML, показала прирост производительности на несколько десятков процентов. Скорость работы отдельных функций удалось повысить в 10 раз.

Автоматическое распараллеливание на потоки управления базируется на средствах поддержки когерентного доступа в общую память для многоядерных и многопроцессорных систем [19]. В большинстве вычислительных задач наибольшую выгоду приносит распараллеливание циклов, соответственно, в разработанной технике циклы, подходящие для распараллеливания, вырезаются компилятором в отдельные процедуры. В них передается управляющий параметр (идентификатор потока), с помощью которого определяются исполняемая ветка и остальные параметры процедуры, передающиеся через стек. Для комплекса «Эльбрус-3М1» распараллеливание проводится в два потока. Разработанная техника позволяет распараллеливать как отдельный цикл, так и целое гнездо циклов.

Для поиска и корректного распараллеливания циклов используются несколько типов анализа, которые базируются на использовании аналитических структур оптимизирующего компилятора: анализ потока управления для нахождения сходимых циклов, анализ потоков данных, позволяющий определить особенности потока передачи данных в цикле, анализ указателей и зависимостей внутри цикла, выявляющий итерационные зависимости между операциями. Поддержка исполнения распараллеленных программ обеспечивается с помощью библиотеки libepl, предоставляющей компилятору упрощенный интерфейс для управления потоками.

Абсолютный прирост производительности за счет распараллеливания на задачах из пакетов SPEC 95 и SPEC 2000 составил для двухпроцессорного ВК «Эльбрус-3 М1» в среднем 1,42, или 77% от показателей одновременного исполнения двух задач.

4.3. Высокопроизводительная библиотека

Высокопроизводительная библиотека расширяет возможности быстрого и эффективного исполнения программ, предоставляемые оптимизирующим компилятором, применительно к ряду функций, которые имеют ключевое значение в важных приложениях.

Комплекс библиотечных функций, написанных на языке C++ и оптимизированных для исполнения на процессорах семейства «Эльбрус», включает:

- функции для работы с векторами (арифметические, логические, математические, статистические, функции преобразования и т. д.);
- функции линейной алгебры (реализующие функции математических пакетов BLAS1/2/3 и LAPACK);
- функции обработки сигналов (одномерное быстрое преобразование Фурье, статистические функции (корреляционные), генерация сигналов и шумов, генерация окон, изменение частоты и масштаба сигнала);
- функции обработки изображений (создание и уничтожение, заполнение и копирование, арифметические и логические операции, преобразования, статистические функции, фильтрация, цветовые преобразования, двумерные преобразования Фурье);
- функции преобразования видеоизображений (преобразования блоков изображений, квантизация и деквантизация, оценка и компенсация движения и т. д.);
- функции преобразования трехмерных структур (бросание лучей и поиск максимальной интенсивности);
- функции графики (рисование и закрашивание объектов, перекрашивание области).

4.4. Система динамической двоичной трансляции x86 → «Эльбрус»

Неизменно актуальной проблемой при создании новых архитектурных платформ является перенос на них большого объема программного обеспечения, разработанного для уже выпускаемых микропроцессоров, — необходимо либо портировать его, либо создавать заново, что, как правило,

нереально. Если же попытаться обеспечить совместимость создаваемой архитектуры с уже существующими, то ее возможности по части внедрения новых идей будут весьма ограниченными. Хорошо зарекомендовавшим себя способом решения проблемы является технология динамической двоичной трансляции, которая при наличии необходимой аппаратной поддержки была введена в состав ОПО «Эльбрус» [34].

Были реализованы два подхода к построению системы двоичной трансляции. При первом из них она работает между микропроцессором и запускаемыми на нем x86-кодами, транслируя коды BIOS, операционной системы, драйверов и прикладных программ. Вычислительный комплекс на базе микропроцессора «Эльбрус» с системой полной двоичной трансляции для пользователя неотличим от вычислительного комплекса на базе x86-микропроцессоров. При втором подходе эта система является обычным Linux-приложением и работает под управлением ОС Linux. Она позволяет запускать Linux-приложения для платформы x86, которые могут работать одновременно с приложениями в кодах платформы «Эльбрус».

В состав двоичного транслятора включены интерпретатор и три транслятора, функционирующие на последовательных этапах оптимизации. Каждый из них запускается применительно к определенному участку кода, сформированному на предыдущем этапе, если число исполнений этого кода превысит заданный порог. Таким образом увеличивается эффективность результирующего кода.

В процессе интерпретации кода набирается трасса — последовательность (медленно) исполняемых интерпретатором линейных участков, следующих в порядке возрастания адресов. После того как количество исполнений трассы превышает некоторый порог, запускается процедура ее трансляции неоптимизирующим шаблонным транслятором.

Шаблонный транслятор принимает на вход трассу, полученную интерпретатором, и преобразует каждую ее инструкцию в двоичный код целевой архитектуры. Инструкция набирается из нескольких готовых шаблонов (подготовка операндов, операция, запись результатов). Результирующий код помещается в кэш трансляций, благодаря чему оттранслированным кодом можно пользоваться как новой функцией, вызов которой эквивалентен интерпретации трассы.

Быстрый регионный компилятор включает в себя урезанный набор базовых оптимизаций, которые позволяют получить существенный прирост производительности регионов (выделенных областей неоднократно исполняемого

кода) при наименьших затратах времени на их компиляцию. Таким образом достигается оптимальное суммарное время работы всей системы двоичной трансляции. На этом этапе включается механизм планирования, компоновки операций промежуточного представления в широкие команды, также позволяющий осуществить некоторые оптимизации.

Оптимизирующий регионный компилятор является транслятором самого высокого уровня. В результате его работы получается максимально эффективный результирующий код, хотя при этом затрачивается много времени на саму трансляцию. По внутреннему устройству он похож на классические языковые оптимизирующие компиляторы, от которых отличается в основном ограничениями на скорость компиляции (и, соответственно, на использование затратных по времени алгоритмов) и наличием семантических ограничений, накладываемые семантикой исходных двоичных кодов.

На рис. 4.1 показаны результаты сравнения производительности системы полной двоичной трансляции, работающей на микропроцессоре

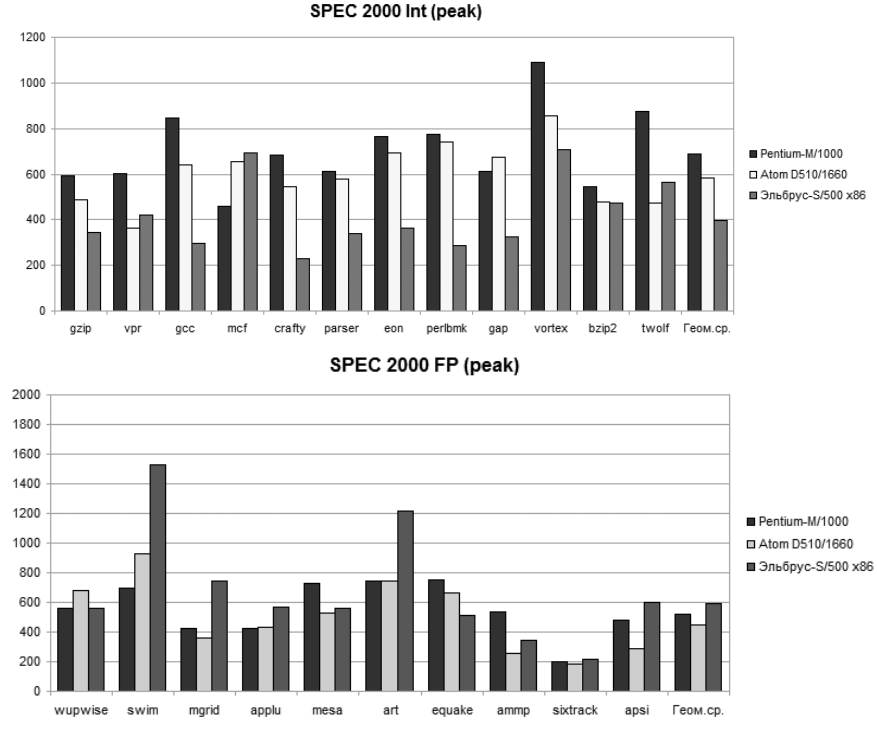


Рис. 4.1. Результаты сравнения производительности системы динамической двоичной трансляции x86 → «Эльбрус» и микропроцессоров с архитектурой x86

«Эльбрус-S» (частота 500 МГц), с производительностью микропроцессоров Pentium-M (1000 МГц) и Atom D510 (1660 МГц), проведенного для целочисленных и вещественных задач.

4.5. Обеспечение защищенного исполнения программ на языках C и C++

4.5.1. Семантические основы

Для реализации сложных систем, сочетающих в себе множество функций, выбираются наиболее гибкие языки программирования. Именно поэтому в качестве базовых языков ОПО «Эльбрус» были выбраны языки C и C++, привлекательной стороной которых является большой набор готовых библиотек и классов, которые динамически подключаются к системе непосредственно во время исполнения. Однако эти языки не имеют безопасных реализаций на существующих архитектурных платформах. В них имеется ряд конструкций, неправильное использование которых может привести к нарушению работы надежного, хорошо проверенного модуля или класса. В первую очередь это относится к операциям доступа в память и передачи управления по динамически вычисляемым адресам. Реализованный в архитектуре «Эльбрус» подход был призван обеспечить гарантию того, что объекты, не входящие в интерфейс модуля, невозможно прочитать или модифицировать из других модулей.

В языках программирования высокого уровня существуют аналогично формулируемые понятия контекста модуля, функции или блока. В частности, контекст модуля составляется из объектов языка, которые могут быть в нем использованы. Проблемы, связанные с контекстом, возникают при переходе от представления модуля на языке высокого уровня к его представлению в архитектуре. Дело в том, что контекст некоторой точки программы в архитектуре может оказаться значительно шире, чем ее же контекст в исходном тексте. Это означает, что на низком уровне существует возможность получить доступ к тем объектам, к которым на высоком уровне в этой точке доступ был закрыт, то есть обойти интерфейс модуля. Такую возможность следует исключить.

4.5.2. Контекстная защита

Концепция контекстной защиты заключается в том, что определенный на этапе сборки контекст модуля не должен быть нарушен во время исполнения. Для доступа к контекстному объекту другого модуля из данного модуля необходимо, чтобы данному модулю была предоставлена ссылка на

этот объект, а все, что связано с процессами создания, передачи, модификации и уничтожения ссылок на объекты, должно находиться под строгим контролем [35].

Контроль границ данных необходим для того, чтобы через ссылку на открытую для доступа переменную невозможно было получить доступ к закрытой (глобальной) или локальной переменной, расположенной в смежной области памяти.

Контроль границ кода должен исключить случайное использование данных в качестве кода и не допускать работы с кодом как с данными, приводящей к возможности модификации кода, а также передачи управления в точку, которая для этого не предназначена.

Контроль соответствия данных и кода требует, чтобы код всегда выполнялся в своем контексте. Если из функции одного модуля вызывается функция другого модуля, то одновременно с передачей управления должна произойти и смена контекстов, то есть передача управления и переключение контекста должны быть объединены в атомарное действие.

Контроль интерфейса функции призван исключить возможность получения доступа к локальным или глобальным объектам модуля, из которого произошел вызов или возврат, путем отделения области передачи параметров и возврата значения от областей локальных данных вызывающей и вызываемой функций.

Контроль чистоты памяти следит, чтобы в составе «мусора», которым заполнена запрошенная и выделенная область памяти, не оказались ссылки на объекты, доступ к которым из данной точки программы должен быть закрыт.

Контроль зависших ссылок исключает доступ к новому объекту через сохранившуюся ссылку на уничтоженный объект, которая является нарушением межмодульной защиты.

4.5.3. Реализация защищенного исполнения программ

Строгая контекстная защита, включая защиту объектов классов, обеспечивается программно-аппаратными средствами. Аппаратная поддержка является обязательной, поскольку одними программными средствами нельзя защититься от подделки указателей и нарушения границ объектов, обнаружить обращение к уничтоженному объекту или предотвратить незаконный доступ к приватным данным объекта.

Однако обеспечить строгую межмодульную защиту только аппаратными средствами невозможно. За формирование контекста любой точки программы отвечают компилятор, реализующий семантику языка, а также редактор связей, объединяющий отдельные единицы компиляции в готовые к выполнению программы. Операционная система, в свою очередь, обеспечивает выделение памяти и формирование ссылок на объекты, а также поддерживает механизм контроля зависших ссылок, защищенную реализацию межпроцедурных переходов и исключений, динамическую загрузку программ, механизмы отладки программ.

Аппаратная поддержка

Модульный подход. Понятие модуля явно отражено в архитектуре «Эльбрус». Модуль представляет собой две логически связанные области, в одной из которых, предназначенной для данных, размещаются глобальные и статические переменные модуля, в другой — функции модуля. С точки зрения архитектуры модуль является единицей защиты. Перед началом исполнения кода функции дескрипторы областей кода и данных того модуля, которому принадлежит эта функция, загружаются в специальные глобальные регистры из таблицы дескрипторов модулей. Они называются дескрипторами текущего модуля и используются аппаратными командами для контроля границ кода и данных модуля в целом. При вызове функции из другого модуля и возврате из нее происходит автоматическое переключение дескрипторов текущего модуля на соответствующие области кода и данных.

Дескрипторы и теги. Понятие дескриптора является обобщением понятия указателя. Дескрипторы используются для представления ссылок на объекты. Принцип, положенный в основу защиты, заключается в том, что доступ к объекту из другого модуля можно осуществить только через дескриптор этого объекта. Дескрипторы внутренних объектов модуля, закрытых для межмодульного доступа, ни в коем случае не должны оказаться в контексте других модулей.

Разница между дескрипторами и обычными указателями заключается в том, что помимо адреса объекта в дескрипторе сохраняется некоторая дополнительная информация, существенная для защиты. Состав этой информации зависит от типа объекта, на который ссылается дескриптор.

Для защиты самих дескрипторов используется механизм тегов. Значение тега дескриптора зависит от типа объекта, на который ссылается дескриптор. Значения тегов данных, которые не являются дескрипторами, отличаются от всех допустимых значений тегов дескрипторов, что позволяет легко

отделить дескрипторы от прочих объектов. Аппаратура не допускает сборки нового дескриптора из отдельных, уже существующих частей, и тем более его создания вручную в обход аппаратуры или операционной системы.

Операции создания и использования дескрипторов осуществляются под строгим контролем аппаратуры. Дескриптор формируется только специальной аппаратной командой при создании объекта текущего модуля или вызове привилегированной функции операционной системы одновременно с созданием нового объекта. Это атомарное действие, в процессе которого проверяется, что создаваемый объект расположен в области кода, которую описывает дескриптор кода текущего модуля.

Операции, предназначенные для работы с дескрипторами определенных объектов, проверяют, что дескрипторам приписаны правильные теги, в то время как все остальные операции «портят» их таким образом, чтобы дескриптор не мог быть больше использован по назначению.

Значения тегов дескриптора, полученного из целого, отличаются от значений тегов настоящих дескрипторов. К такому дескриптору нельзя применять операции чтения или записи, так как доступ по произвольному адресу, полученному из целого числа, является нарушением контекста.

Контроль границ данных и кода обеспечивается за счет того, что в качестве указателей на данные в программах на языках C и C++ применяются дескрипторы переменных и дескрипторы объектов соответственно, а в качестве указателей на функции в программе на языке C применяются дескрипторы функций. Чтение или модификация кода функции через ее дескриптор невозможны. При создании дескриптора функции гарантируется попадание базового адреса в диапазон адресов кода модуля.

Контроль соответствия данных и кода интерфейсу функции обеспечивается за счет того, что атомарные аппаратные операции межпроцедурных передач управления включают в себя автоматическое переключение контекста.

Связующая информация, на основании которой выполняются возврат из функции и межпроцедурные переходы, а также переключаются контексты, размещается не непосредственно в стеке вызовов, а в отдельном системном стеке, который недоступен непривилегированной программе.

Все возможные для дескрипторов текущего модуля значения формируются на этапе загрузки модулей и складываются в системную таблицу дескрипторов модулей, недоступную непривилегированной программе. При

переключении контекстов модулей значения из этой таблицы загружаются в дескрипторы модуля.

Описание области для параметров вызова и для возврата значения передается в виде дескриптора. Нарушить границы этой области невозможно даже в случае передачи переменного числа параметров.

В отличие от механизма вызовов и возвратов из функций, механизм межпроцедурных переходов имеет минимальную аппаратную поддержку и в основном базируется на программной реализации, которая обсуждается далее.

Контроль чистоты памяти и зависших ссылок осуществляется под управлением защищенной операционной системы и рассматривается в следующем разделе. Во время выделения области в стеке вызовов при очередной активации функции происходит очистка этой области. Для контроля ссылок на уничтоженные локальные переменные применяется аппаратно-программный механизм, в котором аппаратная часть реализует создание ссылок таким образом, чтобы время жизни ссылки не могло быть больше времени жизни самой локальной переменной.

Поддержка в операционной системе

Выделение и освобождение памяти под объекты. Операционная система должна взять на себя реализацию части функций управления памятью, которые традиционно включались в библиотеки динамической поддержки реализации языков. В первую очередь это касается функций выделения и освобождения памяти под объекты, таких как `malloc`, `realloc`, `calloc`, `free` в языке C, а также `new`, `delete` в языке C++. При этом должны решаться сразу несколько проблем, связанных с защитой, с одной стороны, и с эффективной реализацией функций управления памятью — с другой.

С точки зрения защиты операционная система должна взять под полный контроль выделение памяти под объекты и формирование тегированных ссылок на них. Наиболее эффективным представляется выделение памяти квантами, пропорциональными степеням числа 2, под объекты, размер которых не превышает одной виртуальной страницы (при этом память под объекты больших размеров может выделяться квантами, пропорциональными размеру страницы). Объекты, занимающие одинаковые кванты памяти, размещаются в одной странице (в дескриптор объекта при этом записывается точный размер этого объекта). Для такой страницы заводится специальная документация, содержащая информацию о занятых квантах внутри страницы и адресе первого свободного кванта внутри страницы, по которому и будет размещаться новый объект при его генерации. При освобождении квант прописывается

значениями «неинициализированные данные», его не занимают под новые объекты, а при освобождении всех квантов внутри страницы она помечается как свободная. Если уничтожается объект, занимающий полные страницы, последние также помечаются как свободные.

Контроль над указателями, смотрящими в стек. Когда значение ссылки на локальную переменную записывается в объект, время жизни которого больше, чем время жизни самой переменной, возникает аппаратное прерывание. Далее операционная система должна обеспечить контроль над такими указателями. Возможны две реализации такого контроля: первая базируется на учете всех ссылок на локальную переменную процедуры в специальной документации, связанной с активацией этой процедуры; вторая использует дополнительные виртуальные страницы, которые совмещаются по физической памяти со страницами, содержащими соответствующую локальную переменную в стеке.

Поддержка реализации межпроцедурных переходов. Реализация межпроцедурных переходов, таких как `longjmp` в языке C или `throw` в языке C++, связанная с определением точного места, куда должно быть передано управление после раскрутки стека, также требует специальной поддержки со стороны операционной системы. Существенную трудность представляет реализация пары функций `setjump` — `longjmp` языка C, вызываемых в различных модулях. Они оперируют общим буфером `jump_buf`, через который передается информация, влияющая на корректность исполнения модуля, вызвавшего `setjump`. Искажение информации в буфере с последующим вызовом функции `longjmp` может привести к неверной работе функции после передачи в нее управления, что эквивалентно нарушению межмодульной защиты.

Межпроцедурные переходы чаще всего используются для обработки исключительных ситуаций. Подготовка к обработке исключительных ситуаций выполняется всегда, когда есть вероятность их возникновения, но сами исключительные ситуации возникают сравнительно редко. Таким образом, в механизме нелокальных переходов установка метки, соответствующей адресу возврата из функции `set_jump`, должна быть простой и быстрой операцией, от которой зависит эффективность всего механизма, в то время как сам переход может выполняться сравнительно медленно. Принятая реализация, которая опирается на аппаратные средства поддержки вызова процедур и защиту метки процедуры от подмены с помощью тега, сохраняет в `jmp_buf` информацию, необходимую для гарантированного возврата в функцию, из которой произошел вызов `setjump`, по правильному адресу. Однако из-за того, что `jmp_buf` является глобальной структурой данных, доступной всем модулям, адрес возврата в нем может быть подменен адресом

возврата межпроцедурного перехода из другого `jmp_buf`. Чтобы гарантировать, что управление будет передано на ожидаемый адрес возврата, в `jmp_buf` помещается дополнительная метка, на которую передается управление из любого вызова функции `longjmp`. Дополнительно эта функция передает адрес метки перехода и адрес возврата из цепочки вызовов, приведшей к ее вызову. По результату сравнения этих двух адресов определяется, была ли вызвана соответствующая функция `setjump`, и только если это так, управление передается на нужный адрес.

При реализации механизма исключений языка C++ используется другой интерфейс с операционной системой. То, что языковый интерфейс не требует заведения глобальных буферов, а может быть реализован через механизм параметров, существенно облегчает поддержку безопасной межпроцедурной передачи управления. В стеке вызовов в связующей информации делается пометка функции, в которой встретился оператор `try` (установлена «ловушка»). Исполнение оператора приводит к вызову функции операционной системы, которая ищет по стеку помеченную функцию, в ее локальных данных находит структуру с меткой передачи управления, заносит в эту структуру ссылку на тип исключения и передает управление в функцию. Далее сама функция проверяет тип исключения и передает управление найденному оператору `catch` или повторно возбуждает исключение, если оператор не найден.

Поддержка в компиляторе и редакторе связей

Реализация операций с адресами. Компилятор использует специальные аппаратные команды при работе с адресами. В первую очередь это относится к адресной арифметике. В то время как простейшие операции продвижения указателя по массиву реализуются обычными арифметическими командами, получение дескриптора подмассива или дескриптора поля объекта или структуры требует использования специальных операций.

Формирование дескрипторов объектов реализуется посредством вызова специальных функций операционной системы, которые одновременно выделяют память под эти объекты, а для заведения объектов и локальных областей процедур в стеке используются специальные аппаратные команды, вырабатывающие дескрипторы этих объектов. При формировании дескрипторов функций также используются специальные аппаратные команды. Размещение данных в памяти требует правильного выравнивания всех ссылок на объекты.

Инициализация глобальных данных. Присваивание начальных данных глобальным переменным обычно выполняется операционной системой при

загрузке программы с использованием образа памяти, в котором ссылки на объекты программы корректируются с помощью таблицы перемещений. Такой подход неприемлем, поскольку позволяет нарушить защиту. Для формирования ссылок в глобальных переменных компилятор создает специальный код инициализации, который формирует ссылки на глобальные переменные и на функции, используя для этого дескрипторы модуля и соответствующие аппаратные команды. При статической сборке программы редактор связей объединяет коды инициализации отдельных единиц компиляции в одну функцию. Функция инициализации запускается при загрузке каждого модуля перед началом исполнения программы.

Копирование данных и объектов. Операторы присваивания структур, массивов и объектов часто требуют копирования больших кусков памяти. Зачастую такие действия реализуются вызовом библиотечной функции `memcopy`, которая выполняет побайтовую пересылку данных. Однако в защищенном режиме исполнения тегированные данные при таком копировании теряют свои теги, а дескрипторы массивов и объектов превращаются в числовые данные, которые нельзя использовать для доступа в память. Чтобы избежать этого, копирование необходимо выполнять поэлементно, используя для этого специальные аппаратные команды, позволяющие сохранить ссылки в неприкосновенности.

Еще одна проблема — это копирование неинициализированных данных. Семантика языка позволяет использовать частично инициализированные данные при копировании (например, в конструкторе копирования языка C++), поэтому при выполнении этой операции ошибки не должны выдаваться. Это достигается также с помощью специальных операций пересылки данных.

Интерфейс редактора связей и загрузчика. При загрузке модуля необходимо сформировать все его внешние ссылки на другие модули (ссылки на глобальные данные, функции и классы). Такие ссылки размещаются в специальной области глобальных данных модуля. Редактору связей передаются дескрипторы всех модулей, и по подготовленной компилятором информации он формирует дескрипторы на объекты чужих модулей для каждого модуля, участвующего в связывании. Таким образом, редактор связей также участвует в безопасной реализации языков программирования.

Результаты переноса программ в среду защищенного исполнения

При переносе программ в среду защищенного исполнения был обнаружен ряд проблем, которые распределяются по нескольким группам:

1. Обращение к неинициализированным данным (хотя контроль их использования не является обязательным с точки зрения защищенного исполнения, но является очень распространенной ошибкой).
2. Выходы за границу объектов (массивов) — ошибка переполнения буфера.
3. Использование свойств аппаратной платформы, таких как размеры типов данных, выравнивания указателей по размерам данных числовых типов и пр.
4. Отклонения от стандарта языка, такие как использование неявных типов данных, характерное для старого стиля программирования (стиль Кернигана — Ритчи), работа со стандартными библиотеками без предварительных описаний функций, использование конкретной реализации языковых конструкций с неопределенным поведением.
5. Преобразование целого в указатель.
6. Запись в глобальную память ссылок, смотрящих на локальные переменные.

Эти группы проблем можно объединить в три категории.

1. Реальные ошибки в программах, которые представляют собой явное нарушение защиты или могут привести к нему. К этой категории относятся проблемы из групп 1 и 2.
2. Опасная работа с указателями, которая может привести к нарушениям защиты. К этой категории относятся проблемы из групп 5 и 6.
3. Использование непереносимых свойств языка или его конкретной реализации, которые являются источником трудностей переноса и также могут привести к ошибкам. К этой категории относятся проблемы из групп 3 и 4.

Перечисленные проблемы потребовали доработки и исправления ошибок в соответствующих программах, чтобы можно было исполнить их в защищенной среде. Вместе с тем после отладки исправления ошибок программы могут успешно исполняться на обычных машинах (это было практически проверено), где отсутствует само понятие защищенного исполнения, но при этом будут обладать более высокой надежностью.

При защищенном исполнении программ на языках C и C++, оттранслированных с использованием описанной аппаратной и системной поддержки, фиксируются любые нарушения защиты памяти, воспринимаемые

в обычных системах как неопределенное поведение. При этом опасные и сложные ошибки, отнимающие в традиционных системах значительное время даже у опытных программистов, хорошо локализуются. Благодаря режиму защищенного исполнения на государственных испытаниях ВК «Эльбрус-3М1» была продемонстрирована высокая эффективность в части обнаружения ошибок (табл. 4.1), в том числе при переносе и исполнении задач пользователей, международного пакета SPECint 95 и международного пакета SAMATE, который содержит собранные по всему миру ошибочные фрагменты широко распространенных программ. Основные типы обнаруженных ошибок — нарушение границ объектов (buffer overflow), использование неинициализированных данных, опасных конструкций языка или опасных отклонений от стандарта языка.

Таблица 4.1. Эффективность поддержки защищенной реализации языков программирования

Категория задач	**Всего задач**	**Задач с найденными ошибками**
Задачи пользователей	7	4
Пакет SPECint95	8	7
Пакет негативных тестов SAMATE на нарушение защиты	888	874

Таким образом, обеспечиваются условия для создания надежного программного обеспечения при выполнении в сжатые сроки масштабных проектов с участием больших коллективов разработчиков.

Контрольные вопросы и задания

1. Какие виды операционных систем поддерживаются для вычислительных комплексов «Эльбрус»?
2. Какой из режимов реального времени поддерживает ОС «Эльбрус»?
3. Какие средства ОС «Эльбрус» предназначены для поддержки пользовательского интерфейса?
4. Для чего предназначена система тестирования ОС «Эльбрус»?
5. Перечислите, какие тесты входят в состав системы тестирования.

6. Для чего предназначена система динамической двоичной трансляции?
7. Какие инструменты включает в себя система динамической двоичной трансляции?
8. Из каких частей состоит система программирования ВК «Эльбрус»?
9. Что такое транслятор? Какие виды трансляторов существуют? Какой используется в системе программирования ВК «Эльбрус»?
10. Какие этапы трансляции характерны только для оптимизирующего компилятора «Эльбрус»?
11. Перечислите методы распараллеливания, которые могут применяться для оптимизации программ в вычислительных комплексах семейства «Эльбрус».
12. Какие методы распараллеливания используются оптимизирующим компилятором «Эльбрус»?
13. Каково назначение системы обеспечения двоичной совместимости?
14. Каково назначение высокопроизводительной библиотеки?
15. Какие группы функций реализует высокопроизводительная библиотека?
16. В чем заключается концепция контекстной защиты?
17. В чем заключается контроль границ данных?
18. В чем заключается контроль границ кода?
19. В чем заключается контроль интерфейса функции?
20. Каким образом реализуется защищенное исполнение программ?
21. Каким образом защищенное исполнение программ поддерживается операционной системой?
22. Каким образом защищенное исполнение программ поддерживается компилятором?
23. Каким образом защищенное исполнение программ поддерживается редактором связей?

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1
Вычислительные комплексы «Эльбрус-90микро» (варианты исполнения)

ВК «Эльбрус-90микро» в шкафном исполнении для стационарных применений (рис. П1.1 и П1.2, табл. П1.1) предназначен для использования в информационно-вычислительных и управляющих системах, в том числе системах непрерывного действия, работающих в реальном масштабе времени, а также в научных и промышленных вычислительных центрах коллективного пользования.

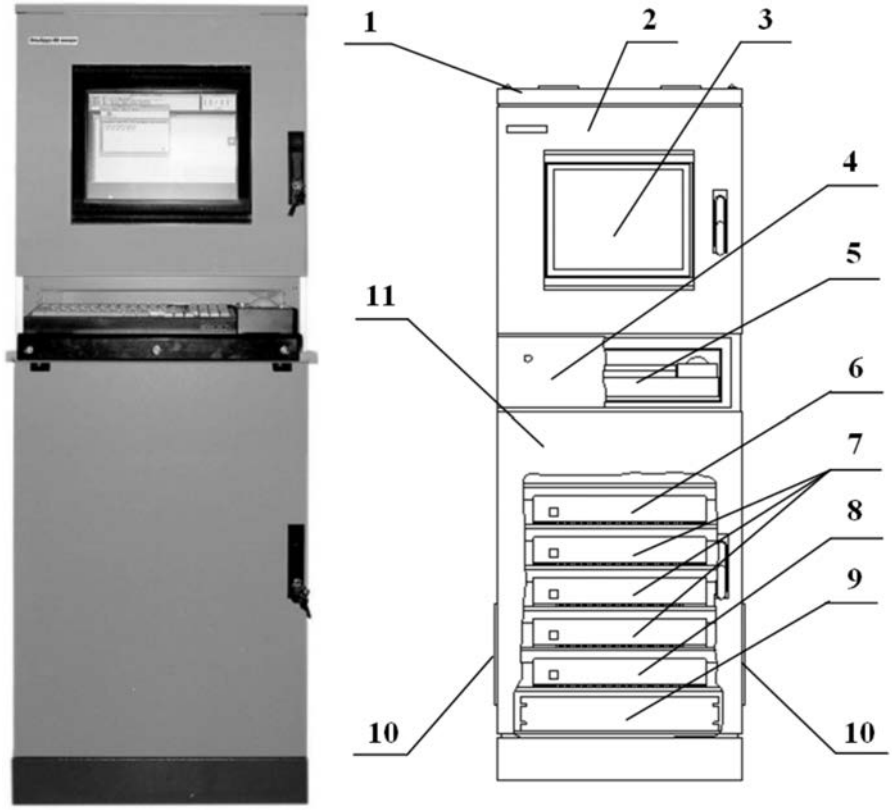


Рис. П1.1. ВК «Эльбрус-90микро» в шкафном исполнении для стационарных применений: *1* — шкаф; *2* — дверь шкафа отсека видеомонитора; *3* — видеомонитор ММП15; *4* — крышка шкафа отсека пульта оператора; *5* — пульт оператора КМ 114-60-20; *6* — устройство вычислителя системного УВС-М; *7* — устройство сопряжения с внешними абонентами УСВА-М; *8* — устройство сопряжения УС19-М; *9* — источник бесперебойного питания (ИБП); *10* — воздушный фильтр; *11* — дверь шкафа отсека аппаратуры

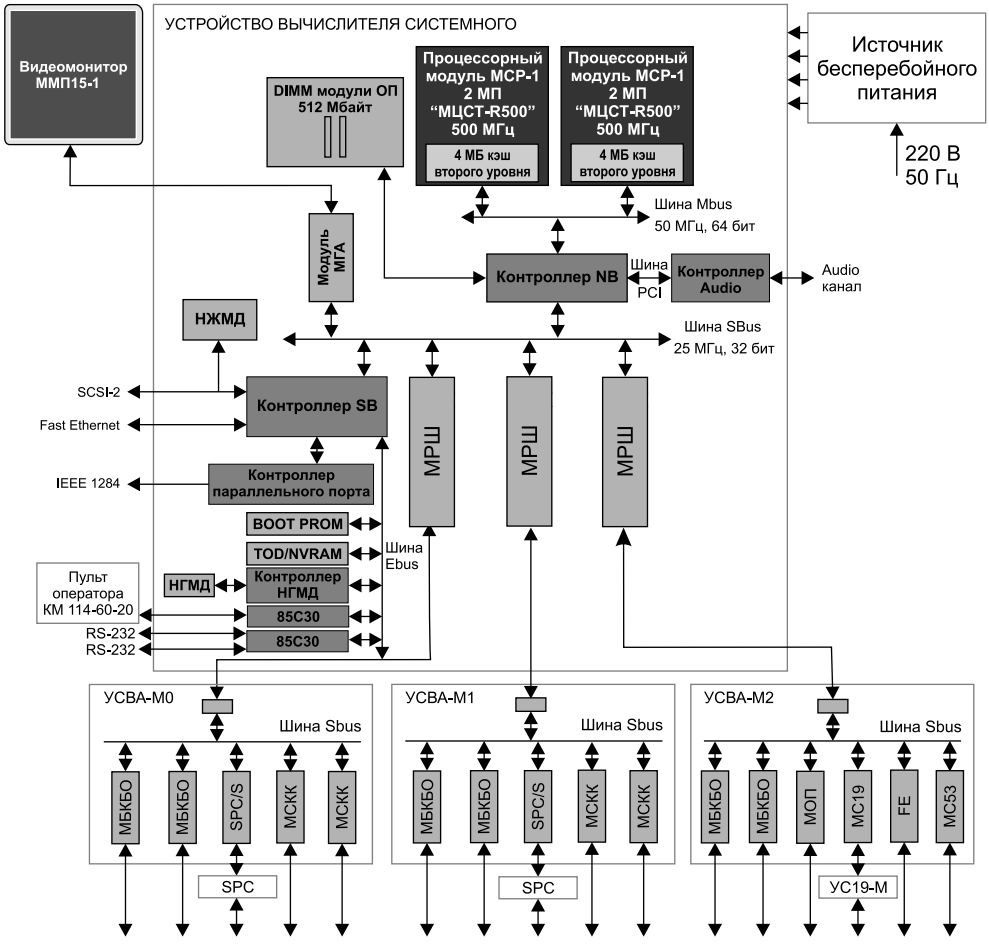


Рис. П1.2. Структурная схема стационарного ВК «Эльбрус-90микро»

Таблица П1.1. Технические характеристики стационарного ВК «Эльбрус-90микро»

Параметр	**Значение**
Тип процессора	МЦСТ-R500
Количество процессоров	2...4
Тактовая частота процессора, МГц	500
Производительность одного процессора, MIPS/MFLOPS	440/205
Оперативная память, Мбайт	512
Внутрипроцессорная кэш-память, Кбайт	48 (32 для данных, 16 для команд)

Таблица П1.1 (*окончание*)

Параметр	Значение
Внешняя кэш-память, Мбайт	4
Встроенная в УВС-М внешняя память, Гбайт, не менее	36
Напряжение питающей сети, В	220 ± 22
Частота питающей сети, Гц	50 ± 1
Потребляемая мощность, Вт, не более	800
Система охлаждения	Встроенная воздушного типа
Каналы ввода/вывода	Fast Ethernet, SCSI, ЕС ЭВМ, RS-232/RS-423, Centronics, БК-3М, с АПД 5Ц19, 5Ц53
Средняя наработка на отказ, ч	9000
Срок службы, лет	12

Перебазируемый ВК «Эльбрус-90микро» в шкафном исполнении (рис. П1.3 и П1.4, табл. П1.2) предназначен для использования в мобильных системах управления и обработки информации.

Рис. П1.3. Внешний вид перебазируемого ВК «Эльбрус-90микро»

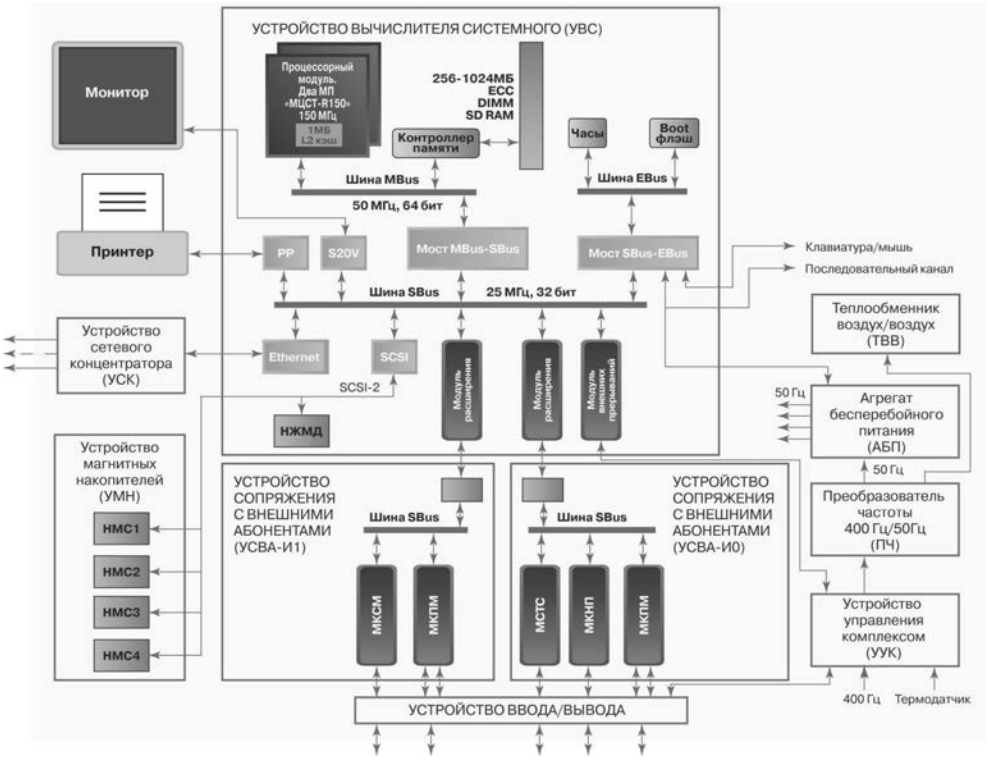


Рис. П1.4. Структурная схема перебазируемого ВК«Эльбрус-90микро»

Таблица П1.2. Технические характеристики перебазируемого ВК «Эльбрус-90микро»

Параметр	**Значение**
Тип процессора	МЦСТ-R150
Количество процессоров	4
Тактовая частота процессора, МГц	150
Объем оперативной памяти, Мбайт	256...1024
Объем дисковой памяти, Гбайт	2...36
Производительность ВК в полной комплектации (SPECint95/SPECfp95)	10/8
Первичная сеть	220 В/50 Гц, 220 В/400 Гц
Энергопотребление, Вт	400
Система охлаждения	Встроенная воздушного типа
Возможности расширения сопряжения с внешними устройствами	SBus (12 слотов)
Средняя наработка на отказ, ч	9000
Срок службы, лет	12

Конструкция ВК представляет собой герметичный, теплоизолированный шкаф, обеспечивающий пыле- и влагозащиту, имеющий системы подогрева и термостатирования.

Электронное оборудование ВК размещено в высокопрочных шасси конструктива, обеспечивающего стойкость к вибро- и ударным нагрузкам. Система электропитания комплекса устойчива к перебоям электроснабжения в первичной электросети.

ВК «Эльбрус-90микро», выполненные в конструктиве «Евромеханика» в соответствии с требованиями стандарта cPCI, рассчитаны на использование в жестких условиях эксплуатации. Далее представлены два варианта конструктивного исполнения этих комплексов — для перебазируемых (рис. П1.5) и встраиваемых (рис. П1.6) систем.

В перебазируемых системах можно использовать 4-слотовый и 8-слотовый варианты. Они имеют герметичный корпус со встроенной замкнутой системой охлаждения воздушного типа. Система охлаждения 8-слотового корпуса имеет также внешний незамкнутый контур охлаждения. В корпус вставляются модули вычислительного оборудования, выполненные в формате «Евромеханика» 6U. Конструкция корпусов обеспечивает работоспособность оборудования в условиях вибро- и ударных нагрузок.

Рис. П1.5. 8-слотовый ВК «Эльбрус-90микро» для перебазируемых систем

Технические характеристики 8-слотового ВК приведены в табл. П1.3, структурная схема — на рис. П1.7.

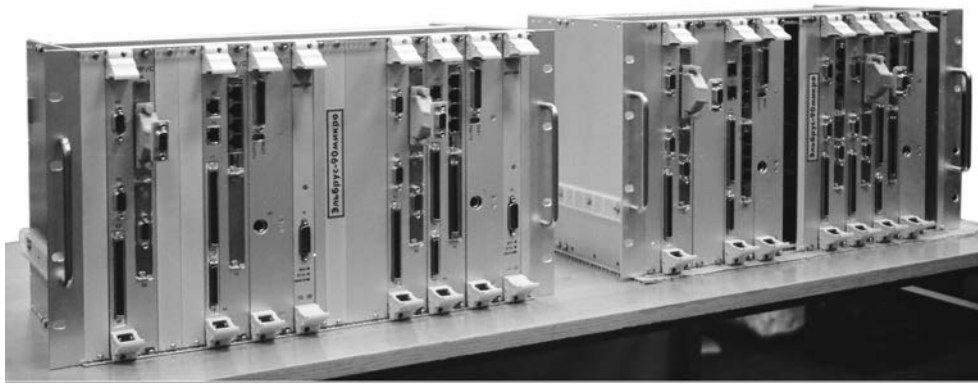


Рис. П1.6. Встраиваемые ВК «Эльбрус-90микро»

Таблица П1.3. Технические характеристики 8-слотового ВК «Эльбрус-90микро»

Параметр	**Значение**
Тип процессора	МЦСТ-R500
Количество процессоров	4
Тактовая частота процессора, МГц	500
Объем оперативной памяти, Мбайт	1024
Объем дисковой памяти, Гбайт	36
Производительность в полной комплектации (SPECint95/SPECfp95)	35/28
Первичная сеть	220 В/50 Гц, постоянный ток 27 В
Энергопотребление, Вт	650
Средняя наработка на отказ, ч	9000
Среднее время восстановления, мин	20
Срок службы, лет	12

ВК «Эльбрус-90микро» в конструктиве автоматизированного рабочего места (АРМ) оператора (рис. П1.8 и П1.9, табл. П1.4) предназначен для использования в качестве рабочего места оператора информационно-вычислительных и управляющих систем, в том числе системах непрерывного действия, работающих в реальном масштабе времени, комплектования стендов для разработки функционального программного обеспечения, в качестве рабочих мест программистов и др.

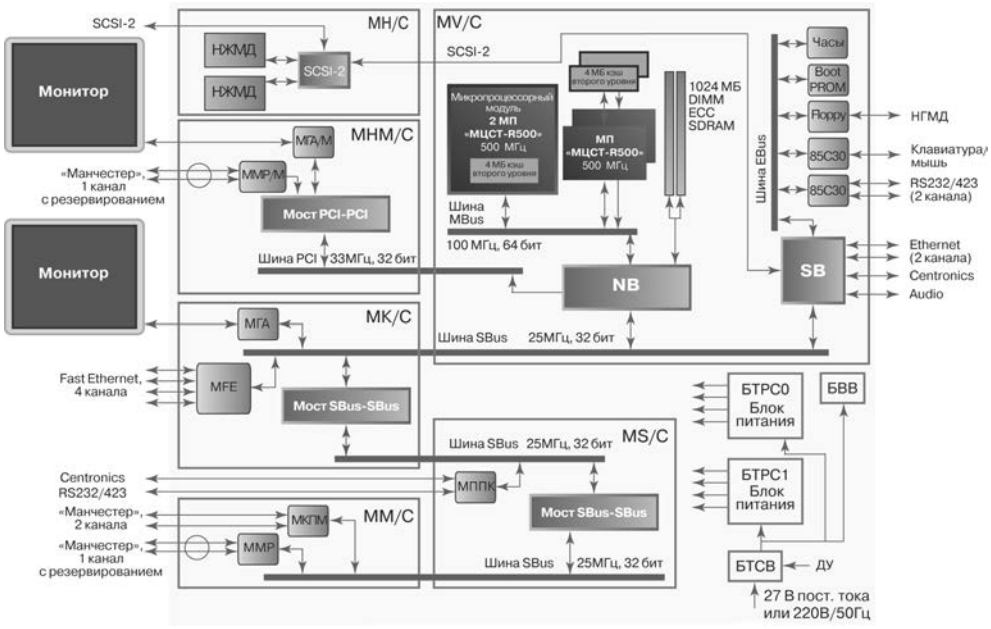


Рис. П1.7. Структурная схема 8-слотового ВК «Эльбрус-90микро»

Рис. П1.8. ВК «Эльбрус-90микро» в конструктиве АРМ оператора

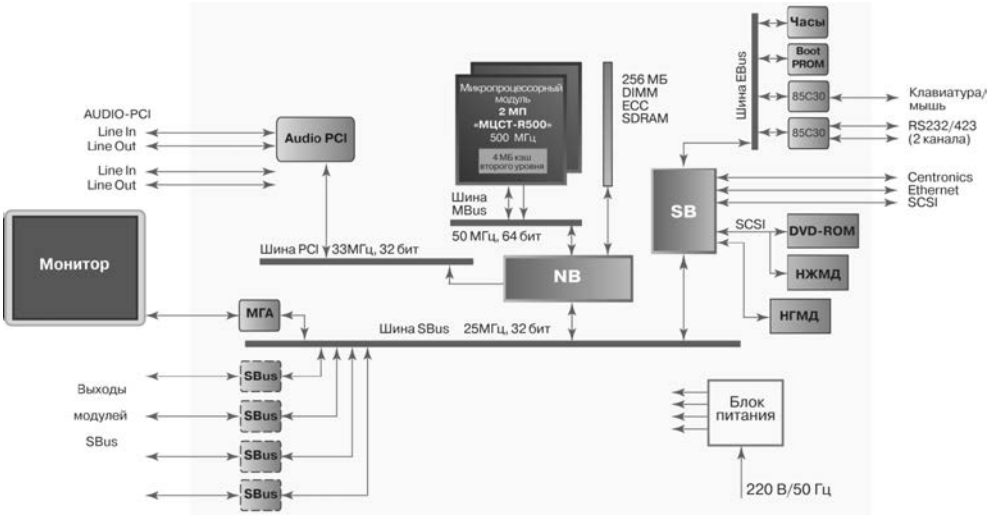


Рис. П1.9. Структурная схема ВК «Эльбрус-90микро» в конструктиве АРМ оператора

Таблица П1.4. Технические характеристики ВК «Эльбрус-90микро» в конструктиве АРМ оператора

Параметр	**Значение**
Тип процессора	МЦСТ-R500
Количество процессоров	До 4
Тактовая частота процессора, МГц	500
Объем оперативной памяти, Мбайт	256
Объем дисковой памяти, Гбайт	36
Производительность ВК в полной комплектации (SPECint95/ SPECfp95)	10/8
Периферийная шина	SBus (4 слота)
Первичная сеть	220 В/50 Гц
Энергопотребление, Вт	100
Средняя наработка на отказ, ч	10 000
Среднее время восстановления, мин	20

Вычислительные комплексы «Эльбрус-90микро» в конструктиве PC ATX (рис. П1.10 и П1.11, табл. П1.5) предназначены для использования в стационарных системах управления и обработки информации. Они изготавливаются

в двух вариантах исполнения — однопроцессорном и двухпроцессорном. Однопроцессорный ВК оснащен периферийными шинами SBus и PCI, двухпроцессорный имеет периферийные шины PCI и порты USB.

Рис. П1.10. ВК «Эльбрус-90микро» в конструктиве PC ATX

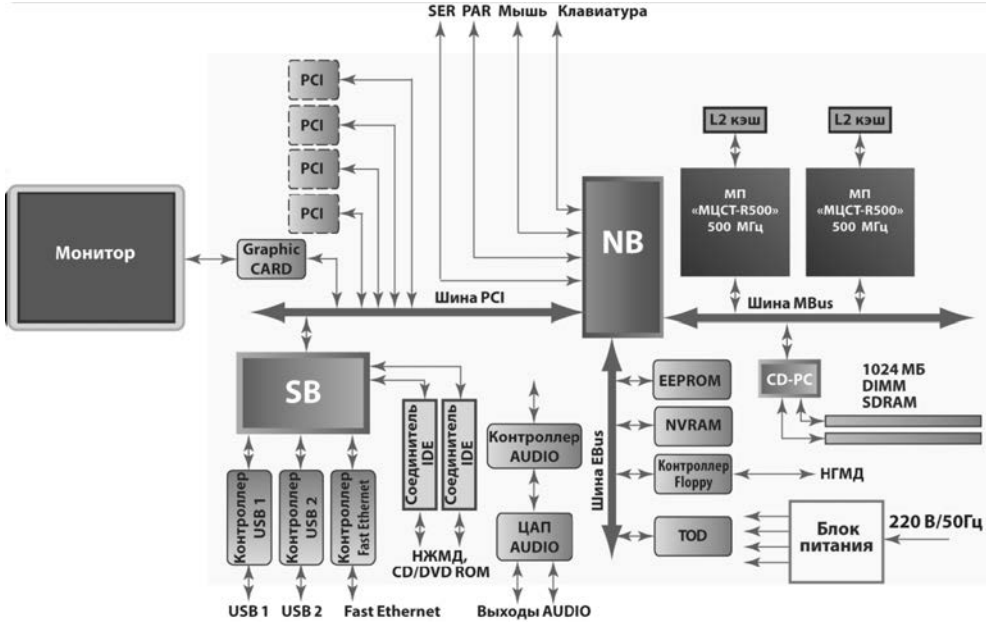


Рис. П1.11. Структурная схема двухпроцессорного ВК «Эльбрус-90микро» в конструктиве PC ATX

Таблица П1.5. Технические характеристики ВК «Эльбрус-90микро» в конструктиве PC ATX

Параметр	Значение	
	Однопроцессорный	**Двухпроцессорный**
Тип процессора	МЦСТ-R500	МЦСТ-R500
Количество процессоров	1	2
Тактовая частота процессора, МГц	500	500
Объем оперативной памяти, Мбайт	512...1024	512...1024
Объем дисковой памяти, Гбайт	36	36
Производительность в полной комплектации (SPECint95/SPECfp95)	10/8	20/16
Периферийная шина	PCI, 4 слота	PCI, 4 слота
Первичная сеть	220 В/50 Гц	220 В/50 Гц
Энергопотребление, Вт	100	100
Средняя наработка на отказ, ч	10 000	10 000
Среднее время восстановления, мин	20	20

Вычислительный комплекс «Эльбрус-90микро» для носимых применений (ноутбук) в жестких условиях эксплуатации (рис. П1.12 и П1.13, табл. П1.6) производится в конструктиве фирмы «Элинс» и обладает надежностью, низким энергопотреблением и компактностью.

Рис. П1.12. Внешний вид ноутбука

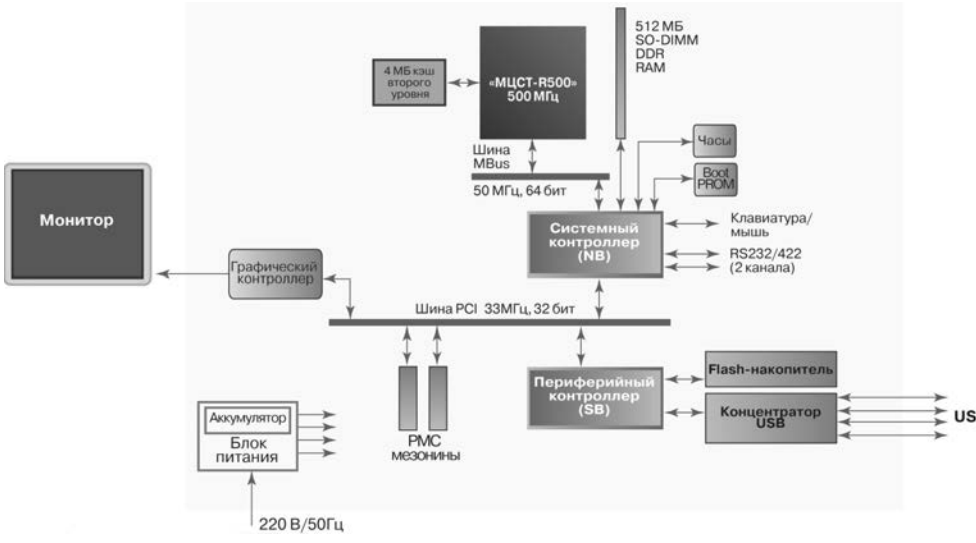


Рис. П1.13. Структурная схема ноутбука

Таблица П1.6. Технические характеристики ноутбука

Параметр	**Значение**
Тип процессора	МЦСТ-R500
Тактовая частота процессора, МГц	500
Производительность, MIPS/MFLOPS	500/200
Объем оперативной памяти, не менее, Мбайт	512
Объем видеопамяти, не менее, Мбайт	8
Диагональ экрана, ..."	15
Разрешение экрана, бит	1024×768×18
Яркость экрана, кд/м^2	350
Флеш-накопитель, Гбайт	До 4
Каналы ввода/вывода: — USB — RS232/422 — Ethernet 10/100	 До 4 До 2 1
Средства расширения	2 PMC-мезонина
Спутниковая навигация	GPS/«ГЛОНАСС»-приемник
Питание, В	10...36, постоянный ток
Потребляемая мощность, Вт	27

Параметр	Значение
Встроенная аккумуляторная батарея	3 А/ч, 16,8 В
Внешняя аккумуляторная батарея	10 А/ч, 14,4 В
Рабочая температура, °С	–20...+50
Вибростойкость	2g
Удары многократные	15g, 5...15 мс
Удары однократные	100g, 15 мс
Допустимая высота падения на пол, м	0,75
Возможность применения в дегазирующих растворах и агрессивных средах	Да
Защита от статической и динамической пыли	Да
Возможность применения в соляном (морском) тумане	Да

Перечень интерфейсных модулей ВК «Эльбрус-90микро» приведен в табл. П1.7.

Таблица П1.7. Перечень интерфейсных модулей ВК «Эльбрус-90микро»

Тип модуля	Технические характеристики	Конструктивное исполнение
МРШ Модуль расширения шины SBus	Подключение до 100 каналов (контроллеров)	1-слотовая SBus-карта
МСКК Модуль стандартного канала «канал — канал»	Темп обмена — 600 Кбайт/с; расстояние связи — 50 м; количество каналов — 8	1-слотовая SBus-карта
МСКК/С Модуль стандартного канала «канал — канал»	Темп обмена — 600 Кбайт/с; расстояние связи — 50 м; количество каналов — 2 × 8	CompactPCI, «Евромеханика 6U»
МСКА Модуль стандартного канала «канал — абонент»	Темп обмена — 600 Кбайт/с; расстояние связи — 50 м; количество каналов — 16	1-слотовая SBus-карта
МБКБО Модуль быстрого оптического канала	Темп обмена — 80 Мбайт/с; расстояние связи — 500 м; количество каналов — 1	1-слотовая SBus-карта
МС53 Модуль связи с АПД 5Ц53	Темп обмена — 15 Кбит/с; расстояние связи — 400 м; количество каналов — 2	1-слотовая SBus-карта

Таблица П1.7 (*окончание*)

Тип модуля	**Технические характеристики**	**Конструктивное исполнение**
МС19 Модуль связи с АПД 5Ц19 (программный и аппаратный варианты)	Темп обмена — 9,6 Кбит/с; расстояние связи — 20 м; количество каналов — 4	1-слотовая SBus-карта
МКПМ, ММР, ММР/М Модули последовательного канала по ГОСТ 267.65.52–87 («Манчестер 2»)	Темп обмена — 1 Мбит/с; расстояние связи — по ГОСТ 267.65.52–87; количество каналов — 1 + резерв	МКПМ, ММР: 1-слотовая SBus-карта; ММР/М: CompactPCI (мезонин)
ММПК (МПК) Модули последовательных и параллельных каналов («Стык-2», RS-232 и IEEE 1284)	Количество каналов: — RS-232 — 8; — IEEE 1284 — 1	1-слотовая SBus-карта
ММ/С, ММР/С Модули последовательного канала по ГОСТ 267.65.52–87 («Манчестер 2») и расширителя SBus	Темп обмена — 1 Мбит/с; расстояние связи — по ГОСТ 267.65.52–87; количество каналов: — «Манчестер 2» — 2; ММР/С — резерв; количество слотов SBus — 2	CompactPCI, «Евромеханика 6U»
МГА, МГА/М, МГА/Р Модули графических адаптеров	МГА, МГА/М: 1280 × 1024, 76 Гц, 8...32 бит, 8 Мбайт; МГА/Р: 1600 × 1200, 60 Гц, 8...32 бит, 8 Мбайт	1-слотовая SBus-карта, CompactPCI (мезонин), PCI

ПРИЛОЖЕНИЕ 2
Шина MBus

Соединение микропроцессорных модулей с оперативной памятью и модулями ввода/вывода выполняется через высокопроизводительную 64-разрядную шину MBus. Она полностью синхронная, все передачи управляются синхроимпульсами с частотой 40...120 МГц. Шина поддерживает блочные передачи размером до 128 байт с пиковой скоростью передачи 800 Мбайт/с.

MBus обеспечивает командное слово в начале каждой транзакции и данные при завершении транзакции. Командное слово содержит адрес памяти, тип транзакции (например, чтение или запись), ее размер (например, 1 байт или 32 байта) и другую информацию.

Ведущее устройство, которое начинает транзакцию на шине, называется мастером. Ведомое устройство, в которое адресуется транзакция, называется подчиненным.

В MBus-системе применяется централизованный арбитраж. Каждое устройство, которое может быть мастером, имеет сигнал запроса к арбитру и сигнал разрешения от арбитра. Устройства могут использовать шину, когда она разрешена, если шина свободна.

Шина MBus поддерживает многопроцессорность с когерентными кэшами. Протокол когерентности кэша, используемый в MBus, совпадает с протоколом MOESI (modified, owned, exclusive, shared, invalid), использующим пять возможных состояний общих данных: модифицированный, собственный, исключительный, разделяемый и незначащий [Jim Handy].

Назначение физических сигналов шины MBus представлено в табл. П2.1.

Сигналы могут быть входными I (In) или выходными O (Out).

Рассмотрим подробно назначение каждого сигнала.

MCLK — сигнал синхронизации MBus.

MAD[63:0] — адрес памяти и данные памяти. Во время адресной фазы MAD[35:0] содержит физический адрес. Остальные сигналы MAD[63:36] содержат специфическую информацию, которая будет описана далее. Во время фазы данных MAD[63:0] содержит данные для передачи. Байты расположены, как показано на рис. П2.1. Для передач, содержащих меньше чем двойное слово (8 байтов), данные должны быть выровнены.

Таблица П2.1. Назначение физических сигналов шины MBus

Сигнал	Вход или выход	Назначение сигнала
MCLK	I	Синхроимпульсы MBus
MAD[63] — MAD[0]	I/O	Адрес, управление, данные
MAS#	O	Строб адреса
MRDY#	O	Готовность данных
MRTY#	I	Повтор
MERR#	I/O	Ошибка
MSH#	I/O	Разделяемые данные
MIH#	O	Запрет памяти
MBR#	I/O	Запрос шины
MBG#	I	Грант шины
MBB#	I/O	Занятости шины
IR[3] — IR[0]	I	Уровень прерывания
MID[3] — MID[0]	I	Идентификатор модуля
AERR#	O	Асинхронная ошибка
RSTIN#	I	Сброс МП

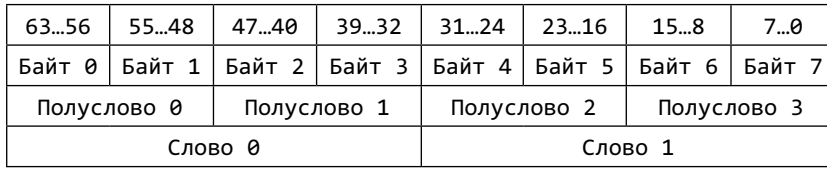


Рис. П2.1. Размещение разрядов, байтов, полуслов и слов на шине MBus

MAS# — строб адреса памяти. Этот сигнал устанавливается ведущим устройством шины во время самого первого такта шинной транзакции. Этот такт называется адресным тактом или адресной фазой транзакции. Все другие такты указываются относительно MAS#. Например, A+3 указывает на третий такт после выдачи MAS#, а сам такт выдачи MAS# есть A+0.

MRDY# — готовность транзакции MBus. Этот сигнал является одним из трех разрядов, используемых для кодирования состояния транзакции, как показано в табл. П2.2. Только когда установлен MRDY#, передаются значащие данные. Три разряда состояния (MRDY#, MRTY# и MERR#) обычно выдаются адресуемым (подчиненным) устройством.

MRTY# — повтор транзакции MBus. Этот сигнал является одним из трех разрядов, используемых для кодирования состояния транзакции, как показано в табл. П2.2. Только когда установлен MRTY#, подчиненное устройство хочет, чтобы ведущее устройство немедленно удалило текущую транзакцию и начало заново. Ведущее устройство освободит шину после такого уведомления о перезапуске.

MERR# — ошибка транзакции MBus. Этот сигнал является одним из трех разрядов, используемых для кодирования состояния транзакции. Установка сигнала MERR# показывает, что произошла ошибка шины.

Таблица П2.2. Кодировка разрядов состояния транзакции

MERR#	MRDY#	MRTY#	Значение
H	H	H	Пустой такт
H	H	L	Освободи и повтори
H	L	H	Значимая передача данных
H	L	L	Резерв
L	H	H	Ошибка 1 — ошибка шины
L	H	L	Ошибка 2 — таймер
L	L	H	Ошибка 3 — неисправимая
L	L	L	Повтори

Примечание: H — высокий уровень, L — низкий уровень.

Если выдается любой тип подтверждения, отличный от значимой передачи данных, тогда такт, в котором он выдается, является последним тактом транзакции независимо от того, сколько еще тактов подтверждения должно бы быть выдано.

MBR# — запрос шины MBus. Этот сигнал выдается ведущим устройством на шину MBus, чтобы занять ее. Имеется по одному сигналу MBR# от каждого ведущего устройства, поступающему в арбитр шины MBus.

MBG# — грант шины MBus. Этот сигнал выдается внешним арбитром, когда шина предоставляется определенному ведущему устройству. Имеется по одному сигналу MBG# на каждое ведущее устройство.

MBB# — занятость шины MBus. Этот сигнал выдается во время всей передачи, начиная с выдачи MAS# и до выдачи последнего MRDY# или первого другого оповещения, которое заканчивает передачу (такого, как оповещение об ошибке).

MIH# — запрет памяти. Этот сигнал имеется только в MBus-модулях с когерентными кэшами. Он выдается хозяином блока кэша после получения его адреса. Сигнал информирует оперативную память о том, что текущая транзакция типа «когерентное чтение» или «когерентное чтение и гашение» должна быть проигнорирована.

MSH# — разделяемый блок кэша. Этот сигнал присутствует только в MBus-модулях с когерентными кэшами. Когда на шине появляется транзакция типа «когерентное чтение», каждый модуль на MBus должен поискать адресуемый блок в своем справочнике кэша. Если обнаруживается значащая разделяемая копия, то должен быть выдан сигнал MSH#.

RSTIN# — входной сигнал сброса модуля. Этот сигнал должен устанавливать всю логику в модуле в исходное состояние.

AERR# — сигнал обнаружения асинхронной ошибки модуля. Этот сигнал выдается модулем как сообщение, что им была обнаружена внутренняя ошибка. Микропроцессор перейдет в режим ошибки при входе в любую исключительную ситуацию с запретом прерываний. Находясь в режиме ошибки, микропроцессор автоматически выполнит сброс по таймеру.

IRL[3:0] — уровень запроса прерывания. Эти сигналы передают уровень запроса прерывания в микропроцессор. Каждый процессорный модуль получает соответствующий набор IRL[3:0].

ID[3:0] — идентификатор модуля. Эти сигналы передают идентификатор модуля. Идентификатор выдается микропроцессором в течение адресной фазы каждой транзакции в разрядах MID[63:60]. Идентификатор также используется для задания отдельного адресного пространства при идентификации, инициализации и конфигурации модуля.

Рассмотрим структуру командного слова MBus, передаваемого по линиям MAD[63:0]. Командное слово выдается ведущим устройством для ведомого устройства в адресной фазе транзакции MBus, когда устанавливается сигнал MAS#. В фазе передачи данных транзакции MBus сигналы MAD[61:0] используются для передачи данных. На рис. П2.2 представлена структура командного слова.

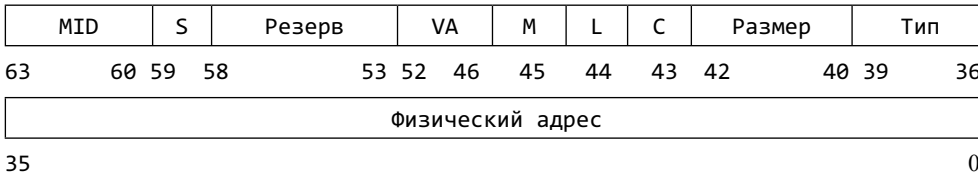


Рис. П2.2. Структура командного слова MBus

MID — идентификатор модуля MBus. Это поле содержит значение ID[3:0] ведущего устройства в этой транзакции. MID выдается всем MBus-модулям и позволяет ведомым устройствам повторно подсоединяться после выдачи подтверждения «освободи и повтори».

S — признак обращения супервизора. Сигнал устанавливается, чтобы указать, что MBus-транзакция была начата процессором в режиме супервизора. Этот рекомендательный разряд не используется в MBus-транзакции, но может использоваться ведомыми устройствами.

VA — старшие разряды виртуального адреса. Микропроцессор не использует эти разряды, так как использует кэш с физической адресацией.

M — признак режима Boot. Микропроцессор не использует этот разряд.

L — признак блокировки. Он предназначен для блокировки ведомого устройства конкретным ведущим устройством. Шина блокируется путем удержания сигнала MBB# в течение всей операции. Микропроцессор устанавливает признак блокировки, когда выполняет неразрывные операции.

C — признак кэшируемости. Когда этот признак установлен, он отмечает кэшируемые данные, то есть обращение должно выполняться в кэш.

Размер — размер операции в байтах согласно табл. П2.3.

Таблица П2.3. Кодировка размера транзакции

Размер[2]	Размер[1]	Размер[0]	Размер транзакции
L	L	L	Байт
L	L	H	Полуслово (2 байта)
L	H	L	Слово (4 байта)
L	H	H	Двойное слово (8 байт)
H	L	L	Группа 16 байт
H	L	H	Группа 32 байта
H	H	L	Группа 64 байта
H	H	H	Группа 128 байт

Примечание: H — высокий уровень, L — низкий уровень.

Тип — указывает тип операции согласно табл. П2.4.

Таблица П2.4. Кодировка типа транзакции

Тип[3]	Тип[2]	Тип[1]	Тип[0]	Размер	Тип транзакции
L	L	L	L	Любой	Запись
L	L	L	H	Любой	Чтение
L	L	H	L	32 байта	Когерентное гашение
L	L	H	H	32 байта	Когерентное чтение
L	H	L	L	32 байта	Когерентные запись и гашение
L	H	L	H	32 байта	Когерентные чтение и гашение
L	H	H	L		Резерв
L	H	H	H		Резерв
H	x	x	x		Резерв

Примечание: H — высокий уровень, L — низкий уровень.

Физический адрес — 36 разрядов физического адреса.

Рассмотрим временные диаграммы транзакций MBus.

Некэшируемые транзакции используются для ввода-вывода и инициализации процессора. Для таких обращений не производится просмотра шины (snooping). Такие транзакции используются для считывания и записи.

Некэшируемые операции чтения (для кодов и данных) используют READ-транзакцию. Для этого используется однотактная передача. Размер транзакции может составлять байт, полуслово, слово или двойное слово. Операция чтения 8 байт показана на рис. П2.3.

Ведущее устройство выдает адрес, информацию о состоянии и устанавливает на один такт сигнал MAS#. Ведущее устройство также устанавливает сигнал занятости шины MBB# на все время выполнения транзакции. Сигнал MBB# удерживается высоким 1/2 такта перед освобождением.

Адресуемое ведомое устройство выдает данные и устанавливает сигнал MRDY# на один такт.

Временная диаграмма представлена в предположении, что ведущее устройство уже прошло фазу арбитража и получило сигнал разрешения работы на шине MBG#.

Групповое чтение используется в операциях пересылки «память — память». Операция чтения 32 байт показана на рис. П2.4.

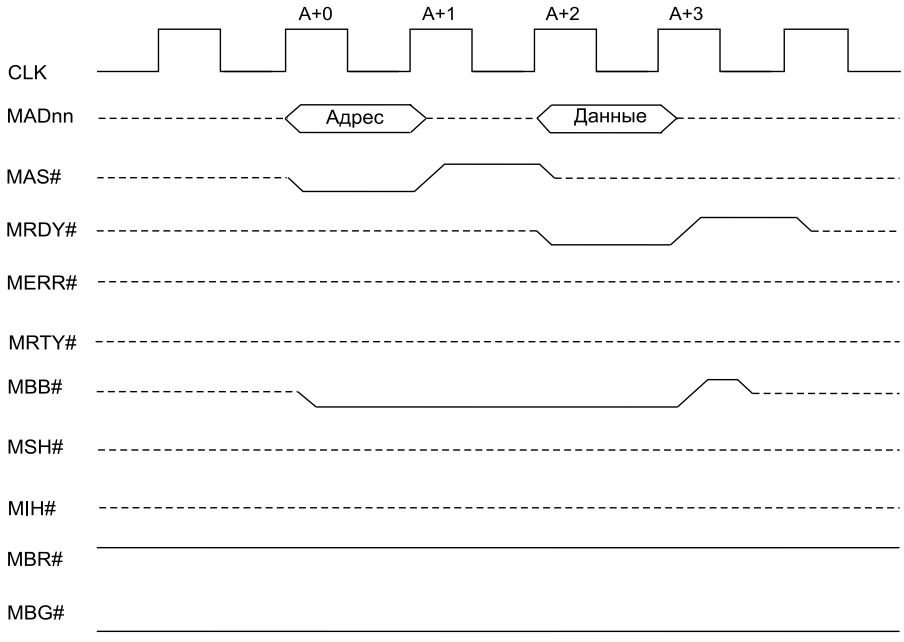


Рис. П2.3. Временная диаграмма операции чтения MBus

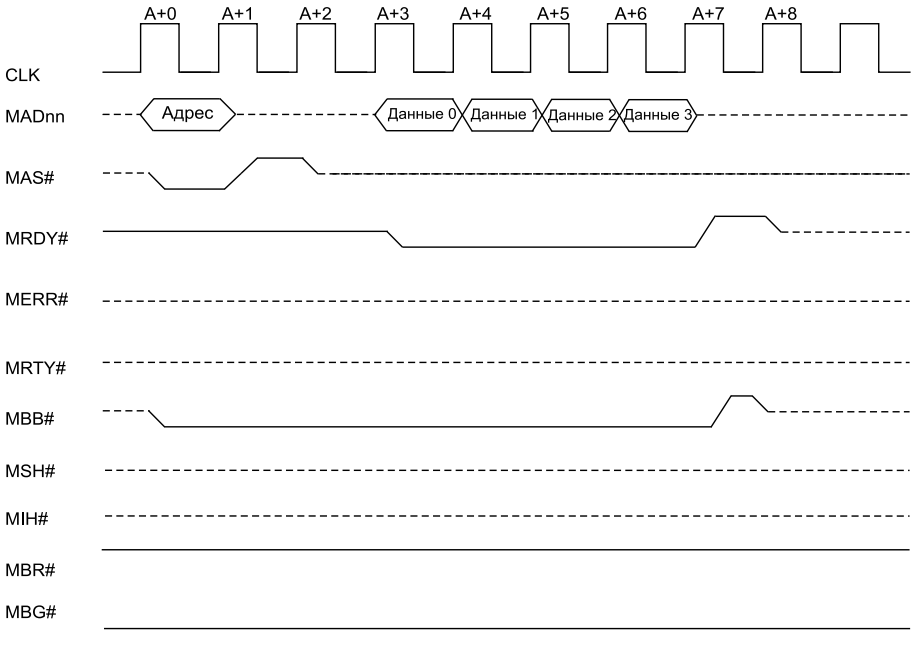


Рис. П2.4. Временная диаграмма группового чтения MBus

Операция чтения может быть выполнена при любом размере передачи данных, определенном разрядами «размер». Транзакции чтения обеспечивают считывание адресуемого слова первым в блоке передаваемых данных. Транзакции, читающие меньше 8 байт, будут иметь неопределенные значения в неиспользуемых байтах.

Адресуемое ведомое устройство выдает данные и устанавливает сигнал MRDY# на один такт для каждой передачи 64-разрядного слова. Данные могут выдаваться подряд или с некоторой задержкой между выдачами отдельных слов в зависимости от возможности ведомого устройства.

Временная диаграмма представлена в предположении, что ведущее устройство уже прошло фазу арбитража и получило сигнал разрешения работы на шине MBG#.

Некэшируемые операции записи используют WRITE-транзакцию. Для этого используется однотактная передача. Размер транзакции может составлять байт, полуслово, слово или двойное слово.

Временная диаграмма выполнения операции записи 8 байт показана на рис. П2.5.

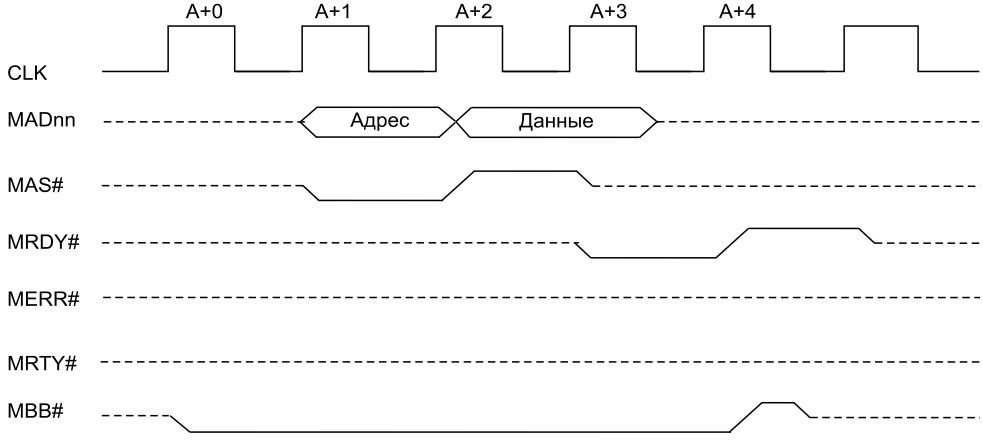


Рис. П2.5. Временная диаграмма операции записи MBus

Ведущее устройство выдает адрес, информацию о состоянии и устанавливает на один такт сигнал MAS#. Затем ведущее устройство по тем же шинам выдает данные.

Адресуемое ведомое устройство устанавливает сигнал MRDY# на один такт, после чего ведущее устройство снимает сигнал занятости шины MBB#.

Групповая запись используется при выполнении операций пересылки «память — память».

Операция записи 32 байт показана на рис. П2.6. Ведущее устройство выдает данные сразу после адресной фазы и после получения каждого сигнала MRDY#.

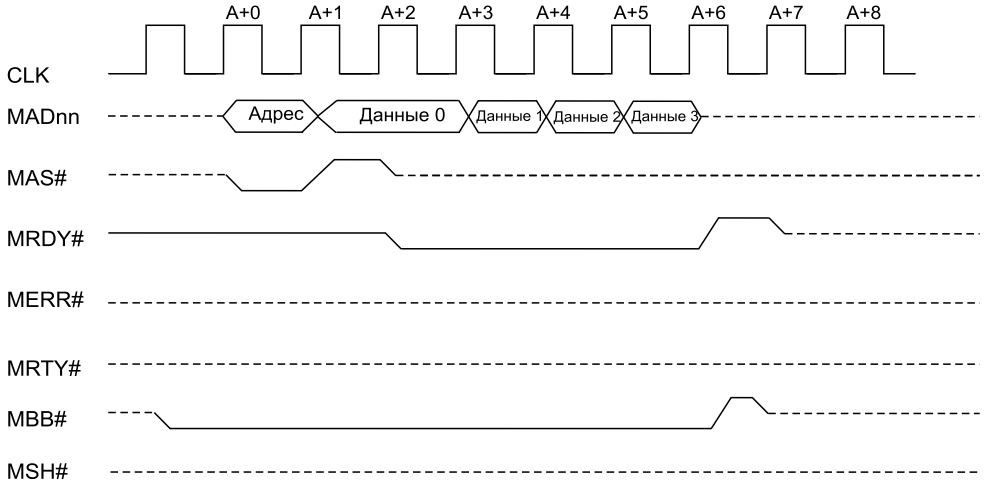


Рис. П2.6. Временная диаграмма групповой записи MBus

Адресуемое ведомое устройство выдает сигнал MRDY# после каждого получения данных.

Когерентные транзакции используются для кэшируемых обращений. Транзакции когерентного чтения используются для чтения данных из текущего владельца, которым могут быть память или другой кэш.

Когерентное чтение используется при промахах во всех кэшах данных и кэше команд процессора. Если другой кэш является владельцем данных, он выставляет в ответ сигнал MIH# и выдает данные.

Транзакции когерентного чтения с гашением используются для того, чтобы прочитать данные из текущего владельца для выполнения операции их замещения. Использование этой транзакции предполагает, что данные будут модифицированы при получении в процессоре. Текущий владелец должен сбросить у себя признак владелец, и все копии этого блока данных во всех кэшах должны быть погашены. После завершения транзакции микропроцессор будет исключительным владельцем данного блока данных.

Первое двойное слово должно быть начальным адресом транзакции. Двойные слова из памяти должны поступать в составе блока из 32 байт. Процессор использует данные сразу же при их появлении.

Операция когерентного чтения 32 байт показана на рис. П2.7.

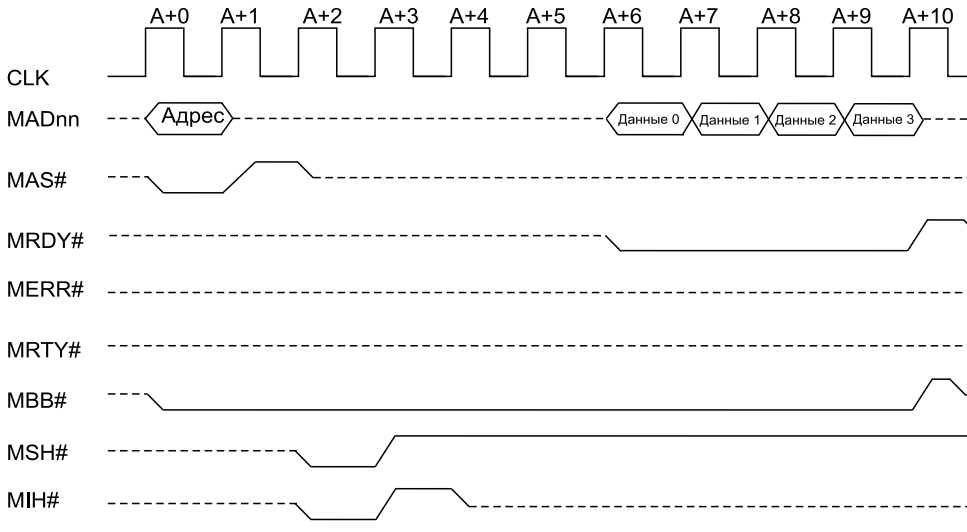


Рис. П2.7. Временная диаграмма когерентного чтения MBus

Операция когерентного чтения является групповой операцией обслуживания согласованного состояния кэш-памятей в многопроцессорной системе. Когерентные чтения выполняются с данными, которые должны быть размещены во внутренних и внешних кэшах микропроцессора. Участниками транзакции когерентного чтения являются запрашивающий кэш и другие кэши, которые проверяют наличие адресуемых данных (snoop) и память. Возможны три сценария проверки.

1. Проверяющий кэш, который не имеет запрашиваемого блока, просто игнорирует транзакцию.

2. Проверяющий кэш, который имеет копию запрашиваемого блока, но не является его владельцем, просто вставляет сигнал MSH# на один такт (в интервале тактов от A+2 до A+7). Он также отметит эту копию как разделяемую, если еще не сделал этого.

3. Проверяющий кэш, который является владельцем запрашиваемого блока, выставит сигналы MSH# и MIH# на один такт (в интервале тактов от A+2 до A+7) и начнет выдавать запрашиваемые данные не

раньше, чем через четыре такта после выдачи сигнала MIH#. Если его собственная копия была помечена как исключительная, она будет переведена в разделяемые.

При получении блока данных запрашивающий кэш пометит блок как исключительный, если на шине не был выставлен сигнал MSH#.

Временная диаграмма выполнения операции когерентного гашения показана на рис. П2.8.

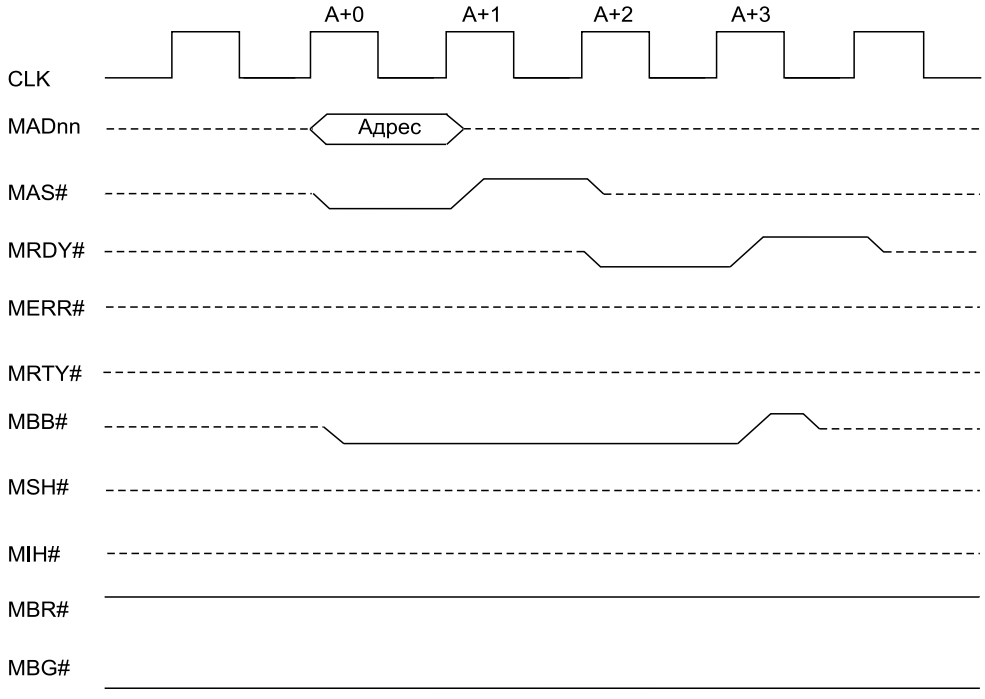


Рис. П2.8. Временная диаграмма операции когерентного гашения MBus

Транзакция когерентного гашения выдается, когда выполняется запись в разделяемый блок данных в своем кэше. Перед тем как запись реально будет выполнена, все другие кэши в системе должны иметь погашенными свои копии адресуемого блока.

Операция когерентного гашения выполняется над блоком размером 32 байта. Ведущее устройство выдает адрес, информацию о состоянии и устанавливает на такт сигнал MAS#. Ведущее устройство устанавливает сигнал занятости шины MBB# на все время выполнения транзакции.

Память отвечает за сигнал подтверждения в этой транзакции. Она выставляет сигнал MRDY# в такте A+2 или позже.

Транзакция когерентного чтения и гашения выдается, когда выполняется запись в блок данных, отсутствующий в своем кэше. Транзакция когерентного чтения проверяет все кэши в системе. Если имеется совпадение адреса, но кэш не является владельцем блока, то кэш сразу погасит этот блок. Если имеется совпадение адреса и кэш является владельцем этого блока, то кэш выставит сигнал MIH# и выдаст требуемые данные. После того как данные будут выданы, кэш погасит копию этого блока. Сигнал MSH# не выдается во время транзакции когерентного чтения и гашения.

Транзакция когерентной записи и гашения выдается при записи только в память. При этом адресуемый блок данных должен быть погашен во всех кэшах системы. Проверке подлежат все они. Если имеются совпадения адреса, то копии адресуемого блока гасятся во всех кэшах. Сигналы MIH# и MSH# не выставляются на шине в течение этой транзакции.

ПРИЛОЖЕНИЕ 3
Общая характеристика операций микропроцессора «Эльбрус»

Аппаратура преобразует загруженное командное слово в набор поступающих на исполнение операций, имеющих в общем случае вид, представленный на рис. П3.1.

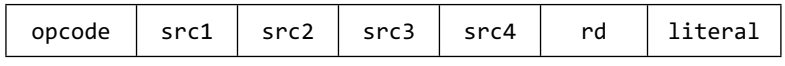


Рис. П.3.1. Структура команды внутреннего представления: opcode — тип операции; src1, src2, src3, src4 — адреса операндов или короткие константы с признаками значимости; rd — адрес результата; literal — константа из кода программы

Количество полей категории src зависит от значения opcode:

- признак значимости в одном поле src1 соответствует одноместной операции;
- признак значимости в полях src1 и src2 соответствует двухместной операции;
- признак значимости в полях src1, src2, src3 соответствует трехместной комбинированной операции или двухместной операции с одним из операндов формата квадрослово;
- признак значимости в полях src1, src2, src4 соответствует операции записи;
- признак значимости в полях src1, src2, src3, src4 соответствует операции записи с одним из операндов формата квадрослово, для считывания которого необходимы два порта регистрового файла.

Записываемое значение в операциях записи задается адресом операнда src4. Он подается на отдельный порт чтения в регистровом файле с задержкой в 1 такт, пока вычисляется адрес обращения в память в адресном сумматоре исполнительного устройства.

Целочисленные и вещественные арифметические операции, логические операции и операции сдвига используют в зависимости от типа операции один, два или три операнда и формируют результат в виде числового значения, набора флагов или предиката (операции сравнения). В качестве

операндов могут быть числовые значения с форматом слово или двойное слово из регистрового файла или константы, представленные в коде операции. Константы могут кодироваться в поле адреса операнда или как ссылка на дополнительные один или два слога литерала для задания констант форматов 32 и 64 соответственно. Результаты в виде числового значения или флагов помещаются в регистровый файл, в виде предикатов — в предикатный файл. Определены также вещественные арифметические операции с расширенным вещественным 80-разрядным форматом.

Целочисленные и вещественные операции могут использоваться в комбинированных трехадресных операциях. Комбинированная операция состоит из двух последовательно выполняемых операций. Первая операция выполняется с первым и вторым операндами, вторая — с результатом первой операции и третьим операндом. Результат второй операции заносится по адресу назначения в регистровый файл.

Операции над упакованными значениями используют операнды и формируют результат в формате двойного слова. В нем могут быть представлены байты и полуслова (только для операций над целыми значениями), одинарные и двойные слова. Операции над упакованными значениями определены для целых и вещественных данных.

Операции над предикатами позволяют считать предикаты из предикатного файла, выполнить функцию «логическое И» над двумя предикатами или их инверсиями (что реализует наиболее употребительный набор бинарных логических функций), записать вычисленный предикат в предикатный файл или направить его на управление условным выполнением арифметических операций или операций передачи управления.

Операции обращения в память выполняют считывание и запись. Определены следующие форматы значений: байт, полуслово, одинарное слово, двойное слово, квадрослово.

Обращения в память сопровождаются спецификатором, задаваемым явно или формируемым по умолчанию и определяющим режим обращения. Спецификатор по умолчанию задает обращение в память с размещением данных в кэшах микропроцессора. Спецификатор, явно определенный в дополнительном слоге, позволяет задать обращения в память, минуя кэши микропроцессора, без трансляции виртуального адреса в физический.

Адрес обращения в память формируется как результат индексации дескриптора. Первый операнд задает дескриптор со значением в формате

квадро, второй операнд задает индекс со значением в формате одинарного слова. Третий операнд в операциях записи — это записываемое в память значение.

Обращения в память к глобалам и командам выполняется с использованием аппаратных регистров, хранящих соответствующие дескрипторы. В этих случаях первый и второй операнды — это индексы в формате одинарного слова.

В операциях обращения в память к массиву адреса дескриптора и индекса задаются в устройстве обращения к массивам, а литеральное смещение задается в дополнительном литеральном слоге.

В операциях обращения в память в незащищенное пространство дескриптор и индекс — это операнды в формате двойного слова.

Операции преобразования адресных объектов выполняют индексацию заданного дескриптора и помещают результат в регистровый файл.

Операции с тегом выполняют считывание и установку внешнего тега для заданного значения.

Операции обращения к управляющим регистрам выполняют чтение и запись заданных управляющих регистров.

Операции подготовки передачи управления формируют значение указателя команды для предстоящей передачи управления и выполняют подкачку кода в заданный регистр подготовки передачи управления.

Операции передачи управления выполняют переключение дешифрации команд на заданное подготовленное направление и формируют новое регистровое окно в стеке процедур и стеке связующей информации (для процедурных передач управления).

Операции поддержки наложений цикла выполняют установку и продвижение управляющих регистров выполнения цикла.

Операции над кэшами позволяют очистить заданный кэш или его строку, а также прочитать значение заданной строки кэша.

В определенных динамических ситуациях микропроцессор выполняет некоторые действия, не описанные в программном коде (откачка/подкачка регистрового файла, вход в прерывание). В этих случаях формируются соответствующие аппаратные команды.

Таким образом, параллельная структура микропроцессорного ядра «Эльбрус» позволяет в различных сочетаниях дешифровать в каждом такте:

- до десяти арифметико-логических операций;
- до четырех операций обращений в память (до четырех — на чтение, до двух — на запись);
- до трех операций обработки булевых значений;
- до четырех операций формирования литералов;
- условную передачу управления;
- операцию приращения счетчика цикла;
- до двух групповых операций подкачки элементов массива с продвижением адресов;
- до четырех операций считывания подкачанных элементов массива;
- до шести операций вычисления условий, управляющих условным выполнением команд.

В результате в одной ШК может содержаться до 23 операций для программы, обрабатывающей массивы, и до 14 операций для скалярных программ.

ПРИЛОЖЕНИЕ 4

Распределение слогов упакованной ШК по полям исполнительной ШК (схема рассеивания)

Распределение слогов по полям дешифруемой широкой команды выполняется с помощью шкал, имеющихся в заголовке и слоге SS (Stubs syllable — слог коротких операций). Кроме битов шкалы используется и другая информация из заголовка о структуре ШК.

По особенностям использования данных заголовка при дешифрации ШК все слоги делятся на группы. Вначале заполняются поля слогов первой группы, затем — второй и третьей групп.

Вторая группа слогов содержит слог команды управления С1 и пять слогов, из которых выделяются десять полусловных слогов: ALE0, ALE1, ALE3, ALE4, AA0, AA1, AA2, AAЗ, AA4, AA5. (Полуслоги ALE являются расширением соответствующих слогов АЛК, полуслоги АА кодируют операции, которые пересылают в регистровый файл элементы массивов, предварительно подкачанные в буфер.)

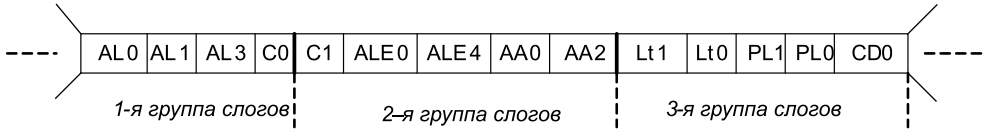
Третья группа содержит следующие слоги (справа налево): CD0, CD1, CD2 (слоги условного выполнения), PL0, PL1, PL2 (слоги предикатных логических каналов), Lt0, Ltl, Lt2, Lt3 (слоги литералов). Слоги третьей группы упакованы справа налево в отличие от слогов первой и второй групп.

Принцип рассеивания слогов иллюстрируется рис. П4.1–П4.3. В качестве примера приведена упакованная широкая команда, выделенная из строки программного кода и содержащая слоги всех групп (рис. П4.1, *а*).

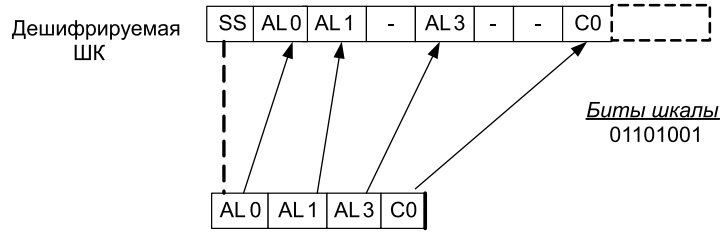
Рассеивание слогов первой группы выполняется с помощью 8-разрядной шкалы из заголовка. Состояние бита шкалы определяет наличие (единица) или отсутствие (нуль) данного слога в дешифрованной команде. Соответствие битов шкалы и слогов — позиционное слева направо. Принцип рассеивания слогов первой группы показан на примере наличия в ШК четырех слогов (рис. П4.1, *б*).

Рассеивание слогов второй группы по полям дешифруемой ШК осуществляется с помощью 10-разрядной шкалы заголовка по аналогии с рассмотренным ранее и 4-разрядного поля middle слога SS, каждый разряд которого управляет распределением пары полуслогов АА. Например, левый бит определяет наличие в команде АА0, АА2, следующий бит — АА0, ААЗ

полуслогов и т. п. (рис. П4.2). В верхней части формата дешифруемой ШК приведены названия незаполненных полей.



а) упакованная ШК



б) принцип рассеивания слогов первой группы

Рис. П4.1. Принцип рассеивания слогов первой группы: *а* — упакованная ШК; *б* — принцип рассеивания слогов первой группы

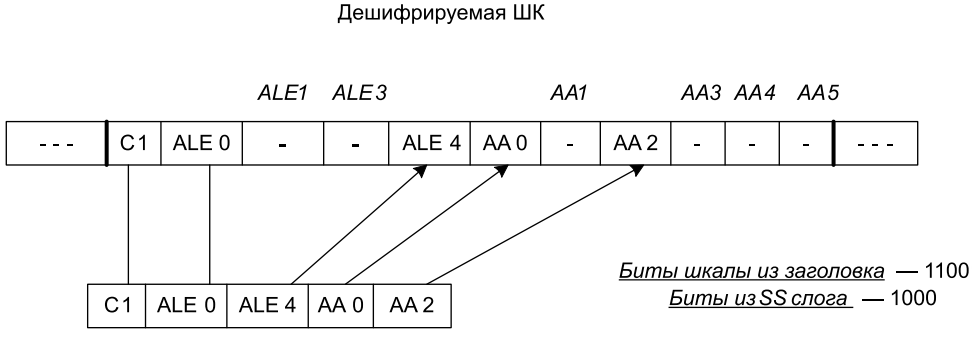


Рис. П4.2. Принцип рассеивания слогов второй группы

Принцип рассеивания слогов третьей группы (рис. П4.3) имеет свои особенности. Слоги CD, как и PL, функционально не различаются между собой, поэтому если для выражения некоторого действия достаточно одного слога CD, то в ШК присутствует только CD0, если необходимы два слога CD, то в команде должны быть слоги CD0 и CD1, и так далее до трех слогов

CD. Все сказанное относится и к слогам PL. В результате вместо, например, слога CD0 не может быть поставлен слог CD1 или на месте слога PL1 оказаться слог PL2.

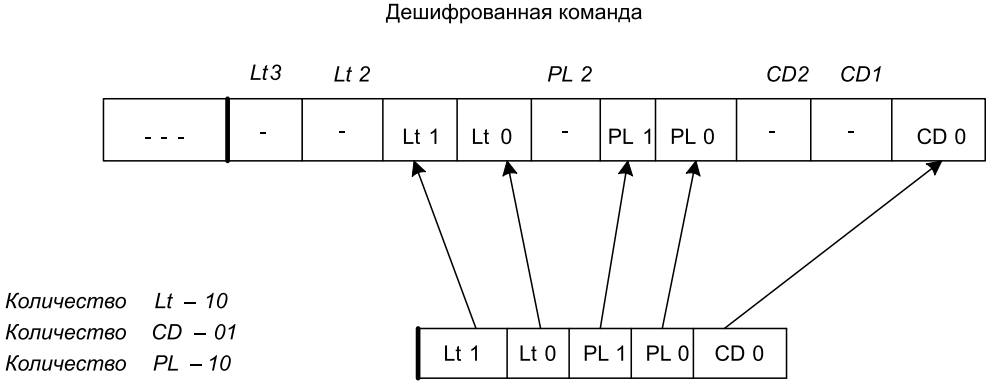


Рис. П4.3. Принцип рассеивания слогов третьей группы

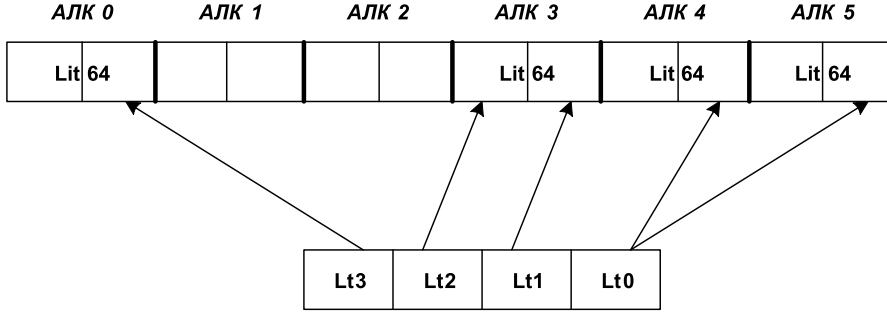
Эта особенность слогов CD и PL отразилась на принципе их рассеивания, которое основано на использовании четырех битов заголовка, два старших разряда которого задают количество слогов CD, а код следующей пары битов — число слогов LP в упакованной широкой команде. Принцип заполнения полей дешифруемой ШК слогами третьей группы, кроме слогов литеральных констант, показан на рис. П4.3.

Что касается слогов литералов, их рассеивание по каналам АЛУ осуществляется во втором такте дешифрации ШК. Для каждого АЛК в дешифруемой ШК предусмотрено 64-разрядное поле, в которое может быть записан литерал длиной в полуслово, слово или двойное слово. Распределение литералов основано на анализе адресов второго операнда A2 (Srs2) операций арифметических каналов. Эти адреса могут представлять собой номера регистров РгФ или ссылки на литеральные константы Lt0...Lt3. Каждый слог может содержать либо 32-разрядное знаковое значение, либо любую из половин 64-разрядного значения. Следовательно, 64-разрядный литерал размещается в двух смежных слогах LTS.

Признаком использования литерала в качестве второго операнда является код старшей тетрады 1101 8-разрядного адреса A2. Последующими битами адреса кодируются длина литерала (16, 32 или 64 разряда), признак старшей или младшей половины слова (для 16-разрядной константы) и номер литерала (два младших разряда адреса). Кодировка адреса и пример рассеивания литералов для операций АЛК с номерами 0, 3, 4

и 5 показаны на рис. П4.4. Признак ссылки на литералы — код старшей тетрады адреса А2 1101.

а) кодировка длин литералов в поле адреса второго операнда



б) рассеивание литералов по АЛК

(адреса А 2 в АЛК: 0 — 1101 1011; 3— 1101 1101; 4 — 1101 0100; 5 — 1101 0000)

Рис. П4.4. Кодировка длин и рассеивание литералов: *а* — кодировка длин литералов в поле адреса второго операнда; *б* — рассеивание литералов по АЛК

Код младшей тетрады адреса А2 АЛК0 1011. Код ее старших разрядов (10) соответствует словному литералу, следующая пара разрядов (11) — номер слога константы — Lt3. Таким образом, в качестве второго операнда в АЛК0 передается 32-разрядная константа Lt3.

Адрес второго операнда в АЛК3 содержит код младшей тетрады 1101. Следовательно, он адресует к константе двухсловной длины (код старших

разрядов тетрады 11). Двухсловный литерал (64-разрядная константа) занимает два смежных слога, поэтому в АЛК при рассеивании выдаются слог, номер которого задан младшими разрядами адреса, и следующий слог.

Адреса А2 каналов 4 и 5 указывают на полусловные литералы, поскольку старший разряд младшей тетрады равен нулю. Младшими двумя разрядами (00) выбирается слог LT0, а третий справа разряд содержит признак половины слова. В результате в АЛК4 в качестве второго операнда поступит 16-разрядный код старшей половины слога LTS0, а в канал 5 — 16-разрядный код младшей половины этого же слога.

Таким образом, после рассеивания литералов на этапе конвейера БА (В) дешифрация широкой команды заканчивается.

ПРИЛОЖЕНИЕ 5
Организация стеков

Стек процедур. Состоит из двух частей, одна из которых располагается в регистровом файле, а для другой части отводится участок памяти на случай переполнения РгФ.

Регистровая часть стека процедур (рис. П5.1), называемая аппаратурной вершиной стека (АВС), описывается дескриптором текущего окна ДО (Current window descriptor, WD) и указателем аппаратурной вершины стека УВС (Procedure stack hardware top pointer, PSHTP).

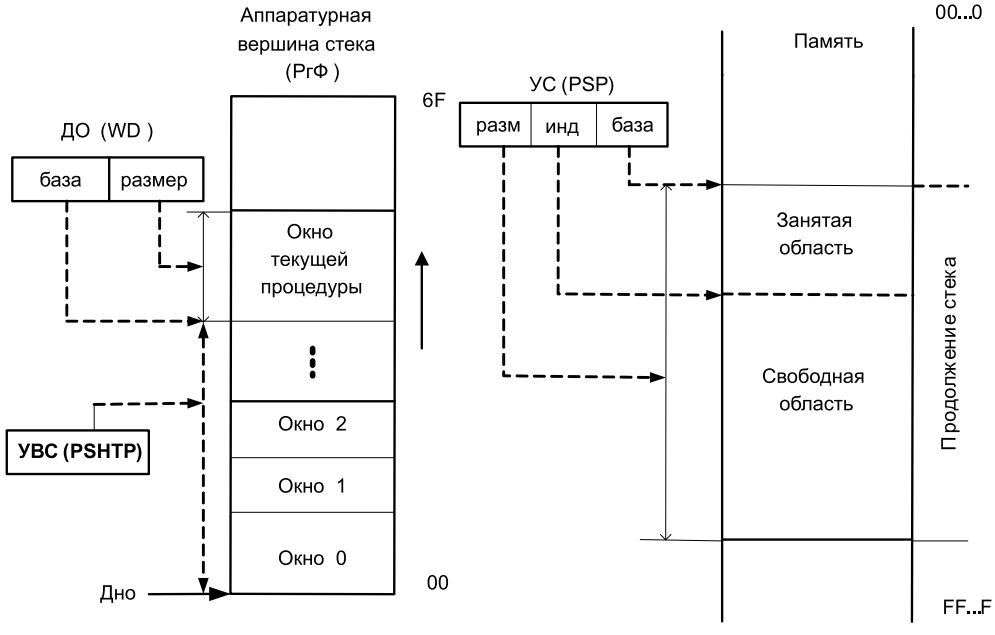


Рис. П5.1. Принцип организации процедурного стека

Поле базы дескриптора окна (WD.base) хранит физический адрес первого регистра окна, а поле размера (WD.size) — количество регистров в нем. Окно выровнено по границам квадрослова (четыре младших разряда адреса базы в терминах байтовой адресации и количества установлены в нуль, то есть размер окна кратен паре регистров РгФ).

Указатель аппаратурной вершины стека УВС (PSHTP) содержит в своем поле УВС.инд (PSHTP.ind) длину регистровой части стека процедур,

исключая текущее окно, то есть число занятых регистров под текущим окном. В этом случае значение ind можно рассматривать как индекс регистра дна стека процедур. В данном случае для определения дна стека обычное индексирование базы (сложение) надо заменить вычитанием, поэтому первый регистр, с которого производится откачка, размещается в РгФ по физическому адресу, равному WD.base — PSHTP.ind.

Заметим, что значение индекса ind может оказаться отрицательным (при возврате). Отрицательное число определяет количество регистров, необходимое для подкачки в РгФ.

Вторая часть стека процедур, размещаемая в памяти, описывается указателем стека процедур УС (Procedure stack pointer, PSP), который содержит виртуальный адрес базы этой части стека и ее размер. Поскольку с этой областью памяти связаны откачка и подкачка АВС, то необходимо указать индекс первого свободного байта, с которого начинается запись откачиваемых с РгФ данных. Он задается в поле указателя стека (PSP.ind), чем и определяется граница между свободной и занятой частями области стека процедур, размещенной в памяти.

Стек пользователя. Стек пользователя организуется с помощью регистра базы стека РгБСП (User Stack Base Register, USBR) и дескриптора ДСП (User Stack Descriptor, USD).

Регистр базы стека содержит виртуальный адрес дна текущего стека пользователя (стек пользователя «опрокинут»). Дескриптор стека пользователя описывает свободное пространство памяти. Его состав зависит от признака режима обслуживания стека пользователя, защищенного или незащищенного, который содержится в дескрипторе.

Принцип организации незащищенного стека пользователя приведен на рис. П5.2. В левой части рисунка, демонстрирующей исходное состояние стека, показаны регистр РгБСП (USBR), хранящий адрес базы стека пользователя (дна стека), дескриптор ДСП (USD) с полями базы (Б) и размера (Р), фреймы (выделенные области памяти стека по запросам процедур) и свободная область стека.

Дескриптор незащищенного стека пользователя хранит виртуальный адрес базы свободной области и ее размер. Кроме этого, дескриптор содержит флаг защищенного режима, который в данном случае сброшен.

По операции CALL вызывается новая процедура, причем состояние стека не изменяется. В то же время по операции GETSP, в которой указывается

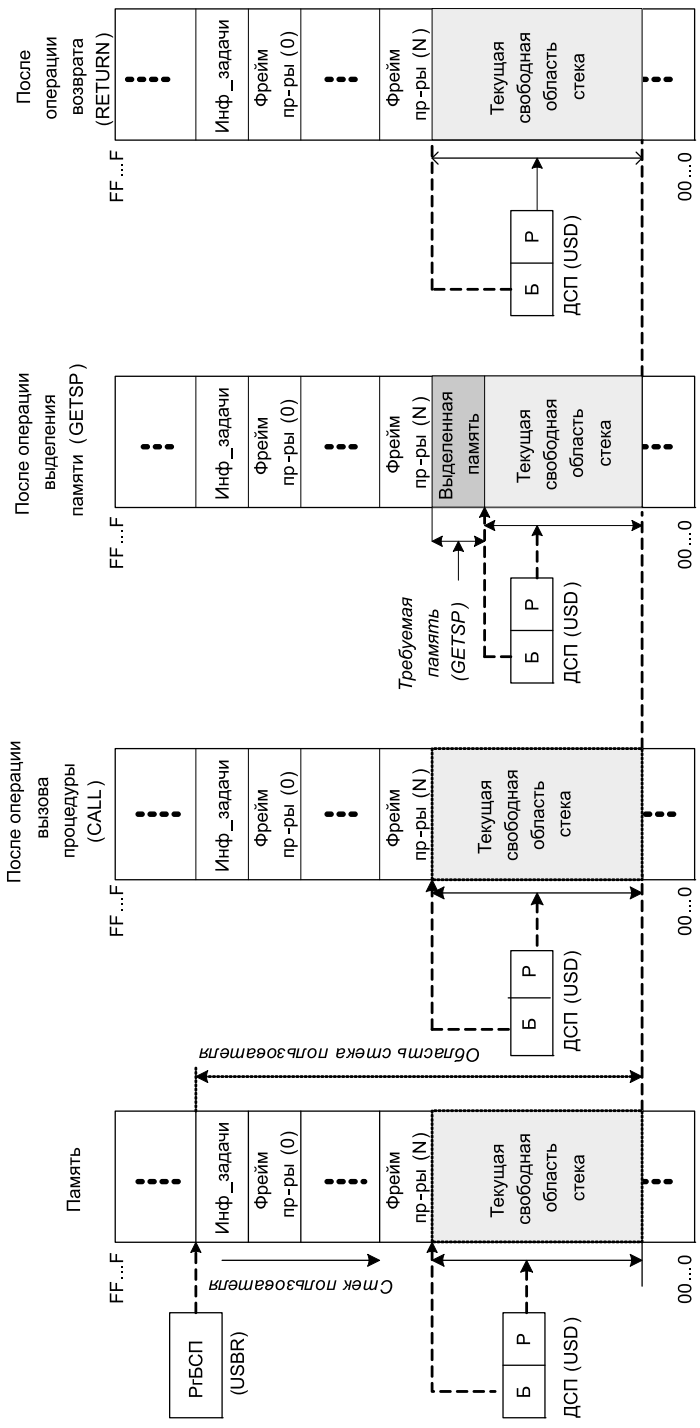


Рис. П5.2. Принцип организации незащищенного стека пользователя

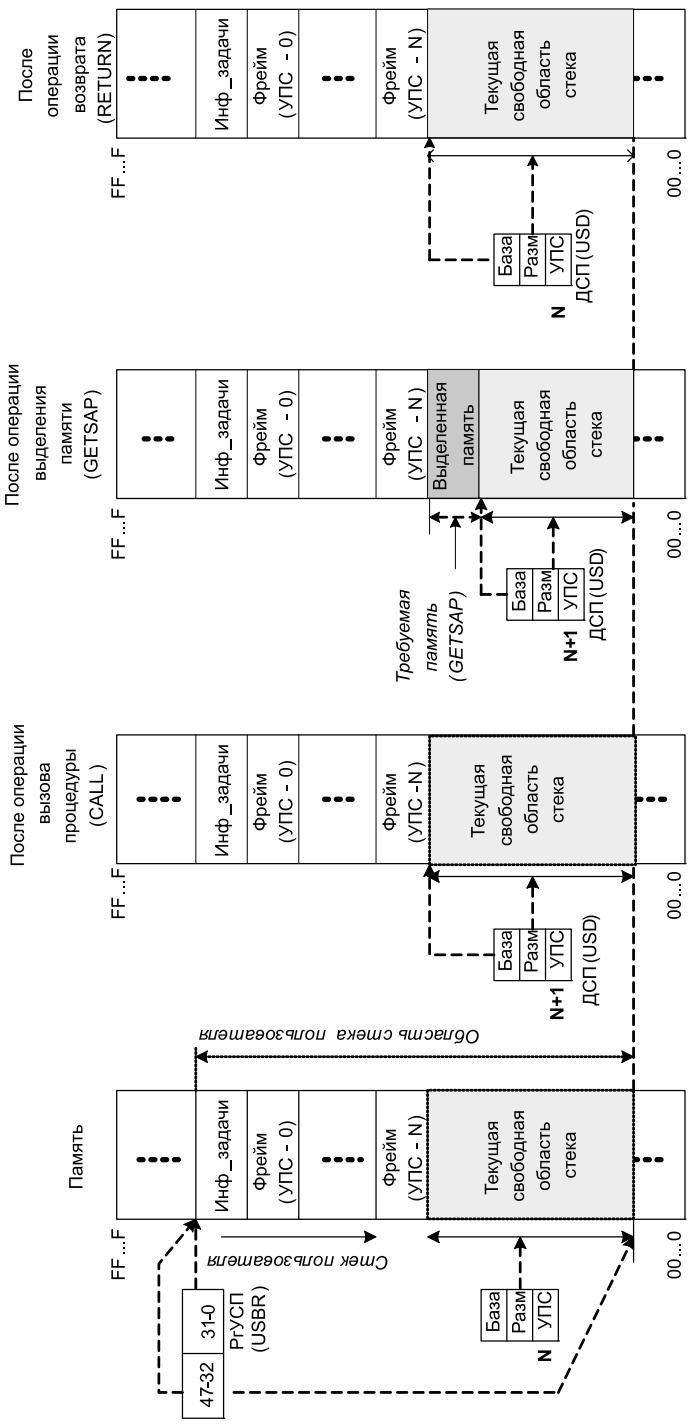


Рис. П5.3. Принцип организации защищенного стека пользователя

требуемый объем памяти, выделяется новый фрейм и формируется дескриптор на оставшуюся свободную область стека.

По завершении процедуры выполняется возврат на прерванную процедуру по команде RETURN, при котором выделенная область передается в ресурс свободной памяти. Делается это путем модифицирования дескриптора ДСП, переставляющего указатель базы свободной области.

В случае применения защищенного стека пользователя (рис. П5.3) старшие разряды виртуального адреса на регистре РгБСП (47:32) используются как старшие разряды всех виртуальных адресов текущего стека, что отображено дополнительными стрелками от поля «47-32» указателя РгУСП (USBR). Для удобства описания вводят фиктивный регистр, содержащий это общее базовое значение, называемый регистром базы стека SBR (Stack Base Register), а все указатели и дескрипторы стека пользователя вместо абсолютного адреса содержат 32-разрядные индексы относительно SBR. Следовательно, виртуальный адрес для обращения в стек пользователя образуется путем добавления к общим старшим разрядам (47:32) значения индекса.

Дескриптор защищенного стека содержит установленный флаг защищенного режима, относительную базу, с которой начинается свободная область в стеке пользователя, и ее размер. Кроме этого, в нем предусмотрено поле для динамического уровня текущей процедуры (УПС) — ее номера в стеке процедур (procedure stack level, psl), характеризующего глубину стека процедур. Значения УПС в прерванной и запущенной процедурах равны соответственно N и N + 1.

В остальном принцип работы защищенного стека пользователя аналогичен рассмотренному ранее.

Освобождение памяти происходит в соответствии со стековой дисциплиной: первым освобождается фрейм, выделенный последним. С этой целью при входе в процедуру до первого заказа памяти дескриптор сохраняется, а при возврате — восстанавливается. Освобожденная память может быть выделена при последующих запросах.

Переиспользование виртуальной памяти в рассматриваемом стеке порождает проблему защиты данных. Нарушение защиты может быть вызвано использованием данных, оставшихся от отработавшей процедуры, или применением зависших указателей, то есть указателей на уже не существующие фреймы. Первая проблема решается автоматической чисткой выделяемой памяти, заключающейся в заполнении ее данными с тегом пустого слова,

а использование зависших указателей исключается введением ограничений в работе со стековыми указателями, которые сводятся к следующему.

Указатели на текущий фрейм процедуры можно сохранять только в текущем фрейме либо передавать в качестве параметра в вызываемую процедуру (передавать вверх по стеку процедур). Поэтому такие указатели нельзя записывать в область глобальных данных, передавать в качестве возвращаемого значения или запоминать в глубине стека. Указанные ограничения касаются как стека пользователя, так и стека процедур.

Для реализации этих ограничений указатели на данные в стеке выделены среди всех адресных типов путем присвоения им отдельных внутренних тегов. Например, указатель на массив (Array Pointer (AP)) отличается внутренним тегом от аналогичного указателя на массив, размещенный в стеке пользователя (Stack Array Pointer (SAR)).

Кроме того, стековые указатели имеют дополнительные средства контроля. При выделении памяти в стеке значение уровня процедуры psl записывается во вновь сформированный указатель и в дальнейшем используется для контроля следующим образом:

- уровень процедуры в указателе не может быть меньше текущего уровня, хранящегося в дескрипторе стека пользователя, следовательно, указатель может быть записан только в текущий фрейм;
- в процессе работы с регистрами стека процедур любая операция не может пользоваться указателем, поступившим из завершившейся процедуры, в качестве возвращаемого значения. В этом случае динамический уровень в указателе будет больше уровня активной процедуры, что недопустимо.

Значение psl можно использовать также для контроля других ограничений в работе с указателями на данные из стека.

Стек связующей информации. Верхняя часть стека (аппаратурная вершина) связующей информации (рис. П5.4) размещается в памяти CR, которую называют также файлом связующей информации — ФС (Chain File, CF). Предназначается эта часть стека для окон незавершенных процедур, выделяемых по мере их запуска. Обычно окно занимает два регистра CR, обеспечивающих хранение квадрослова. В него пакуется вся связующая информация для текущей процедуры, к которой относятся указатель команды возврата, содержимое предикатного файла, положение и размер окна в регистровом файле, индекс дескриптора модуля компиляции, в котором

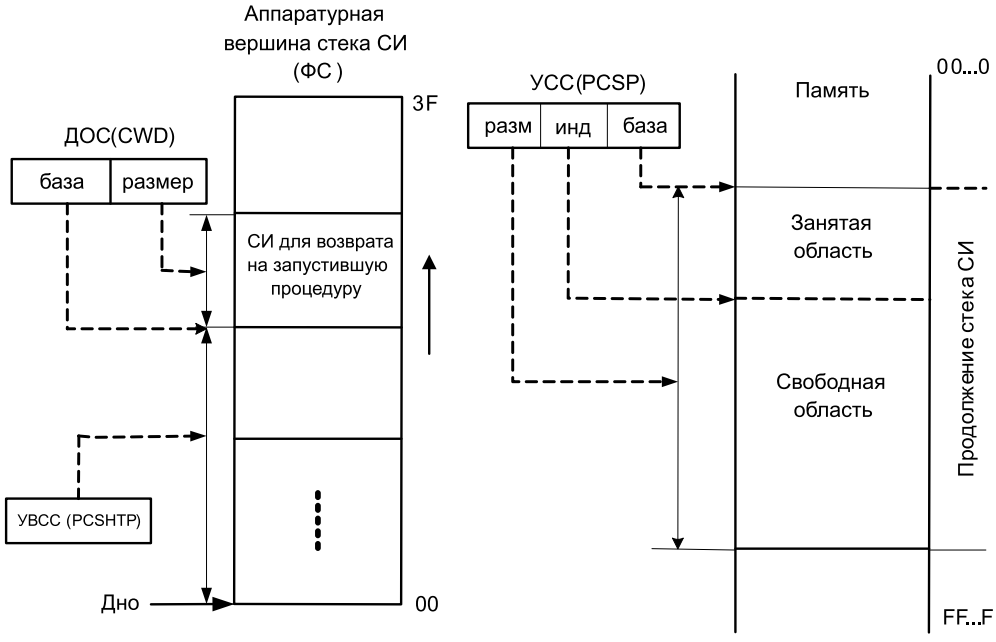


Рис. П5.4. Принцип организации стека связующей информации

описана процедура, и другие объекты. Окно текущей процедуры описывается дескриптором ДОС (CWD), в котором указываются база окна и его размер. Помимо дескриптора регистровая часть стека описывается указателем аппаратурной вершины стека связующей информации (УВСС) (PCSHTP). Он, как и соответствующий указатель стека процедур, описывает верхнюю часть стека связующей информации. Указатель содержит количество занятых регистров CR, расположенных ниже текущего окна. Следовательно, по нему можно определить адрес дна стека, или адрес регистра, с которого начинается откачка. Отрицательное значение указателя соответствует числу регистров, которое необходимо подкачать из памяти в файл связующей информации.

Вторая часть стека размещена в памяти и предназначается для хранения данных, откачиваемых из аппаратурной вершины в случае ее переполнения. Аналогично процедурному стеку, эта часть описывается указателем, имеющим виртуальный адрес базы, размер и индекс относительно базы первого «пустого» слова, с помощью которого осуществляется определение адреса откачки по памяти.

Таким образом, стеки процедур и связующей информации имеют одинаковую организацию. Оба стека имеют регистровую часть (аппаратурную

вершину стека), которая используется текущей процедурой и хранит данные ранее запущенных, но не завершенных процедур. Поскольку емкость регистровой части ограничена, возможно ее переполнение. В этом случае автоматически происходит пересылка данных из регистровой части в подготовленный участок памяти (откачка). Очевидно, что необходим и обратный процесс — пересылка данных из памяти в аппаратурную вершину стека — подкачка.

Что касается стека пользователя, то он предназначен для размещения динамических структур данных, например массивов. Механизм выделения динамической памяти с использованием стандартных процедур здесь заменен более эффективным способом: введены стек и операции, с помощью которых сама процедура может выделить в стеке область памяти необходимого размера. По завершении процедуры эта область возвращается в ресурс свободной памяти. Возникшие проблемы защиты данных при переиспользовании памяти эффективно разрешены снабжением указателей на память в стеке специальными тегами и введением в указатели и дескрипторы уровня (номера) процедуры в стеке процедур. Этот номер позволяет определить принадлежность указателя к той или иной процедуре и тем самым решить проблему защиты данных.

ПРИЛОЖЕНИЕ 6
Выполнение аппаратурных операций откачки и подкачки содержимого регистров РгФ

Откачка. Регистр РгФ имеет 84 разряда и может хранить значения, имеющие формат слова, двойного слова или extended (80-разрядное вещественное число, или тип FX — Floating point eXtended — расширенный формат вещественного числа), с их тегами. При хранении данных типа FX 68 разрядов регистра отводятся для дробной части с тегом, а в дополнительном 16-разрядном поле размещаются код порядка и знак.

В памяти для хранения одного значения формата extended потребуются две 68-разрядные ячейки: одна назначается для мантиссы с тегом, другая — для порядка. Принцип размещения данных типа FX показан на рис. П6.1, где 68-разрядные значения дробных частей с тегами обозначены буквами A, B, C, D, а 16-разрядные части — соответствующими буквами с индексом «пор». Остальные значения словного и двухсловного форматов размещаются в регистрах РгФ и памяти обычным образом. Квадрослово располагается в двух регистрах РгФ.

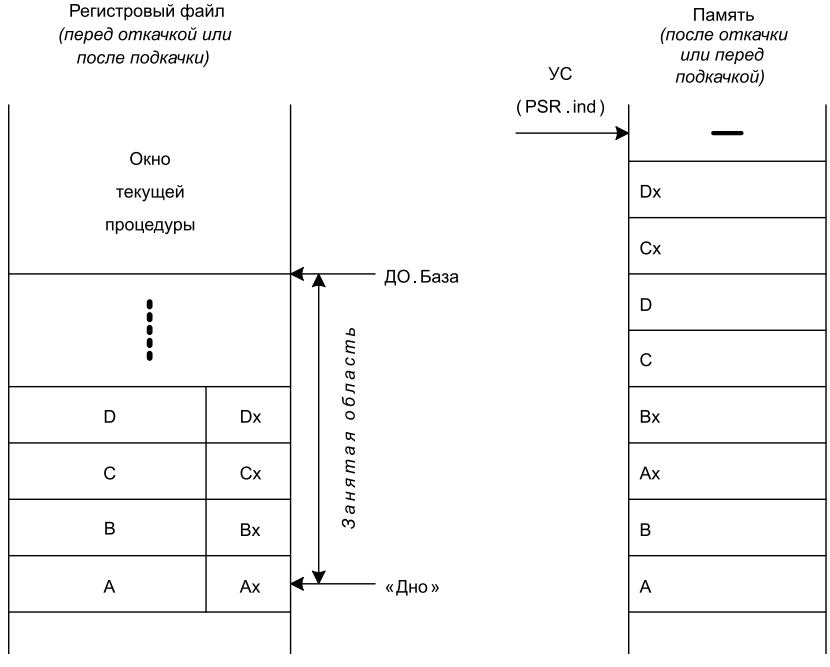


Рис. П6.1. Расположение значения Extended в регистровом файле и памяти

Особенность хранения значений extended отражается на организации откачки. Она может выполняться в одном из двух режимов: обычном и режиме откачки вещественных 80 (mode_80).

В обычном режиме в каждом такте откачиваются два регистра РгФ. Для обеспечения защиты РгФ синхронно с записью в память освободившиеся регистры прописываются данными с тегом «пустышки». В результате после окончания откачки вся откачанная область регистрового файла оказывается заполненной «пустышками».

В режиме откачки вещественных 80 в каждом такте откачиваются также два регистра РгФ, но вначале пересылаются в память мантиссы двух чисел, затем откачиваются поля регистров, хранящих порядки. В результате в каждом такте прописываются четыре ячейки памяти.

Первыми откачиваются данные из регистров «дна» стека процедур (A, B, $A_{пор}$, $B_{пор}$). В каждом цикле откачки указатели стека процедур модифицируются: поле индекса указателя УВС (PSHTP.ind) уменьшается («дно» перемещается вверх по АВС), значение УС (PSR.ind) увеличивается, открывая доступ к следующей ячейке памяти.

Для простоты тип данных каждого конкретного регистра не рассматривается, а применяется метод группирования данных одной процедуры. В соответствии с ним процедура флагом WD.fx объявляет наличие у нее переменных типа FX. Кроме того, в указателе АВС имеется индекс самого верхнего регистра (PSHTP.fxind), который содержит (или содержал) данные типа FX. Данные нижележащих регистров предполагаются вещественными 80. Анализ поля указателя PSHTP.fxind позволяет выбрать режим откачки: пересылать данные из дополнительных полей регистров РгФ либо эти поля не использовать.

На время откачки команды, вызвавшие откачку (очистки стека или установки окна), блокируются в фазе базирования (фазе B). Операция откачки выполняется четырьмя каналами АЛУ, работающими одновременно: два из них (2 и 5) осуществляют пересылку данных с двух регистров РгФ в память, каналы 1 и 4 обеспечивают запись в эти регистры пустых значений.

Подкачка. Характер процесса подкачки определяется типом подкачиваемых данных. В режиме mode_80 в каждом такте подкачиваются два значения вещественного 80, занимающих четыре ячейки памяти (см. рис. П6.1). В обычном режиме за один такт происходит запись 68-разрядных значений в четыре регистра РгФ. Таким образом, если в приложении используются

только целочисленные операции, скорость подкачки (откачки) возрастает почти вдвое по сравнению с вариантом использования смешанных типов данных.

В процессе подкачки в каждой ее итерации происходит увеличение значения УВС (PSHTP), в результате чего дно АВС опускается, открывая доступ к следующим регистрам РгФ. Одновременно уменьшается значение УС (PSP.ind) и указатель перемещается в направлении базы части стека, расположенной в памяти.

Процесс подкачки в регистровый файл происходит во время блокировки (на фазе В) широкой команды, которая следует за широкой командой с операцией возврата из процедуры или прерывания. Если начать процесс подкачки, не дожидаясь прихода команды, следующей за командой возврата, то при прерывании не будет известен адрес команды, на которую надо перейти после возврата из прерывания. Если произошло прерывание, а подкачка не завершилась, то после обработки прерывания на команде возврата из процедуры обработки прерывания повторно устанавливается режим подкачки. После поступления следующей команды на фазу В начнется процесс подкачки, при этом начатые, но незаконченные подкачки повторятся.

В процессе подкачки задействуются одновременно четыре АЛК (0 и 2, 3 и 5). В режиме подкачки вещественных FX каналы 0 и 2 загружают две части порядков, а каналы 3 и 5 — две дробные части. Каждый канал использует дополнительные индексы, обеспечивающие обращение к памяти за своими частями и запись их в РгФ по назначению. В обычном режиме каналы 0 и 2 выполняют подкачку 68-разрядных значений.

ПРИЛОЖЕНИЕ 7
Формат операций обращения и хранения данных для кэш-памяти L2$

Формат операции обращения в L2$. Операция обращения в кэш второго уровня имеет формат, представленный на рис. П7.1.

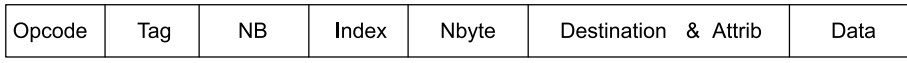


Рис. П7.1. Формат операции обращения в кэш второго уровня

Здесь:

- Opcode — код операции (чтение/запись, количество байт и др.);
- Tag — разряды поля тега [39:16] физического адреса;
- Index — разряды номера строки [15:8] физического адреса;
- NB — разряды номера банка [7:6] физического адреса;
- Nbyte — разряды номера байта внутри кэш-блока [5:0] физического адреса;
- Destination & Attrib — разряды адреса назначения (RF, AAU, L1$ и т. д.);
- Data — записываемые данные (в каналах 2 и 5).

Формат хранения данных в L2-кэше. В состав кэш-блока данных входят 8 двойных слов данных, а также теги данных (4 бита на двойное слово) (рис. П7.2).

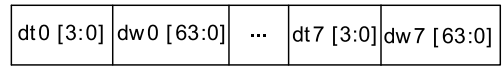


Рис. П7.2. Структура блока данных L2$

Каждому блоку данных соответствует адресный тег — 24 бита и биты состояния — 5 бит. Расположение и назначение битов состояния L2-кэша показаны на рис. П7.3.

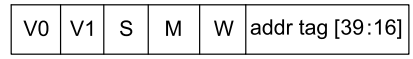


Рис. П7.3. Структура адресного тега и битов состояния блока данных L2$

Здесь V0, V1 — значимости половин кэш-блока данных dw 0...3, dw 4...7. При подкачке данных из ОП половины кэш-блока могут поступать не последовательно, а с разрывом во времени, зависящим от частоты внешнего интерфейса. Минимальный разрыв равен 1 такту. Порядок поступления зависит от 5-го разряда адреса, вызвавшего промах (miss). Разделенная значимость обеспечивает доступность половины блока сразу после ее прихода.

S — shared — признак наличия данных в кэшах других процессоров. При записи в блок с S = 1 вырабатывается запрос Coherent Invalidate в MAU. Invalidate — составляющая запроса CI (Coherent Invalidate) или запроса CRI (Coherent Read & Invalidate) — передается в L2$ (в физическую копию тегов) для уничтожения строки L2$, то есть производится чистка кэша. При поступлении ответа на Coherent Invalidate S сбрасывается в 0, M устанавливается в 1. До поступления ответа на запрос CI запись считается глобально не выполненной.

M — modified — признак модифицированных данных. При M = 1 данные не могут быть вычеркнуты из L2$. При замещении блока производится операция откачки (write-back). При поступлении запроса Coherent Read или Coherent Read&Invalidate от других процессоров производится операция выдачи модифицированных данных (implicit write-back).

W — wait — признак ожидания — незаконченная операция обмена с ОП. W устанавливается в 1 при заведении нового блока в L2$ до поступления из ОП всего блока данных. W = 1 запрещает вытеснение блока (в случае двух последовательных промахов (miss) с разными адресами, но в один блок L2$). Состояние wait обеспечивается также наличием значимого блока в буфере записи, то есть наличием незавершенных записей в данный блок.

Addr tag — адресный тег, разряды [39:16] физического адреса.

При внешнем запросе производится считывание тега и битов состояния L2$. В зависимости от типа запроса и состояния строки вырабатывается ответ в MAU:

- Invalid — нужной строки в L2$ нет;
- Shared — строка чистая;

- implicit Write Back — строка модифицирована, следует выдача данных.

Каждому сету (четыре одноименных блока разных столбцов одного банка) соответствует трехбитовая память состояний (псевдоLRU). Трехбитовый код показывает, какой блок из четырех будет заменен при промахе с данным индексом. Память обновляется при каждом попадании в блок и при заведении нового блока. Разряд A определяет пару столбцов (0, 1 или 2, 3). Разряд B выбирает из столбцов 0, 1. Разряд C выбирает из столбцов 2, 3. Например, при попадании в столбец 0 разряд A устанавливается в 1, разряд B устанавливается в 1, разряд C не изменяется. Примеры заведения данных в различные столбцы приведены в табл. П7.1.

При заведении нового блока сначала проверяется, нет ли свободного блока. При этом значения битов состояния и признака ожидания должны быть равны нулю (V0 = 0, V1 = 0, W = 0). Если таких блоков более одного, то выбирается младший по номеру. Если все блоки заняты, механизм LRU выбирает блок для удаления.

Таблица П7.1. Пример заведения данных в различные столбцы банка

A	B	C	Действие
0	0	x	Заведение в столбце 0
0	1	x	Заведение в столбце 1
1	x	0	Заведение в столбце 2
1	x	1	Заведение в столбце 3

ПРИЛОЖЕНИЕ 8
Структура и параметры отображения памяти в таблицах MMU

Структура отображения памяти в MMU для страниц размером 4 Кбайт приведена на рис. П8.1 Для оптимального размещения таблиц страниц в ОП преобразование адресов выполняется с помощью 4-уровневых таблиц страниц.

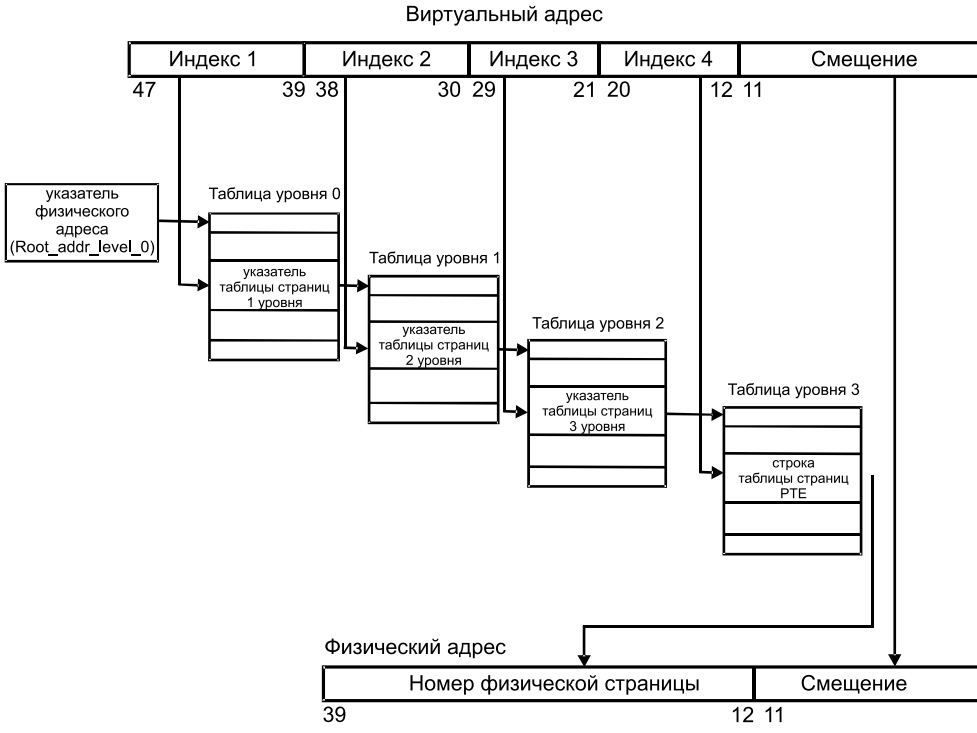


Рис. П8.1. Адресная трансляция с использованием 4-уровневых таблиц страниц

Корневая страница (уровень 0), называемая, как правило, каталогом таблиц страниц, содержит 512 64-разрядных элементов (строк). Каждый из них адресует подчиненную таблицу страниц, то есть допускается существование до 512 подчиненных таблиц страниц (уровень 1). В свою очередь элементы таблиц страниц 1-го уровня адресуют подчиненные таблицы страниц 2-го уровня, а элементы таблиц страниц 2-го уровня адресуют подчиненные таблицы страниц 3-го уровня. Каждая строка (Page Table Entry (PTE)) 3-го уровня адресует физическую страницу. Каждая таблица занимает $512 \times 64 = 4$ Кбайт, то есть ровно 1 страницу.

Таким образом, общее количество адресуемых физических страниц равно 2^{36}, то есть все виртуальное адресное пространство (4 Кбайт × 2^{36} элементов = 2^{48} байт). При этом последний уровень таблицы, описывающий исходное 48-разрядное виртуальное пространство, занимает 2^{27} 4-килобайтовых страниц. Фрагмент внутри него, содержащий PTE, описывающий этот последний уровень таблицы, занимает уже 2^{18} 4-килобайтовых страниц. Следующий уровень — 2^{9} 4-килобайтовых страниц. Наконец, нулевой уровень занимает одну страницу (4 Кбайт), и адресация к нему осуществляется через указатель по физическому адресу: Root_addr_level_0 [47:39].

Разряды виртуального адреса, участвующие в отображении страниц каждого уровня, приведены далее:

- Virt_addr_level_1 [47:30] — 18 разрядов виртуального адреса PTE 1-го уровня таблицы (отображает страницы 2-го уровня таблицы);
- Virt_addr_level_2 [47:21] — 27 разрядов виртуального адреса PTE 2-го уровня таблицы (отображает страницы 3-го уровня таблицы или 4-мегабайтовые страницы пользователя);
- Virt_addr_level_3 [47:12] — 36 разрядов виртуального адреса PTE 3-го уровня таблицы (отображает непосредственно страницы пользователя);
- Virt_addr [11:0] — 12 разрядов виртуального адреса показывают смещение внутри страницы (page size 4 Кбайт).

Страницы размером 4 Мбайт описываются с помощью двух смежных PTE 2-го уровня таблицы.

Структура строки таблицы страниц (PTE) для архитектуры «Эльбрус» представлена на рис. П8.2.

Бит присутствия P (Present) показывает, отображается ли адрес страничного кадра (таблицы страниц или страницы памяти) на страницу в физической

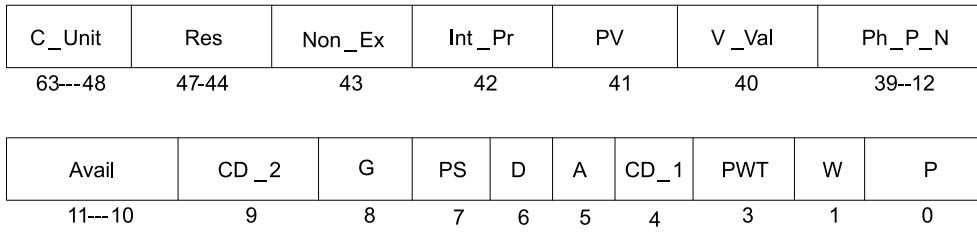


Рис. П8.2. Структура строки таблицы страниц архитектуры «Эльбрус»

памяти. При P = 1 страница присутствует в ОЗУ. При P = 0 страницы в памяти нет, и обращение к этой странице вызывает прерывание типа «страничное нарушение».

Бит W (Writable) разрешает запись в страницу.

Бит PWT разрешает режим сквозной записи (Write Through) в кэш второго уровня L2$.

Биты управления кэшируемостью обращений в кэши разных уровней CD_1, 2 (Cache Disable) позволяют устанавливать следующие режимы:

- 00 — все кэши доступны;
- 01 — DCACHE1 запрещен;
- 10 — DCACHE1, DCACHE2 запрещены;
- 11 — DCACHE1, DCACHE2, ECACHE запрещены.

Бит доступа A (Access) устанавливается микропроцессором в состояние A = 1 при обращении к данному страничному кадру для записи или чтения информации (используется только на 3-м уровне таблицы).

Бит модификации D (Dirty — грязный) устанавливается процессором равным 1 при записи в данную страницу. Для элементов каталога таблиц страниц значение бита D является неопределенным. При загрузке страницы в память операционная система сбрасывает бит D. Если при необходимости выгрузки страницы во внешнюю память оказывается, что для нее D = 0, это означает, что страница не модифицирована относительно копии в оперативной памяти.

Бит размера страниц PS (Page_Size) устанавливается равным 0 (1) при использовании страниц размером 4 Кбайт (4 Мбайт) соответственно.

Бит G (Global) отменяет сравнение контекстов, то есть данная страница считается общей для всех задач.

Биты Avail (Available) используются операционной системой и игнорируются аппаратурой.

Ph_P_N (Phys_Page_Number) [39:12] — физический адрес страницы. Это старшие 28 разрядов 40-разрядного физического адреса. Если PTE отображает 4-мегабайтные страницы, то младшие 10 разрядов [21:12] в физическом номере страницы PPN игнорируются.

V_Val (V_A_Valid) — бит значимости. V_Val = 0 соответствует несуществующей виртуальной странице.

Бит доступа PV (Privileged) показывает, что страница доступна только в привилегированном режиме.

Бит Int_Pr (Int_Addr_Protect) устанавливает запрет обращения в страницу со специальным кодом операции.

Non_Ex (Non_Execute) — бит запрета исполнения.

Res (Reserved) — резервный бит (не используется).

C_Unit (Compilation_Unit) служит индексом блока дескрипторов модуля компиляции в таблице CUT.

Для сравнения на рис. П8.3 представлена структура строки таблицы страниц архитектуры Intel.

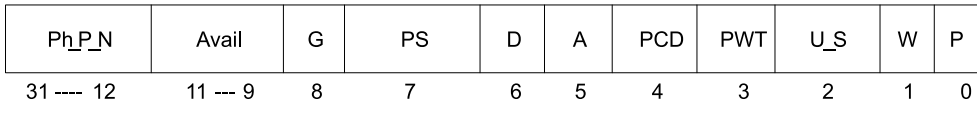


Рис. П8.3. Структура строки таблицы страниц архитектуры Intel

ПРИЛОЖЕНИЕ 9
Алгоритм замещения элементов в строке DTLB

Каждому элементу в DTLB соответствуют бит значимости VAL и бит использования USED. При любом обращении к значимому элементу устанавливается в единицу бит USED того столбца, в котором произошло сравнение. Если в системе используются страницы размером 4 Кбайт и 4 Мбайт, то для каждого размера страниц алгоритм анализирует биты VAL и USED предназначенных ему столбцов.

При заведении нового элемента в DTLB сначала проверяется наличие такого адреса в таблице, в случае попадания (hit) эта ячейка перезаписывается. Замещение ячейки при совпадении адресов имеет наивысший приоритет. При отсутствии совпадения для заведения нового элемента в строке выбирается первый свободный элемент. Если все элементы заняты, то выбирается первый из элементов с нулевым USED-битом. Если при обращении к строке USED-биты всех элементов строки (тех столбцов, которые предназначены для размещения страниц данного размера) оказываются установленными в единицу, происходит их обнуление, за исключением расположенного в столбце, в котором произошло сравнение. Этот же алгоритм используется при замещении регистров, предназначенных для хранения двойных слов MPT.

Обслуживание запросов в L1$ и DTLB занимает 2 такта. Алгоритм выполнения операций на любом канале зависит от состояния регистров MMU. Регистр управления MMU_CR представлен на рис. П9.1.

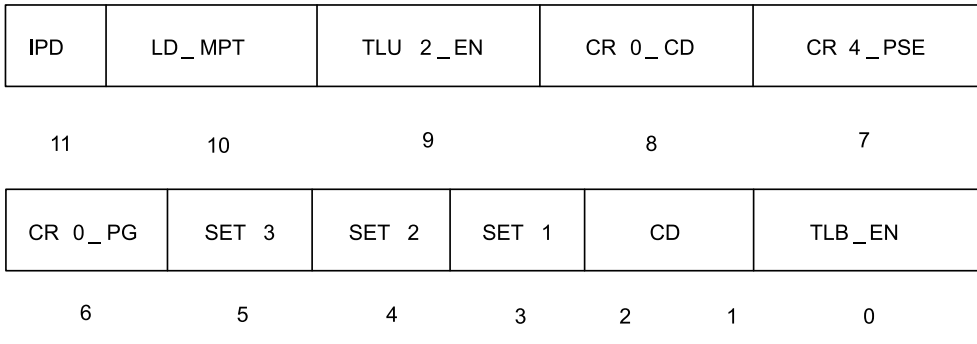


Рис. П9.1. Структура регистра управления MMU

Он содержит биты, которые определяют режимы работы устройства.

- Бит TLB_EN (TLB ENABLE) предназначен для включения режима трансляции первичного виртуального адреса в физический.
- Биты CD (Cache Disable) управляют кэшируемостью обращений в кэши разных уровней.
- Биты SET1, SET2, SET3 определяют, какие столбцы TLB отводятся для размещения страниц размером 4 Кбайт и 4 Мбайт.
- Бит CR0_PG (PAGING ENABLE) включает режим трансляции в DTLB вторичного виртуального адреса в физический.
- Бит CR4_PSE (Page size extension enables) разрешает использование страниц размером 4 Мбайт в системе для Intel-обращений.
- Бит CR0_CD (Cache Disable Secondary) управляет кэшируемостью обращений во вторичное виртуальное пространство.
- Бит TLU2_EN (TLU ENABLE Secondary) включает режим аппаратного поиска по таблице страниц вторичного виртуального пространства в случае промаха в TLB.
- Бит LD_MPT включает проверку битов Memory Protection Table для Intel-обращений на считывание.
- Бит IPD (Instruction Prefetch Depth) определяет глубину подкачки по прямой ветви: 0 — 0 строк, 1 — 2 строки буфера команд.

Рассмотрим функционирование DTLB в режиме трансляции виртуального адреса. Если заявка, поступающая на вход DTLB, имеет признак виртуального пространства «Эльбрус» (ROOT = 0) и в спецификаторе адреса памяти (MAS) не установлен режим отключения трансляции, то включение режима трансляции происходит при установке в единицу бита Enable virtual address translation в MMU Control register — MMU_CR.TLB_EN = 1.

Виртуальный адрес заявки, расширенный полем контекста текущей задачи (содержимое регистра CONT), вместе с признаком виртуального пространства «Эльбрус» участвует в поиске по DTLB.

7 младших разрядов виртуального номера страницы (для страниц размером 4 Кбайт это разряды [18:12], для страниц размером 4 Мбайт — [28:22]) используются в качестве индекса для считывания информации из памяти тегов и строк одновременно из 4 столбцов. Считанная из памяти тегов информация сравнивается с входной заявкой. В сравнении участвуют:

- признак виртуального пространства «Эльбрус» (ROOT);
- оставшиеся 29 разрядов (для страниц размером 4 Кбайт — [47:19]) или 19 разрядов (для страниц размером 4 Мбайт — [47:29]) виртуального номера страницы;
- поле контекста при отсутствии установки в единицу global flag (DTLB_TAG.G = 0).

В случае попадания hit в одном из столбцов на выходе DTLB коммутируется поле физического адреса страницы (DTLB_ENTRY.PHA) соответствующего столбца. Полный выходной физический адрес заявки формируется с учетом размера страницы.

В случае промаха в DTLB спекулятивная операция считывания не инициирует поиска в таблице страниц PT, а завершается и выдает в качестве результата диагностическое значение.

Таким образом, с целью оптимизации размещения таблиц страниц в оперативной памяти преобразование адресов выполняется с помощью 4-уровневых таблиц страниц. В этом случае в ОП должны постоянно находиться лишь каталог таблицы страниц и таблицы страниц выполняемой в настоящее время программы. Остальные таблицы страниц могут временно находиться во внешней памяти.

ПРИЛОЖЕНИЕ 10
Настройки режима работы в реальном времени в ОС «Эльбрус»

RTS_KERN_PRMPT — этот режим позволяет ядру принудительно снимать поток с процессора в ходе планирования вычислений. Переключение происходит, как только появился более приоритетный процесс и ядро готово к этому. По умолчанию режим включен.

RTS_HIRQ_PRMPT — для всех устройств только подтверждение приема внешнего прерывания (TopHalfAck) происходит на прерванном потоке. Следующая фаза обработки прерывания (TopHalfDev) запускается на отдельном системном потоке (одном на каждое прерывание). По умолчанию режим включен.

RTS_SOFT_ON_HARD — BottomHalf запускается на том же потоке, на котором выполняется TopHalfDev. В противном случае активизируется еще один поток для продолжения обработки прерывания. По умолчанию режим включен.

RTS_NO_DYN_PRIO — не осуществляется динамический пересчет приоритетов и переупорядочивание очередей к процессору. Вместо этого используется круговой алгоритм. По умолчанию режим выключен.

RTS_NO_CPU_BLNC — автоматическая миграция процессов между процессорами для поддержания оптимальной загрузки не выполняется, и выключается динамический пересчет приоритетов. По умолчанию режим выключен.

RTS_NO_RD_AHEAD — запрет подкачки впрок при чтении файлов. По умолчанию режим выключен.

RTS_NO_IO_SCHED — запрет оптимизаторов дисковых обменов («элеваторов»). По умолчанию режим выключен.

RTS_PGFLT_RTWRN — выдавать сообщение (с указанием места в программе), если в процессе, работающем в режиме реального времени, возникло прерывание по причине отсутствия страницы в памяти. По умолчанию режим выключен.

RTS_PGFLT_WRN — выдавать сообщение (с указанием места в программе), если в любом процессе возникло прерывание по причине отсутствия страницы в памяти. По умолчанию режим выключен.

RTS_NO_PTIMERS — отключение поддержки таймеров POSIX. По умолчанию режим выключен.

Из приведенных значений с помощью операции | (логическое ИЛИ) формируется аргумент `mask` функции `el_set_rts_active()`.

Для использования режима жесткого реального времени отключенные по умолчанию режимы RTS_NO_DYN_PRIO, RTS_NO_CPU_BLNC, RTS_NO_RD_AHEAD, RTS_NO_IO_SCHED должны быть включены.

Режимы RTS_PGFLT_RTWRN, RTS_PGFLT_WRN, RTS_NO_PTIMERS предназначены для отладки работы в режиме жесткого реального времени.

Список литературы

1. *Велихов Е. П.* Сергей Алексеевич Лебедев // Информационные технологии и вычислительные системы. — 2002. — № 3. — С. 31–35.

2. *Бурцев В. С.* Параллелизм вычислительных процессов и развитие архитектуры ЭВМ. — М.: Торус пресс, 2006.

3. *Борисов Ю. И.* Проектные центры компьютеростроения в программах обеспечения национальной безопасности // Матер. конф. «Перспективы развития высокопроизводительных архитектур. История, современность и будущее отечественного компьютеростроения». — 2008. — № 1. — С. 8–14.

4. *Бабаян Б. А.* История развития архитектур вычислительных машин // Ершовские лекции по информатике. — Новосибирск: Изд-во Ин-та информатики им. А. П. Ершова СО РАН, 2009.

5. *Ким А. К., Перекатов В. И., Сахин Ю. Х.* Развитие и реализация архитектуры вычислительных комплексов серии «Эльбрус» для решения задач ракетно-космической обороны // Вопросы радиоэлектроники. Сер. ЭВТ. — 2010. — Вып. 3. — С. 5–17.

6. *Фельдман В. М.* Многопроцессорные вычислительные комплексы отечественной разработки // Сборник научных трудов ИМВС РАН «Высокопроизводительные вычислительные системы и микропроцессоры». — 2003. — С. 3–15.

7. *Ким А. К., Волконский В. Ю., Груздов Ф. А. и др.* Микропроцессорные вычислительные комплексы с архитектурой «Эльбрус» и их программное обеспечение и др. // Вопросы радиоэлектроники. Сер. ЭВТ. — 2009. — Вып. 3. — С. 5–37.

8. *Исаев М. В., Кожин А. С., Костенко В. О. и др.* Двухъядерная гетерогенная система на кристалле «Эльбрус-2С+» // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 42–52.

9. *Ким А. К.* Российские универсальные микропроцессоры и вычислительные комплексы высокой производительности: результаты и взгляд в будущее // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 5–13.

10. *Ким А. К., Фельдман В. М.* Вопросы создания суперЭВМ на основе современной платформы «Эльбрус» // Приборы. — 2009. — № 1. — С. 36–46.

11. *Волконский В. Ю., Ким А. К., Перекатов В. И., Фельдман В. М.* Микропроцессоры и вычислительные комплексы компании МЦСТ // Электроника. — 2008. — № 8. — С. 62–70.

12. *Мелехин В. Ф., Павловский Е. К.* Вычислительные машины, системы и сети. — М.: ACADEMIA, 2010.

13. The SPARC architecture manual: Version 8. — Englewood Cliffs, N.J.: Prentice Hall, 1992.

14. *Александров Е. К.* Микропроцессорные системы: Учеб. пособие / Е. К. Александров, Р. И. Грушвицкий, М. С. Куприянов и др.; Под ред. Л. В. Пузанкова. — СПб.: Политехника, 2002.

15. *Таненбаум Э.* Архитектура компьютера. — 4-е изд. — СПб.: Питер, 2003.

16. *Фельдман В. М.* Вычислительные комплексы «Эльбрус-90 микро» // Информационные технологии. — 2005. — № 1. — С. 17–26.

17. *Схаба М. З.* Вычислительный комплекс «Эльбрус-90 микро» ВК-15: Учеб. пособие. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2010.

18. *Гусев В. К., Ким А. К., Перекатов В. И., Фельдман В. М.* Структура вычислительных средства серии «Эльбрус»: эволюция проектных решений // Вопросы радиоэлектроники. Сер. ЭВТ. — 2011. — Вып. 3. — С. 5–22.

19. *Волин В. С., Черепанов С. А., Щербина Н. А.* Организация поддержки когерентности в системе на кристалле «МЦСТ-R1000» // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 27–42.

20. The SPARC architecture manual: Version 9. — Englewood Cliffs, N.J.: Prentice Hall, 1993.

21. *Волин В. С.* Организация подкачки кода в процессорном ядре системы на кристалле «МЦСТ-4R» // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 14–27.

22. *Hennessy J. L., Patterson D. A.* Computer Architecture. A Quantitative Approach. — 4 rev. ed. — Elsevier Science, 2006.

23. *Волконский В. Ю., Грабежной А. В., Муханов Л. Е., Нейман-заде М. И.* Исследования влияния подсистемы памяти на производительность распараллеленных программ // Вопросы радиоэлектроники. Сер. ЭВТ. — 2011. — Вып. 3. — С. 22–38.

24. *Галазин А. Б., Грабежной А. В., Нейман-заде М. И.* Оптимизация размещения данных для эффективного использования программ на архитектуре с многобанковой кэш-памятью данных // Информационные технологии. — 2008. — № 3. — С. 35–39.

25. *Галазин А. Б., Ступаченко Е. В., Шлыков С. Л.* Программный метод предварительной подкачки кода в архитектурах со статическим планированием // Программирование. — 2008. — № 1. — С. 67–74.

26. *Исаев М. А., Костенко В. О., Поляков Н. Ю.* Проблемы интеграции универсальных ядер архитектуры «Эльбрус» и DSP-кластера в составе системы на кристалле // Вопросы радиоэлектроники. Сер. ЭВТ. — 2010. — Вып. 3. — С. 70–81.

27. *Кожин А. С.* Проблемы передачи данных между асинхронными доменами вычислительных устройств // Вопросы радиоэлектроники. Сер. ЭВТ. — 2011. — Вып. 3. — С. 130–141.

28. *Семенихин С. В., Ревякин В. А., Ананьев Л. И. и др.* ОС Linux и режим реального времени // Вопросы радиоэлектроники. Сер. ЭВТ. — 2009. — Вып. 3. — С. 37–67.

29. *Федоров А. В.* Оптимизация библиотеки нитей NPTL в составе ОС Linux для систем жесткого реального времени // Программирование. — 2011. — № 4. — С. 73–79.

30. *Ахо В. С., Моника М. С., Сети Р., Ульман Д. Д.* Компиляторы: принципы, технологии и инструментарий. — М.: ИД «Вильямс», 2008.

31. *Иванов Д. С.* Распределение регистров при планировании инструкций для VLIW-архитектур // Программирование. — 2010. — № 6. — С. 74–80.

32. *Волконский В. Ю., Брегер А. В., Бучнев А. Ю. и др.* Методы распараллеливания программ в оптимизирующем компиляторе // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 63–89.

33. *Ермолицкий А. Е.* Методы повышения эффективности векторизации в оптимизирующем компиляторе // Вопросы радиоэлектроники. Сер. ЭВТ. — 2010. — Вып. 3. — С. 41–50.

34. *Воронов Н. В., Гимпельсон В. Д., Маслов М. В. и др.* Система динамической двоичной трансляции x86 Эльбрус // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 89–108.

35. *Ким А. К., Волконский В. Ю., Груздов Ф. А. и др.* Защищенное исполнение программ на базе аппаратной и системной поддержки архитектуры «Эльбрус» // Сб. избр. тр. V Междунар. науч.-практ. конф. «Современные информационные технологии и ИТ-образование». — М.: Изд-во Моск. гос. ун-та, 2010. — С. 22–40.