Московский государственный технический университет
имени Н. Э. Баумана

Учебное пособие

А. Ю. Попов

Организация суперскалярных процессоров

Издательство МГТУ им. Н. Э. Баумана

Московский государственный технический университет
имени Н. Э. Баумана

А. Ю. Попов

Организация суперскалярных процессоров

Рекомендовано Научно-методическим советом
МГТУ им. Н. Э. Баумана
в качестве учебного пособия
по курсу «Организация ЭВМ»

Москва
Издательство МГТУ им. Н. Э. Баумана
2011

УДК 681.3.06
ББК 22.18
П58

Рецензенты: *О. М. Брехов*, *Д. А. Ибрагимов*

Попов А. Ю.

П58 Организация суперскалярных процессоров : учеб. пособие по курсу «Организация ЭВМ» / А. Ю. Попов. — М. : Изд-во МГТУ им. Н. Э. Баумана, 2011. — 57, [3] с. : ил.

Рассмотрена архитектура современных суперскалярных процессоров. Приведены сведения, поясняющие состав и назначение входящих в процессоры подсистем. На примере микроархитектуры Intel P6 дано подробное функциональное описание каждой подсистемы. Приведены функциональные описания современных суперскалярных процессоров фирм Intel, AMD, IBM, Sun, MIPS.

Для студентов 4-го курса, обучающихся по специальности «Вычислительные машины, комплексы, системы и сети».

УДК 681.3.06
ББК 22.18

Современные суперскалярные микропроцессоры обладают многими достоинствами, обеспечивающими высокий уровень производительности в составе ЭВМ и вычислительных систем. Так, согласно информации сайта www.top500.org, 99 % микропроцессоров, используемых в высокопроизводительных вычислительных системах, являются суперскалярными. Знание их внутренней структуры и особенностей функционирования позволяет разрабатывать системное и прикладное программное обеспечение, эффективно использующие аппаратные ресурсы.

Данное пособие знакомит студентов с основами архитектуры и принципами работы современных суперскалярных микропроцессоров ведущих мировых производителей: Intel, AMD, IBM, Sun, MIPS. Более подробную документацию для рассмотренных устройств можно найти на сайтах указанных производителей.

1. Принципы построения суперскалярных процессоров

1.1. Архитектура современных процессоров

С момента появления в 1971 г. первого микропроцессора архитектура ЭВМ претерпела существенные изменения. Сокращение технологических норм позволило разместить на одном кристалле большое число исполнительных устройств: целочисленных арифметико-логических устройств, устройств выполнения операций над числами с плавающей запятой, устройств векторных вычислений и др. Для их эффективного использования потребовалась реализация принципов конвейеризации вычислительного процесса и суперскалярной обработки.

Конвейерная обработка представляет собой процесс, при котором сложные действия разделяются на более короткие стадии. Их параллельное выполнение позволяет более полно использовать обрабатывающие ресурсы конвейера.

Применительно к ЭВМ конвейерная обработка нашла широкое распространение при выполнении процессорным устройством

входного потока команд. Так, цикл команды можно разделить на шесть последовательных стадий: выборку команды, декодирование, вычисление адресов операндов, выборку операндов, исполнение команды и запись результатов. Процесс обработки начинается с выборки очередной команды. После получения команды из подсистемы памяти процессор приступает к ее декодированию, т. е. к преобразованию кода операции в код одного или нескольких последовательных управляющих действий, называемых микрокомандами. Если для исполнения команды необходимо также получить операнды из памяти, то одна из микрокоманд управляет извлечением адресов операндов из команды и преобразованием их в адреса физической памяти. При использовании виртуальной памяти вычисление адреса операнда может занять у микропроцессора длительное время. Сразу же после получения всех предусмотренных в команде операндов, микропроцессор начинает исполнять команду в соответствии с управляющими сигналами, задаваемыми микрокомандой исполнения. Полученный таким образом результат может быть сохранен в памяти или во внутреннем регистре с помощью микрокоманды записи результатов. Общее время цикла выполнения одной команды принято называть *латентностью*.

Идеализированная последовательность исполнения команд на конвейере, состоящем из шести стадий, показана на рис. 1, *а*. Для более полного использования аппаратных ресурсов процессора и соответственно увеличения их производительности используются конвейеры с бóльшим количеством стадий. Такие процессоры называются суперконвейерными. Из рис. 1, *б* следует, что при разделении каждой стадии на два последовательных этапа может быть достигнута бóльшая производительность микропроцессора.

Суперскалярная обработка основывается на способности процессора выполнять более одной простой (скалярной) операции за один такт, что достигается благодаря включению в его состав нескольких параллельно работающих исполнительных устройств. На рис. 1, *в* видно, что дублирование всех блоков конвейерного процессора позволяет завершить обработку четырех команд на два такта быстрее. Реализация этого принципа совместно с внедрением суперскалярной суперконвейерной обработки (рис. 1, *г*) дает

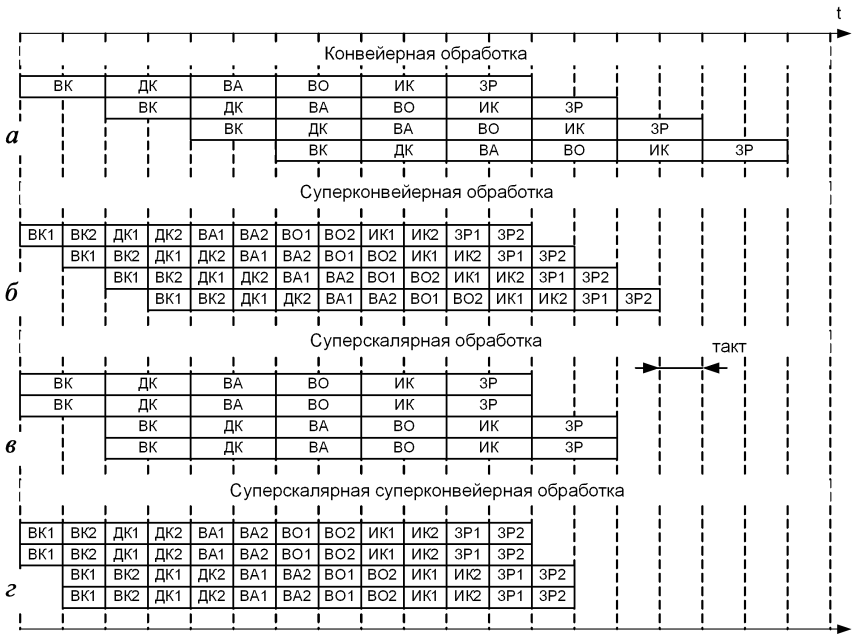


Рис. 1. Пример исполнения команд, состоящих из стадий выборки команды (ВК), декодирования (ДК), вычисления адресов операндов (ВА), выборки операндов (ВО), исполнения команды (ИК), записи результатов (ЗР) с использованием конвейерной структуры (*а*), суперконвейерной структуры (*б*), суперскалярной структуры (*в*), суперскалярной суперконвейерной структуры (*г*)

наилучший результат. В современных микропроцессорах суперскалярность может обеспечиваться двумя способами: исключительно аппаратным способом благодаря выявлению в поступающем потоке команд параллельных ветвей; программно-аппаратным способом благодаря применению специальных оптимизирующих компиляторов для загрузки параллельно работающих устройств.

Внедрение суперскалярности и суперконвейерности явилось существенным достижением разработчиков ЭВМ. Первой успешно реализованной ЭВМ, построенной по указанным принципам, является ЭВМ «Эльбрус-1», созданная в 1978 г. в ИТМиВТ им. С. А. Лебедева. Архитектурные решения, заложенные в эту ЭВМ, используются в современных микропроцессорах. Широкое распространение суперскалярные процессоры получили в 1990-е годы, когда фирма IBM выпустила коммерчески успешный процессор RS/6000

(1990 г.), а фирма Intel выпустила микропроцессор Pentium (1993 г.) и более совершенные процессоры семейства P6 (Pentium Pro, Pentium II, Pentium III в 1995–1999 гг.).

Однако помимо явных преимуществ применение конвейера и дублирование его блоков в микропроцессорах порождают ряд проблем, наиболее значимая из которых обусловлена наличием команд перехода, нарушающих естественный порядок вычислений. Доля таких команд в программах существенна (примерно 10 %), что приводит к сбою ритмичной работы конвейера. Наибольшую задержку в работе конвейера вызывают команды условных переходов, так как только после полного их выполнения становится возможным определить адрес следующей исполняемой команды. В современных микропроцессорах для сокращения потерь времени при обработке таких команд широкое применение нашли предсказание перехода и исполнение альтернативных ветвей программы.

Заметное увеличение тактовой частоты микропроцессоров и совершенствование архитектуры привели к необходимости ускорения обмена информацией между процессором и оперативной памятью. Достижение этой цели усложняется в связи с тем, что быстродействие оперативной памяти растет гораздо медленнее, чем производительность самих процессоров (разрыв в производительности удваивается ежегодно [1]). Растущая потребность в размерах оперативной памяти приводит к ее разделению на блоки меньшего размера и делает затруднительным интенсивную обработку информации на границе блоков. Для решения указанных проблем используется целый комплекс архитектурных решений.

• *Спекулятивная выборка данных и предсказание направления ветвления.* Оба этих способа заключаются в заблаговременном определении и получении из оперативной памяти тех команд и данных, которые предстоит обработать процессору. Спекулятивная выборка, инициированная самим процессором на основе анализа поступающего потока команд, называется *аппаратной предвыборкой.*

• *Использование команд управления уровнями памяти*: команды принудительной загрузки данных в кэш-память и буферы чте-

ния/записи позволяют оптимизировать процесс вычисления для конкретного микропроцессора. Спекулятивная выборка, инициированная специальными командами, называется *программной предвыборкой*.

• *Неупорядоченное выполнение команд в процессоре.* Этот принцип основан на механизме выявления зависимостей по данным. С помощью такой информации процессор может изменить исходный порядок исполнения команд и микрокоманд, заложенный в программу.

• *Аппаратная реализация косвенной адресации*, ускоряющая вычисление адресов операндов.

• *Виртуализация адресного пространства*, реализующаяся в виде сегментно-страничной и более сложных типов организации памяти. Эти принципы предполагают определение физического адреса оперативной памяти из логического адреса, указанного в команде, по специальным таблицам. Для ускорения доступа к ним применяют специальную кэш-память (Translation Look-Aside Buffer — TLB), в которую помещают часто используемые строки таблиц или результаты преобразований.

• *Использование пакетной передачи информации.* Такой способ применяется в предположении, что информация выбирается из последовательных ячеек ОП, и называется предположением о кучности адресов. Это позволяет ускорить выполнение многих приложений.

• *Использование иерархии памяти.* Это решение предполагает перемещение данных между скоростным процессором и медленной памятью через буферную и кэш-память. Кэш-память гораздо меньше по объему, чем оперативная, и содержит часто используемую процессором информацию. Буферы призваны группировать данные в пакеты, способствуя ускорению их передачи по шинам.

• *Принудительное разделение потоков команд и данных.* Такой способ позволяет оптимизировать обработку разнородной информации, извлекаемой из ОП.

Обобщенная структура процессора, использующего перечисленные архитектурные особенности и основные информационные потоки, показана на рис. 2. В процессоре реализованы несколько

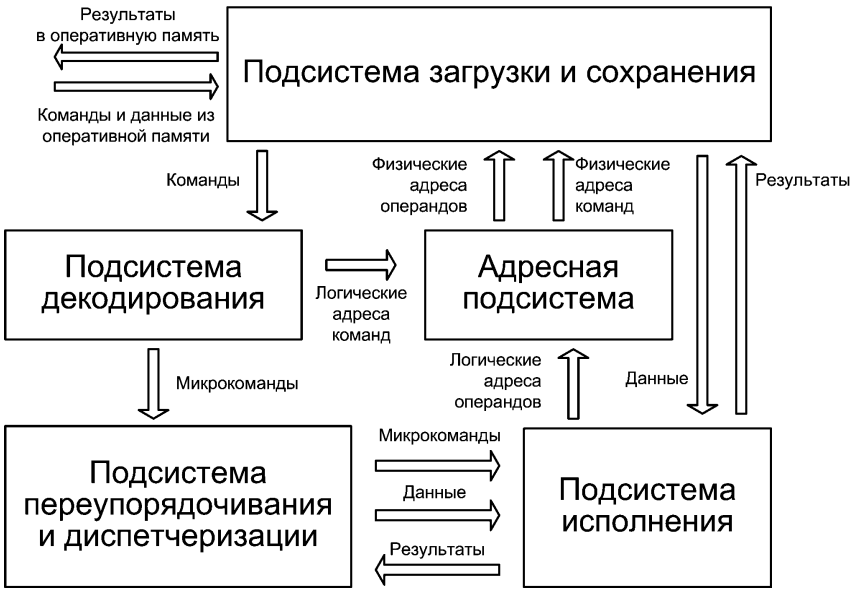


Рис. 2. Обобщенная структура суперскалярного процессора

функционально различных подсистем, обеспечивающих конвейерную и суперскалярную обработку инструкций программы.

Подсистема загрузки и сохранения команд и данных обеспечивает связь процессора с ОП, а также выполняет временное хранение и выдачу другим подсистемам кодов исполняемых команд, операндов и результатов. Эта подсистема получает на вход физические адреса команд и данных, после чего обеспечивает выборку затребованной информации из памяти и передает ее в другие подсистемы. Как правило, эта подсистема состоит из устройства шинного интерфейса, устройства переупорядочивания запросов к памяти и многоуровневой кэш-памяти для хранения команд и данных. В некоторых моделях микропроцессоров в такую подсистему входит контроллер памяти. Более подробное описание логики функционирования этой и других подсистем представлено в следующих подразделах.

Адресная подсистема позволяет микропроцессору своевременно определять адреса востребованных в вычислительном процессе команд и данных, а также обеспечивает преобразование логических адресов в физические и выдает запросы на загрузку операндов

в подсистему загрузки и сохранения и выгрузку их из нее. Эта подсистема, как правило, состоит из устройства вычисления адреса следующей команды, буферов быстрого страничного преобразования (TLB-буферов), блока предсказания направления ветвления.

Подсистема декодирования предназначена для определения последовательности микрокоманд, необходимых для реализации поступающей последовательности инструкций программы. Обычно она состоит из памяти микропрограмм, устройства предекодирования и декодирования, блока переименования регистров.

Подсистема переупорядочивания и диспетчеризации реализует функции хранения декодированных микрокоманд в ожидании поступления их операндов, распределяет микрокоманды по мере их готовности на свободные исполнительные устройства, сохраняет результаты обработки микрокоманд и обеспечивает своевременное удаление исполненных микрокоманд. Последовательность исполнения микрокоманд может отличаться от исходной последовательности инструкций программы. Данная подсистема состоит из таблицы регистровых псевдонимов, буфера переупорядочивания микрокоманд, устройства удаления и восстановления, регистров замещения и архитектурных регистров, буфера готовых микрокоманд, портов запуска.

Подсистема исполнения состоит из устройств, непосредственно обрабатывающих операнды в соответствии с кодами микрокоманд. Помимо арифметических устройств, обрабатывающих числа с фиксированной и плавающей запятой, к таким устройствам относятся блоки, обрабатывающие микрокоманды загрузки и выгрузки операндов (блок связи с памятью), и устройство проверки правильности предсказанных переходов (устройство арифметики переходов).

Разработчики суперскалярных суперконвейерных процессоров ищут эффективные пути дальнейшего повышения их производительности. Характеристики и структура таких процессоров и их подсистем различны. В следующих подразделах будет рассмотрена архитектура суперскалярных процессоров на примере классического семейства P6, эволюционировавшего в более современные архитектуры (Core, Pentium M) [2]. Краткое описание архитектуры

других семейств суперскалярных процессоров (NetBurst, MIPS, Sun SPARC Ultra III, POWER4) приведено в разд. 2.

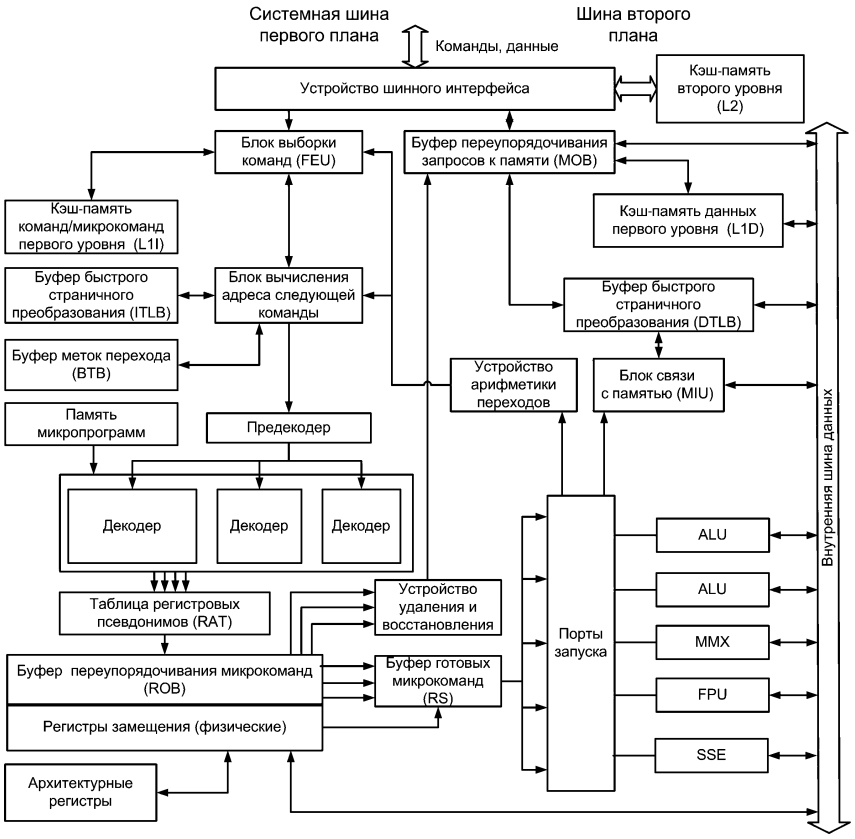


Рис. 3. Структура суперскалярного микропроцессора семейства P6

На рис. 3 более подробно показаны подсистемы суперскалярного микропроцессора семейства P6 и их связи, а их описание приводится в следующих подразделах и в источниках [2, 3, 6, 7, 11].

1.2. Подсистема загрузки и сохранения

Подсистема загрузки и сохранения процессоров семейства P6 состоит из устройства шинного интерфейса, кэш-памяти второго уровня, кэш-памяти команд первого уровня, кэш-памяти данных

первого уровня, блока выборки команд и блока переупорядочивания запросов к памяти.

Устройство шинного интерфейса (*Bus Interface Unit*, *BIU*) обеспечивает взаимодействие процессора с системной шиной, являющейся основным трактом передачи информации в современных ЭВМ. По этой шине процессор может обмениваться информацией с другими процессорами, обращаться к оперативной и кэш-памяти, а также к периферийным устройствам.

Статус центрального тракта передачи данных накладывает жесткие требования на пропускную способность системной шины. С целью ее разгрузки в современных ЭВМ используют иерархию шин. Так, периферийные устройства подключают не напрямую к системной магистрали, а к шинам ввода/вывода, более приспособленным к подключаемым типам устройств (PCI, AGP и т. п.). Контроллеры таких шин реализуют в наборе микросхем — чипсете (chipset), выпускаемом совместно с процессором. На рис. 4 показан возможный вариант подключения периферийных шин через специальную микросхему (концентратор) к системной шине. Отдельное место в этой иерархии занимает шина «второго плана» (Back Side Bus — BSB), соединяющая процессорное ядро с интегрированной на кристалл кэш-памятью второго уровня, что позволяет создать магистраль требуемой разрядности (для последних модификаций семейства P6 разрядность шины второго плана составляет 256 бит, пропускная способность достигает 44,8 Гбит/с). Данная шина также подключается к устройству шинного интерфейса.

Запрос, выдаваемый на системную шину, инициирует новую транзакцию, что требует возврата подтверждения ее выполнения или уведомления инициатора об ошибке. В общем случае заданная последовательность возвращаемых результатов может быть нарушена, что предусматривает применение особых средств идентификации передаваемой информации.

Эта проблема решается с помощью передачи совместно с запросом уникального номера транзакции, по которому процессор идентифицирует возвращаемые результаты. Например, для системной шины процессоров P6 характерна передача запроса за два такта работы шины. В первом такте передается адрес памяти или устрой-

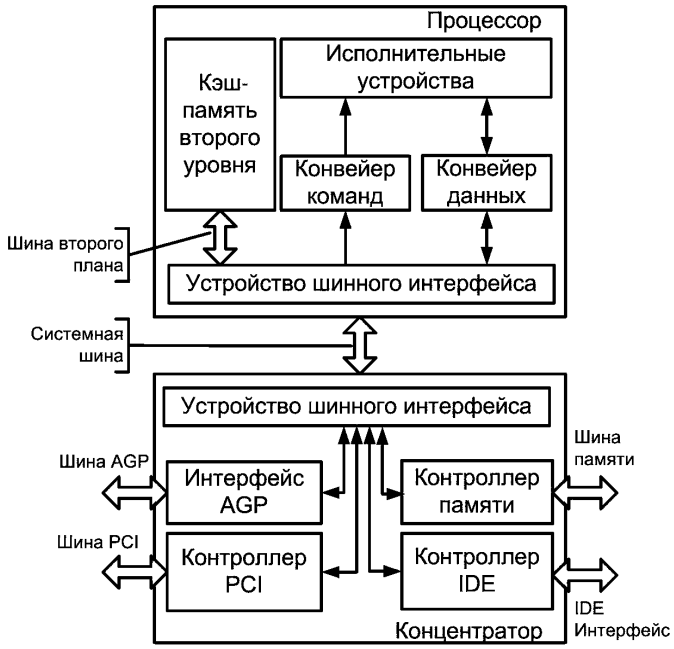


Рис. 4. Пример иерархии шин в современных ЭВМ

ства ввода/вывода и тип запроса, а во втором — идентификатор инициатора, идентификатор транзакции, информация о задействованных линиях передачи данных, параметры кэширования и некоторая дополнительная информация. Идентификаторы инициатора и транзакции являются четырехразрядными, что позволяет подключать к шине до 16 устройств, для каждого из которых на шине может присутствовать до 16 запросов.

Для процессоров AMD характерно разделение двунаправленной системной шины на несколько однонаправленных (технология HyperTransport). Помимо этого, на кристалле реализуется контроллер памяти, что позволяет выполнить разделение шин и ускорить маршрутизацию пакетов, однако ограничивает пользователя в выборе типов ОП.

Инициаторами запросов, выдаваемых процессором на системную шину, могут являться *блок выборки команд* (Fetch Instruction Unit — FIU) и *буфер переупорядочивания запросов к памяти* (Memory Ordering Block — MOB). Первое из указанных устройств призва-

но обеспечить бесперебойное поступление инструкций в конвейер команд, а второе отвечает за перемещение операндов между исполнительными устройствами и ОП. Таким образом, система идентификации транзакций позволяет устройству шинного интерфейса разделить внутри процессора входной поток транзакций на поток команд и поток данных.

Запрос на чтение от процессора к ОП вызывает передачу целого пакета по системной шине в кэш-память верхнего уровня (для процессоров P6 — второго уровня). Размер такого пакета должен совпадать с размером блока, записываемого в кэш-память второго уровня, т. е. с размером одной линейки кэш-памяти. Выгрузка данных из кэш-памяти в ОП происходит также целыми линейками. Для процессора P6 линейка кэш-памяти составляет 32 байт, а для ее загрузки или выгрузки требуется выполнить передачу четырех информационных слоев. Помимо упрощения подсистемы кэш-памяти это позволяет использовать оперативную память, работающую по принципу кучности адресов.

Структура микросхемы синхронной динамической памяти показана на рис. 5. Хранилище данных в такой памяти выполняется в виде нескольких банков, каждый из которых содержит массивы запоминающих элементов типа 2DM. Из этого следует, что при чтении и записи любой порции данных происходит обращение сразу к некоторому количеству запоминающих элементов, называемому страницей или строкой DRAM-памяти. Выбор конкретной страницы из многих хранимых страниц выполняется с помощью части физического адреса — номера строки. Размер страницы, как правило, составляет 2^{11} или 2^{12} байт. Таким образом, любое обращение к динамической памяти вызывает обязательное чтение в специальный буфер 2 или 4 Кбайт данных, а выбор запрошенной информации в пределах страницы осуществляется по второй части адреса — номеру столбца. На рис. 6 показан пример трансляции 32-разрядного физического адреса в адреса выборки строки, банка столбца, который используется в некоторых контроллерах оперативной памяти.

Младшие разряды адреса задают смещение в передаваемом пакете и за ненадобностью могут не передаваться в ОП. Оставшаяся

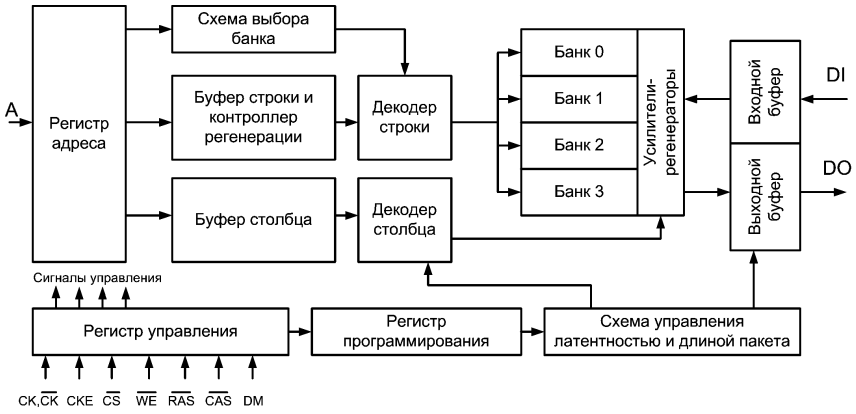


Рис. 5. Структура микросхемы синхронной динамической памяти

Рис. 6. Пример трансляции 32-разрядного физического адреса в адреса строки, банка и столбца

часть адреса делится блочно-циклическим способом на три части. Старшие разряды адреса определяют номер строки динамической памяти. Номер банка определяется средними разрядами, что позволяет ускорить последовательное чтение ОП за счет параллельного закрытия и открытия банков, а также за счет увеличения размера открытых страниц.

Важной особенностью подобного расслоения памяти является различное время доступа к данным, имеющим последовательные адреса. Если запрошенные данные относятся к открытой странице, обращение выполняется быстро. Если же страница запрошенных данных к этому моменту закрыта, доступ к таким данным требует больше времени. При этом количество одновременно открытых страниц может быть больше одной и, как правило, равно количеству банков памяти.

Кэш-память второго уровня (*Cache Level* $2 - L2$) служит для временного хранения команд и данных, востребованных в ходе работы процессора, и в современных моделях процессоров разме-

щается совместно с ними на одном кристалле. Повышение сложности процессорных ядер и технологические нормы производства кристаллов накладывают ограничение на размеры кэш-памяти L2. Тем не менее объем этой памяти достигает нескольких мегабайтов, а размер достигает в некоторых случаях 30 % общей площади кристалла.

Большая информационная емкость кэш-памяти L2 и потребность в минимальном времени поиска, вытеснения и размещения информации определяют наборно-ассоциативный способ организации кэш-памяти L2. Данная ассоциативная память представляет собой множество блоков (линеек кэш-памяти), объединенных в наборы. Ассоциативный поиск выполняется на основе физического адреса ячейки ОП, востребованной процессором. Для этого, как показано на рис. 7, адрес разделяется на три части.

Младшая часть адреса составляет смещение данных внутри линейки и используется в том случае, если кэш-память допускает чтение части линейки. Средняя часть используется для однозначного выбора набора, в котором допускается хранение ячеек ОП по запрошенному адресу. Старшая часть адреса (тег) служит для ассоциативного поиска информации в выбранном наборе. В связи с тем что каждому физическому адресу соответствует только один определенный набор, для поиска искомого тега необходимо небольшое число компараторов, равное числу линеек в наборе. Все линейки одного банка подключаются к общей шине, по которой тег выбранного набора передается на вход компаратора i.

При совпадении тегов выдается сигнал Hit[i], который разрешает выдачу линейки на шину данных. В случае отсутствия искомых данных все сигналы Hit[i] установлены в состояние логического нуля, что порождает информационный сигнал Hit. Обнаружив такой сигнал, устройство управления кэш-памятью может действовать по одному из двух алгоритмов:

- выдать запрос на загрузку данных по указанному адресу в память следующего уровня и в ожидании прихода данных блокировать все обращения к кэш-памяти;
- выдать запрос к памяти следующего уровня и, не дожидаясь получения данных, возобновить обращение к кэш-памяти.

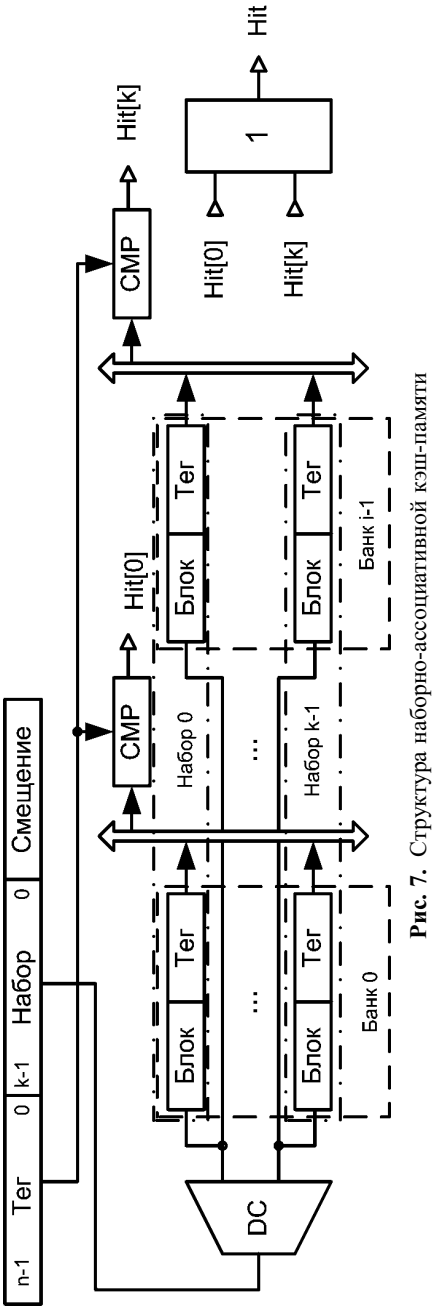


Рис. 7. Структура наборно-ассоциативной кэш-памяти

Первый вариант реализации называется блокируемой кэш-памятью и используется редко, так как не обеспечивает должного быстродействия подсистемы памяти. Действительно, при поступлении очередного запроса в тот момент, когда кэш-память блокирована, необходимо сохранить такой запрос в специальном буфере несмотря на то, что востребованные данные находятся в кэш-памяти и могут быть выданы. Более производительная кэш-память может быть построена по второму варианту и называется неблокируемой. В такой памяти предусматривается возможность блокирования доступа к банку, набору или линейке, ожидающим поступления данных, но для остальных наборов доступ разрешен. Это реализуется хранением совместно с линейкой информации о статусе I (Invalid), указывающем на факт ожидания.

Для кэш-памяти, допускающей модификацию линеек, характерна проблема выбора линейки — претендента для вытеснения. Наиболее выгодным с точки зрения временных затрат является выбор той линейки, которая никак не изменялась с момента ее загрузки в кэш-память: достаточно просто перезаписать в нее новые данные. Для обеспечения такой возможности каждая линейка должна хранить бит достоверности M (Modified), устанавливаемый при внесении любых изменений. Если такие линейки в наборе отсутствуют, используется другой алгоритм, обеспечивающий выбор наименее востребованной линейки. В большинстве случаев это реализуется с помощью алгоритма LRU (Least Recently Used), выбирающего претендента на выгрузку с помощью хранимого совместно с линейкой значения счетчика обращений. Часто используется упрощенная версия данного алгоритма: алгоритм Pseudo LRU, в котором анализируется информация о двоичном дереве поиска последней использованной линейки.

При однозначном определении набора по части адреса в один и тот же набор попадут ячейки, отстоящие друг от друга в ОП на определенное расстояние. Если число таких ячеек, востребованных в программе, превышает число линеек в наборе, неизбежно возникает конфликтная ситуация, называемая проблемой алиасинга. Выражается эта проблема в том, что вследствие необходимости вытеснения на запись очередной линейки тратится больше времени.

Полученные из кэш-памяти L2 команды и данные по отдельным линиям заносятся в *кэш-память команд первого уровня* (Instruction Cache Level 1 — L1I) и *кэш-память данных первого уровня* (Data Cache Level 1 — L1D). Раздельное хранение позволяет оптимизировать структуру обоих устройств в соответствии с выполняемыми ими функциями. Так, кэш-память команд первого уровня предназначена исключительно для чтения и не позволяет модифицировать свое содержимое. Попытка изменить коды команд, хранимых в кэш-памяти L1I, приводит к их модификации в оперативной памяти и загрузке уже измененных кодов. Такая политика изменения содержимого называется сквозной (Write True write policy — WT).

В связи с тем что доступ к оперативной памяти могут осуществлять несколько устройств (процессоров или контроллеров прямого доступа к памяти), необходимо своевременно модифицировать копии данных. Эта задача носит название проблемы когерентности кэш-памяти. Очевидно, что политика WT решает данную проблему, однако существенно замедляет обработку данных из-за большого количества запросов на запись в оперативную память. Сократить их количество позволяет политика обратной записи (Write Back policy — WB), при которой процессор, владеющий кэш-памятью, отслеживает обращения в доступную ему оперативную память. При этом ему необходимо обнаружить все факты изменения кэшируемых данных в оперативную память и модифицировать их в кэш-памяти, а также вовремя выгрузить востребованные иными устройствами данные в оперативную память в случае, когда они были модифицированы в кэш-памяти. Такая политика активно используется для кэш-памяти, хранящей данные, в частности, кэш-памяти L2 и L1D.

Другой особенностью кэш-памяти является то, что ассоциативный поиск информации выполняется по физическому адресу оперативной памяти, в то время как в поступающих в процессор командах указывается логический адрес. Для преобразования требуется обращение к таблице страниц, хранящей информацию об их физическом месте расположения. Это действие следует выполнять для всех команд, содержащих адреса оперативной памяти, по-

этому ускорение преобразования достигается благодаря хранению части таблицы страниц внутри процессора в специальной кэш-памяти — *буфере быстрого страничного преобразования* (*Translation Lookaside Buffer — TLB*). Однако нередко требуемая информация в TLB отсутствует, тогда процессору необходимо сначала загрузить из оперативной памяти часть таблицы страниц и только потом выполнить преобразование. При этом младшая часть адреса — смещение данных внутри страницы — остается неизменной.

Для наборно-ассоциативной кэш-памяти можно записать следующее соотношение: $\text{Р} = \text{Л} \times \text{А} \times \text{Н}$, где Р — размер кэш-памяти в байтах; Л — размер линейки в байтах; А — ассоциативность; Н — количество наборов. Шаг между ячейками, гарантирующий их попадание в один набор, равен $\text{Ш} = \text{Н} \times \text{Л} = \text{Р}/\text{А}$. Разрядность смещения для страниц размером 4 Кбайт составляет 12 бит, что при условии $\text{Ш} < 2^{12}$ позволяет процессору определить номер набора в кэш-памяти до завершения страничного преобразования (рис. 8). Поэтому в отдельных моделях процессоров для ускорения выборки из кэш-памяти выполняется предварительный поиск данных по младшим разрядам логического адреса. После окончания преобразования результаты такой выборки уточняются.

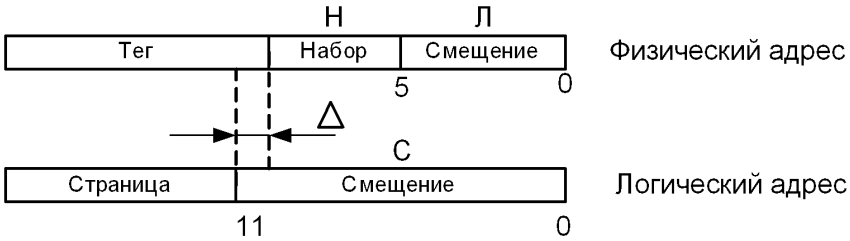


Рис. 8. Проблема выборки данных из кэш-памяти по физическому адресу

В некоторых моделях процессоров кэш-подсистема несколько отличается от рассмотренной ранее. Так, процессор AMD Athlon хранит в кэш-памяти L1I частично декодированные команды. В процессоре Pentium 4 в кэш-памяти хранит полностью декодированные команды — микрокоманды, записанные в виде линейных участков (трасс).

1.3. Адресная подсистема

В функции подсистемы вычисления адресов процессоров P6 входят:

• обеспечение виртуализации адресного пространства, реализуемое *буферами быстрого страничного преобразования*, *блоком связи с памятью* и *блоком вычисления адреса следующей команды*;

• предсказание направления ветвления, реализуемое *буфером меток перехода* и *блоком вычисления адреса следующей команды*.

Рассмотрим назначение этих устройств более подробно.

Блок вычисления адреса следующей команды (Prediction Unit — PU) служит для определения адреса очередной команды и выдачи его в блок FIU. Как уже отмечалось, конвейерная обработка предполагает непрерывное поступление команд, исполнение которых происходит лишь после декодирования и подготовки операндов. Очевидно, что ожидание исполнения команд ветвления существенно замедлило бы обработку, поэтому в процессорах активно используются способы предсказания направления ветвления.

Ускоренное по сравнению с иными командами декодирование команд перехода обеспечивается благодаря предварительному сканированию блока, считанного из кэш-памяти команд L1I блока. Указанное действие позволяет минимизировать потери времени. Далее при обнаружении команды безусловного перехода (команды вызова подпрограмм, возврата из процедур и прерываний, прямых переходов и т. п.) быстро вычисляется логический адрес очередной команды.

Более существенные задержки в работе конвейера вызывают команды условного перехода, что связано с невозможностью вычисления условия перехода до окончания выполнения всех предшествующих команд. Частичное решение этой проблемы достигается применением различных способов предсказания. Первый из них — статическое определение направления ветвления — основан на анализе предварительно собранных сведений. Такая информация внедряется в программный код на этапе компиляции и может быть получена, например, на основе заблаговременно выявленной неравномерности исходов условных переходов программы. Однако даже

при использовании большого количества статистических сведений эффективность такого способа существенно ниже, чем у динамического предсказания.

Динамическое предсказание основано на использовании предыстории переходов, содержащей информацию об исходах выполнения команд условного перехода, поступавших в процессор ранее. Кроме того, применяются известные статистические сведения. Например, переход в сторону убывания адресов (переход назад) целесообразно предсказывать «совершенным», в то время как переход в сторону увеличения адресов (переход вперед), наоборот, целесообразно не предсказывать. Это, в частности, несколько ускоряет обработку циклов с постусловием по сравнению с циклами с предусловием. Кроме того, в зависимости от сложности реализации алгоритмы динамического предсказания позволяют выявить в программе шаблонные кодовые конструкции: циклы, попеременные переходы, маловероятные переходы и др. Статистическая информация накапливается в специальной ассоциативной памяти — *буфере меток перехода* (Branch Target Buffer — BTB), обращение к которой выполняется по целевому программному адресу.

Для своевременной выборки данных, востребованных в командах, используется механизм ранней спекулятивной выборки, который реализуется с помощью *устройства связи с памятью*. Когда на конвейер поступает очередная команда, требующая обращения к памяти для получения операнда, устройство декодирования выдает специальную микрокоманду загрузки (load), которая может быть выполнена еще до завершения исполнения предыдущих команд. Этот механизм, по сути, приравнивает действия по загрузке операндов к отдельной операции, что характеризует организацию ядра микропроцессора, как архитектуру RISC (Reduced Instruction Set Computer).

В набор команд суперскалярных процессоров также включают команды принудительной программной предвыборки, позволяющие умелому программисту оптимизировать вычислительный процесс, сочетая непосредственные вычисления с обращением к памяти. В табл. 1 описаны команды программной предвыборки и их назначение для процессоров Pentium III и Pentium 4.

Таблица 1

Команды программной предвыборки

Команда	Pentium III (32 байта)	Pentium 4 (128 байт)	Примечание для P6
prefetchNTA	Загрузка только в L1D. В L2 не загружаются	Загрузка в L2. В L1D не загружаются	Загрузка в кэш-память первого уровня для немедленного использования
prefetch0	Загрузка в L1D и L2	Загрузка только в L2. В L1D не загружаются	Загрузка в кэш-память всех уровней
prefetch1	Загрузка только в L2. В L1D не загружаются	Загрузка в L2. В L1D не загружаются	Загрузка в кэш-память всех уровней, кроме нулевого (только L2)
prefetch2	Загрузка только в L2. В L1D не загружаются	Загрузка в L2. В L1D не загружаются	Загрузка в кэш-память всех уровней, кроме нулевого и первого (только L2)

Декодированные команды переходов и информация о результате предсказания совместно с другими командами поступают на следующие стадии конвейера. После вычисления всех использованных в таких командах операндов (например, флагового регистра) команда перехода поступает на специальное *устройство арифметики переходов*, которое проверяет правильность предсказания. В случае правильного предсказания дополнительных действий не требуется. При ложном предсказании устройство арифметики переходов инициирует сброс конвейера, заключающийся в выполнении следующих действий:

- восстановление состояния физических регистров на момент последней актуальной команды перехода (такое действие в современных процессорах сводится к стиранию содержимого нескольких регистров);
- изменение статистической информации о неправильно предсказанном переходе;
- запрос очередной команды правильной ветви исходной программы.

1.4. Подсистемы декодирования, переупорядочивания и диспетчеризации

Подсистема декодирования служит для определения последовательности микрокоманд, необходимых для реализации поступающей последовательности инструкций программы, и состоит из предекодеров и декодеров инструкций, а также памяти микропрограмм и блока переименования регистров.

Команды из кэш-памяти команд первого уровня L1I помимо *блока вычисления адреса следующей команды* поступают в блок декодирования, где выполняется предобразование каждой поступившей инструкции в последовательность микрокоманд. Микрокоманды, соответствующие одной инструкции, называются микропрограммой и хранятся в памяти микропрограмм. В современных микропроцессорах для генерации микропрограмм используется несколько декодеров, способных параллельно выдавать на последующие стадии конвейера целые группы микроопераций. Так, в процессорах P6 реализованы три декодера, первый из которых может обработать любую инструкцию и способен генерировать до четырех микрокоманд за один такт, в то время как два других позволяют дешифрировать только простые инструкции, каждая из которых преобразуется в одну микрокоманду. Таким образом, максимальная производительность декодеров в процессорах P6 достигается при компоновке программного кода в блоки по 16 байт, в которых на первом месте размещается сложная инструкция, а за ней следуют две простые инструкции. Иное размещение команд замедлит процесс декодирования.

Сложный формат команд, характерный для CISC-процессоров, приводит к существенному усложнению декодеров. Например, для набора команд процессоров x86 характерно применение префиксов команд, изменяющих размерность операндов и адресов, указывающих на необходимость повторения команд или меняющих сегмент по умолчанию. Наличие префикса в инструкции существенно замедляет декодирование. Для устранения проблемы, связанной с различием длины команд (от 1 до 15 байт), требуется применять дополнительные сложные устройства разметки инструкций, обеспечивающих ускоренный поиск кодов операций и определение их длины.

Преобразование машинных инструкций в последовательность микрокоманд, исполнение которых может происходить переупорядоченно (т. е. в порядке, отличном от предписанного программой), связано с необходимостью выявления и устранения взаимосвязи команд по данным. Так, если две инструкции программы используют одни и те же ячейки памяти, могут возникнуть четыре типа конфликтов по данным (рис. 9).

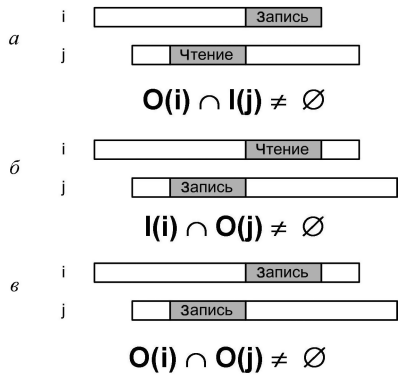


Рис. 9. Конфликт по данным типа «чтение после записи» (*а*), «запись после чтения» (*б*), «запись после записи» (*в*), где O(i) — множество ячеек, изменяемых командой i, I(j) — множество ячеек, читаемых командой j

1. В исходной программе первая команда записывает результат в ячейку k, после чего вторая команда должна прочитать содержимое ячейки k. При переупорядоченном исполнении возможна ситуация, когда вторая команда прочитает содержимое ячейки k до того, как туда будет записан результат работы предшествующей команды. Такой конфликт называется «чтением после записи».

2. Первая команда в исходной программе читает содержимое ячейки k, а следующая команда изменяет ее значение. Переупорядочивание микрокоманд может привести к записи результата выполнения второй команды до чтения операнда в первой команде (конфликт типа «запись после чтения»).

3. Тип конфликта «запись после записи» связан с переупорядоченной записью в ячейку k: вторая команда выполнит запись результата раньше первой команды, что приведет к сохранению старого значения.

4. Тип конфликта «чтение после чтения» при неупорядоченном исполнении не вызывает искажения результатов.

Для выявления и устранения конфликтов типов «чтение после записи» и «запись после чтения» применяется механизм переименования, основанный на использовании *регистров замещения*. Обязанности программных регистров процессора, указываемых в командах, в каждый момент времени может исполнять любой регистр замещения. В связи с этим при декодировании очередной инструкции необходимо иметь полную информацию о текущем назначении инструкций. Далее, если в команде предусмотрено сохранение результата, для него выделяется один из свободных регистров замещения, после чего псевдоним выделенного регистра отмечается в специальной таблице — *таблице регистровых псевдонимов*. Все последующие команды, использующие полученный результат, должны ожидать его вычисления и сохранения (иными словами, достоверности) во вновь предоставленном регистре замещения, что предусматривает хранение специального бита достоверности для каждого регистра. Указанный механизм обеспечивает возможность переупорядоченного исполнения микрокоманд, так как разрешает обработку данных только в том случае, когда все востребованные в микрокоманде операнды достоверны.

Рассмотрим пример переупорядоченного исполнения последовательности команд:

```
MUL BX      ; умножить регистр BX на регистр AX,
            ; результат поместить в DX:AX,
ADD AX,BX   ; к регистру AX прибавить BX,
SUB BX,2    ; уменьшить регистр BX на 2.
```

При попытке выполнить переупорядоченную обработку возникают следующие конфликты:

- вторая команда вызывает запись AX до того, как выполняется чтение в первой команде (конфликт «запись после чтения»);
- третья команда изменяет BX раньше, чем в первой или второй команде осуществляется чтение (конфликт «запись после чтения»);
- вторая команда вызывает чтение AX до его записи, выполняемой в первой команде (конфликт «чтение после записи»).

• вторая команда вызывает запись AX быстрее, чем выполняется запись в первой команде (конфликт «запись после записи»).

Приме́ним описанную ранее процедуру выделения регистров замещения: для первой команды выделим регистры AX′ и DX′, для второй — регистр AX″, для третьей — регистр BX′. В результате получим код программы, в котором будут устранены конфликты типов «чтение после записи» и «запись после чтения». Конфликт типа «чтение после записи» по данным, находящимся в регистре AX′, может быть устранен с помощью бита достоверности содержимого регистра. Полученная последовательность микрокоманд позволяет одновременно исполнить первую и третью команды, после чего может быть выполнена вторая команда:

```
MUL BX      ; умножить регистр BX на регистр AX,
            ; результат поместить в DX′:AX′,
ADD AX,BX   ; к достоверному содержимому регистра AX′
            ; прибавить BX, результат поместить в AX″,
SUB BX,2    ; уменьшить BX на 2, результат поместить в BX′.
```

Описанный выше способ позволяет эффективно устранить конфликты по данным, находящимся в регистрах, а также обеспечивает индикацию готовности операндов. Однако такой способ не позволяет устранить конфликты по данным, находящимся в памяти, так как любая неупорядоченная запись в память для команды, находящейся в неправильно предсказанной ветви программы, неминуемо приведет к ошибочным результатам. Поэтому при обнаружении конфликтов по данным, расположенным в оперативной памяти, микрокоманды запись результатов в память (store) выполняется упорядоченно.

Процесс выделения регистров замещения выполняется после декодирования команд в *блоке переименования регистров*. Для реализации процедур проверки достоверности всех микрокоманд, находящихся на конвейере, используется сложное устройство, называемое *буфером переупорядочивания микрокоманд* (*Reorder Buffer — ROB*). Оно представляет собой кольцевой регистровый файл, содержащий в каждой позиции информацию о коде микрокоманды, номера регистров замещения исходных операндов и результата (рис. 10).

В некоторых моделях процессоров поля операндов и результата

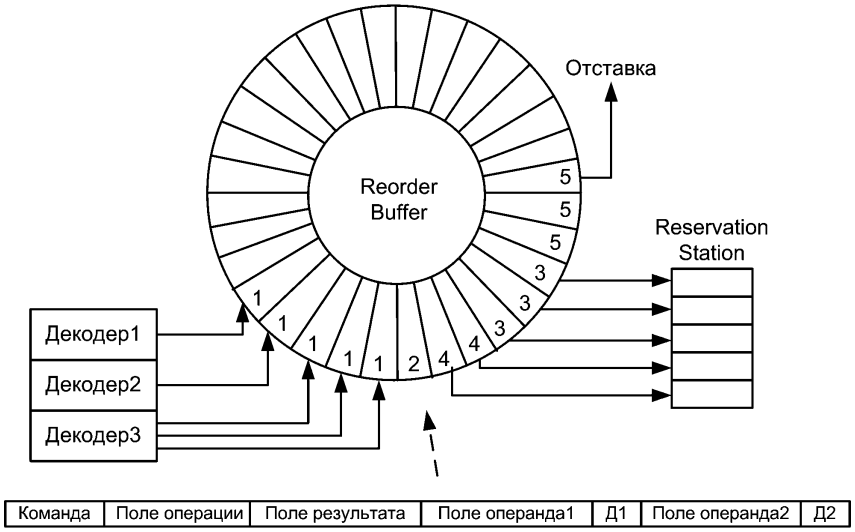


Рис. 10. Механизм неупорядоченного исполнения и упорядоченной отставки микрокоманд в ROB

в позициях блока ROB совмещены с регистрами замещения. Микрокоманды в блоке ROB могут находиться в одной из следующих стадий:

1) не готова к исполнению;
2) готова к исполнению;
3) исполняется;
4) исполнена и ожидает отставки;
5) находится в процессе отставки.

Для записи и удаления микрокоманд в блоке ROB предусмотрены указатели на первую и последнюю микрокоманды. Запись новых микрокоманд выполняется декодерами в конец блока ROB при наличии свободных ячеек буфера. При этом последовательность микрокоманд в блоке ROB строго соответствует исходной программе. В каждом такте работы выполняется сканирование блока ROB и выявляются микрокоманды, готовые к исполнению, т. е. имеющие достоверные операнды. Если такие микрокоманды найдены, они копируются в *буфер микрокоманд*, *готовых к исполнению* (*Reservation Station — RS*). Исполненные микрокоманды удаляются из блока ROB

специальным устройством, называемым *блоком удаления и восстановления*, который выполняет отставку в следующих случаях:

- все микрокоманды данной команды успешно исполнены;
- все микрокоманды ранее поступивших команд выполнены и успешно отставлены.

Команды должны покидать конвейер в исходном порядке, заданном в программе, по следующим причинам:

- наличие команд условного перехода и механизм предсказания приводят к необходимости сброса конвейера в случае неправильно выполненного предсказания ветвления. Таким образом, упорядоченная отставка гарантирует возможность восстановления всех исходных операндов, действительных для микрокоманды условного перехода;
- возможность возникновения конфликта «запись после записи». Очевидно, что конфликт «запись после записи» может быть разрешен с помощью описанного выше механизма неупорядоченного исполнения при соблюдении исходного порядка записи результатов;
- возникновение исключительных ситуаций, требующих незамедлительного прекращения неупорядоченного исполнения, таких, как прерывания или ошибки при исполнении;
- наличие команд упорядоченного исполнения, загрузки и выгрузки, таких, как CPUID, IRET и другие, запрещающие заблаговременное исполнение последующих микрокоманд и выполняемые только после исполнения предыдущих.

В современных процессорах для обеспечения записи в соответствии с исходным порядком команд применяется механизм упорядоченной отставки, по которому микрокоманды записи результатов совместно с данными сохраняются в специальном буфере до тех пор, пока соответствующая им команда не будет отставлена *блоком удаления и восстановления*. Для хранения микрокоманд записи используется специальное устройство, называемое *буфером переупорядочивания запросов к памяти.* С его помощью значительно повышается эффективность подсистемы памяти, так как отложенная запись результатов в память позволяет сократить влияние латентности памяти на ход вычислений в конвейере команд.

Тем не менее возможны случаи, когда программисту требуется обеспечить определенный порядок загрузки и выгрузки независимых данных (зависимые запросы упорядочиваются процессором автоматически). Например, для современных процессоров фирмы Intel характерна возможность совмещения адресных пространств ввода/вывода и ОП. Поэтому при обращении к диапазону физических адресов, соответствующему устройствам ввода/вывода, особенно важно обеспечить заданный порядок исполнения микрокоманд загрузки и выгрузки. Для этого в систему команд современных процессоров включают специальные инструкции упорядочивания загрузки и выгрузки (табл. 2), а также инструкции принудительного освобождения кэш-памяти. Их использование позволяет программисту существенно влиять на работу подсистемы памяти и оптимизировать работу программ. Другим способом строгого упорядочивания запросов к памяти для процессоров Intel является задание некэшируемой разметки физического адресного пространства ввода/вывода в регистрах *MTRR* (*Memory Type Range Registers*).

Таблица 2

Команды управления загрузкой и выгрузкой в процессорах P6

Команда	Назначение	Примечание
lfence	Упорядочивание загрузки	Команда позволяет управлять загрузкой, запрещая переупорядочивать микрооперации загрузки до данной команды с микрооперациями после данной команды
sfence	Упорядочивание выгрузки	Команда позволяет управлять выгрузкой, запрещая переупорядочивать микрооперации выгрузки до данной команды с микрооперациями после данной команды
mfence	Упорядочивание загрузки и выгрузки	Команда позволяет управлять загрузкой и выгрузкой, запрещая переупорядочивать микрооперации загрузки и выгрузки до данной команды с микрооперациями после данной команды

Таким образом, *буфер переупорядочивания запросов к памяти* реализует следующие функции:

• буферизация поступающих с конвейера запросов чтения и записи;

• переупорядочивание запросов с целью оптимизации работы подсистемы памяти. Актуальность использования данного механизма состоит в том, что целевые адреса нескольких последовательных запросов могут относиться к одной и той же линейке кэш-памяти или к одной и той же странице оперативной памяти. Также выгодно группировать однотипные запросы, т. е. выполнять чтение и запись не по отдельности, а целыми пулами;

• исполнение микрокоманд упорядоченной загрузки и выгрузки;

• устранение выявленных конфликтов типа «запись после записи».

Буфер микрокоманд, готовых к исполнению, служит для временного хранения микрокоманд и их операндов перед направлением их в освободившиеся исполнительные устройства. Следует отметить, что буфер микрокоманд, готовых к исполнению, может быть реализован на основе блока ROB, что требует выбора на исполнение одной микрооперации из нескольких готовых к исполнению. Отправка микрокоманд к исполнительным устройствам выполняется через специальные буферы, называемые *портами запуска*. Число портов и способ назначения исполнительных устройств определенным портам различаются в разных моделях процессоров. В процессорах P6 предусмотрено пять портов запуска (рис. 11):

• к порту 0 подключены целочисленное арифметико-логическое устройство 0, блок FPU, устройство для SSE-умножения и деления,

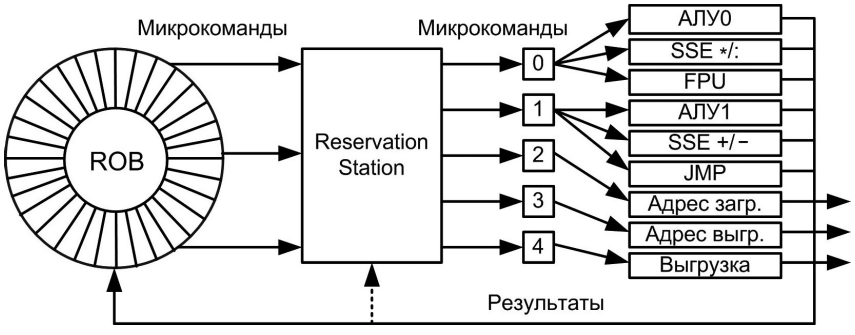


Рис. 11. Механизм исполнения микрокоманд в процессоре P6

а также устройство выполнения логических SSE-операций (за исключением сдвига);

• к порту 1 подключены целочисленное арифметико-логическое устройство 1, устройство для SSE-сложения и выполнения логических SSE-операций и сдвига, устройство арифметики переходов;

• к порту 2 подключено устройство вычисления адресов загружаемых операндов;

• к порту 3 подключено устройство вычисления адресов выгружаемых результатов;

• к порту 4 подключено устройство выгрузки результатов, принимающее результаты из блока ROB и направляющее их в блок переупорядочивания запросов к памяти совместно с адресом, поступающим из блока вычисления адреса.

Большая связанность операндов в блоке ROB позволяет применить более быстродействующий механизм подготовки микрокоманд к исполнению. Так, в процессоре Pentium 4 микрокоманды поступают на исполнение уже тогда, когда участвующие в них операнды еще находятся в стадии вычисления. Как только операнды будут вычислены, они поступают непосредственно в очередь ожидающих их микрокоманд.

1.5. Подсистема исполнения

Подсистема исполнения состоит из устройств следующих типов:

• целочисленных арифметико-логических устройств;

• устройств адресной арифметики;

• устройств обработки чисел с плавающей запятой;

• устройств выполнения целочисленных MMX-операций;

• устройств векторных вычислений над числами с плавающей запятой.

Целочисленные арифметико-логические устройства (*АЛУ*). Процессор P6 содержит два целочисленных АЛУ, что позволяет выполнять две операции за один такт работы процессора. При этом операция сдвига может быть выполнена только на одном из АЛУ, т. е. латентность сдвига составляет один такт при производительности, равной одной операции за один такт. Латентность операций

сложения и вычитания составляет один такт при производительности, равной двум операциям за один такт, что тем не менее часто оказывается недостаточным.

Для ускорения выполнения арифметических операций с 32- и 64-разрядными операндами в последующих моделях процессоров Intel используется конвейеризация АЛУ совместно с увеличением тактовой частоты работы ступеней. Для этого применяется разделение на стадии, в каждой из которых используются АЛУ с меньшей разрядностью. Этот принцип поясняется на рис. 12. Младшие 16 разрядов числа суммируются (вычитаются) в первом такте, в то время как старшие разряды и перенос сохраняются в регистре. Старшие разряды чисел A и B обрабатываются в следующем такте одновременно с обработкой младших разрядов следующих операндов.

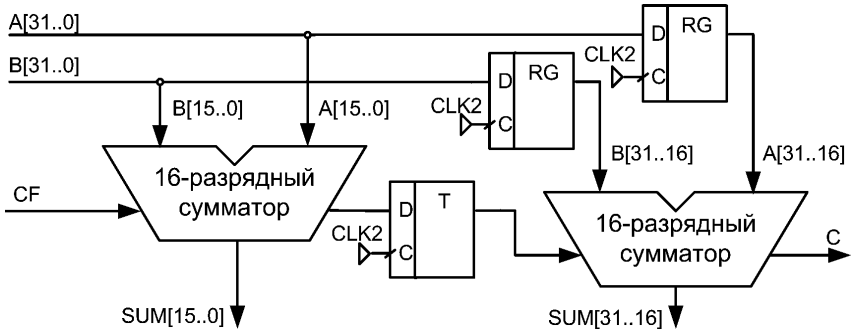


Рис. 12. Конвейеризация операций сложения и вычитания в процессоре Pentium 4

Устройства целочисленного умножения в современных процессорах основаны на древовидных умножителях. Рассмотрим структуру четырехразрядного устройства умножения по схеме Уоллеса. Для получения восьмиразрядного произведения необходимо получить 16 разрядов частных произведений как конъюнкции вида $a_i b_j$ всех разрядов множителей A и B. После этого разряд произведения с номером k определяется как сумма разрядов частных произведений $a_i b_j$ с учетом переносов из младших разрядов, причем $k = i + j$. Данная схема поясняется на рис. 13.

Древовидные n-разрядные устройства умножения состоят из конъюнкторов, сумматоров и полусумматоров. Структура

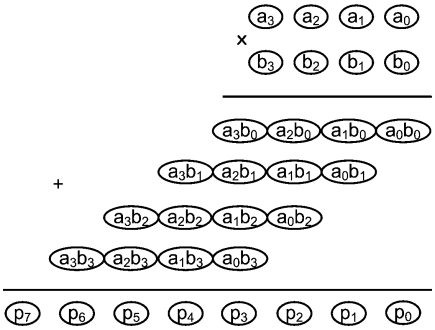


Рис. 13. Общая схема умножения

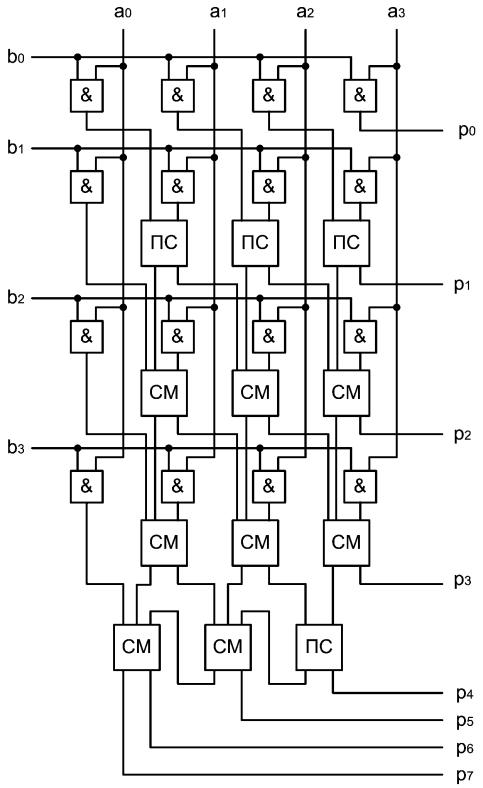


Рис. 14. Схема четырехразрядного древовидного устройства умножения (ПС — полусумматор, СМ — полный сумматор)

четырехразрядного устройства умножения, построенного по схеме Уоллеса, показана на рис. 14. Умножение состоит из нескольких стадий: определения n^2 разрядов частных произведений, сложения разрядов частных произведений для накопления суммы частных произведений и определения переносов, последовательного сложения суммы частных произведений и переносов на заключительной стадии. Указанные действия занимают значительное время, так как для определения старшего разряда произведения требуется передать значение переноса через большое число сумматоров и полусумматоров (для представленного устройства умножения: четыре сумматора и два полусумматора). Поэтому разработчики стремятся использовать схемы ускорения суммирования. Например, при реализации последней стадии для устройства умножения, схема которого приведена на рис. 14, можно использовать более быстродействующий четырехразрядный сумматор с параллельным переносом.

Стадийность процесса умножения способствует его конвейеризации. Для этого устройство умножения реализуется в виде нескольких ступеней, между которыми размещаются дополнительные буферные регистры (рис. 15).

Латентность операции умножения для процессоров P6 составляет пять тактов при производительности, равной одной операции за два такта. Операция деления выполняется за большее число тактов в зависимости от разрядности операндов (до 80 тактов).

Устройство обработки чисел с плавающей запятой (ЧПЗ) процессоров P6 функционирует в соответствии со стандартом IEEE 754. В документе оговорены три формата представления чисел с плавающей запятой: короткий, длинный и расширенный. Во всех форматах порядок числа сохраняется в смещенном коде, а мантисса является нормализованной. Короткий и длинный форматы не предусматривают явного хранения старшего разряда мантиссы (для нормализованного числа он всегда равен единице). Характеристики форматов приведены в табл. 3.

Операции над ЧПЗ состоят из трех этапов.

1. Подготовительный этап. На этом этапе происходит разделение числа в соответствии с форматом на группы разрядов, представ-

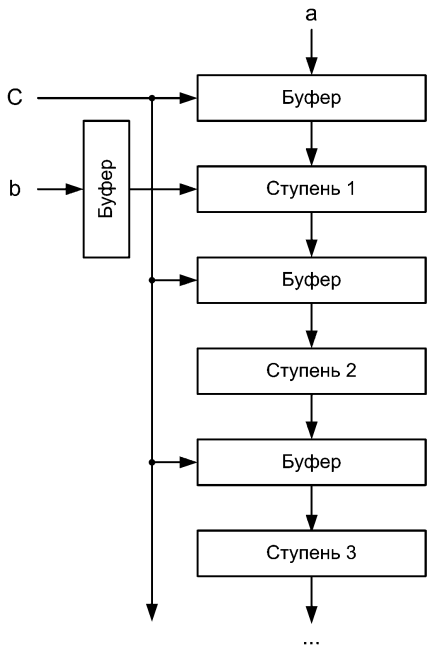


Рис. 15. Конвейеризация устройства умножения

Таблица 3

Сравнение форматов ЧПЗ стандарта IEEE 754

Формат	Длина числа	Длина мантиссы	Длина порядка	Смещение порядка	Диапазон чисел
Короткий	32	24	8	$+127$	$10^{-38}\ldots10^{+38}$
Длинный	64	53	11	$+1023$	$10^{-308}\ldots10^{+308}$
Расширенный	80	64	15	$+16\,383$	$10^{-4932}\ldots10^{+4932}$

ляющих собой мантиссу, порядок и знак числа. Также выполняется проверка на специальные числовые значения (нуль, бесконечность и др.).

2. Выполнение операции. На этом этапе анализируются исходные операнды и определяются разряды результата. В зависимости от типа операции второй этап состоит из приведения порядков, определения знака результата, определения мантиссы результата, определения порядка результата. Также выполняются проверки

на переполнение, потерю значимости мантиссы, потерю значимости порядка, неточности, деления на нуль.

3. Заключительный этап. На этом этапе выполняется проверка на равенство результата специальному числовому значению. Если результат допусти́м для данного типа операции, происходит нормализация результата, в ходе которой возможно возникновение исключительной ситуации. Поэтому на заключительном этапе также выполняются проверки на переполнение, потерю значимости мантиссы и порядка.

Устройство выполнения целочисленных MMX-операций (Multi-Media eXtensions) и *SSE-операций* (Streaming SIMD Extension) предназначены для ускорения приложений, ориентированных на выполнение однотипных действий с большими массивами целочисленных и действительных данных. С данными такого типа обычно работают мультимедийные, графические и коммуникационные программы. Команды для реализации указанных действий используют дополнительные форматы представления чисел, показанные на рис. 16.

Так, в SSE-расширении вводится новый упакованный 128-разрядный тип данных, который состоит из четырех ЧПЗ одинарной точности. Для выполнения микроопераций над ЧПЗ и векторной арифметики в процессоре P6 используются 80-разрядные регистры замещения. Для размещения 128-разрядных операндов требуется выделить два таких регистра, что существенно сокращает производительность процессора при выполнении SSE-команд. В последующих модификациях процессоров Intel (NetBurst, Core) данный недостаток устранен, а размер регистров замещения расширен до 128 разрядов. Латентность операции SSE-умножения составляет три такта при производительности, равной одному такту. Латентность операции SSE-сложения и вычитания двух упакованных двойных слов составляет один такт при возможности обработки одной операции за один такт. Особенностью процессоров P6 является то, что устройства SSE-умножения/деления и SSE-сложения/вычитания/сдвига подключены к разным портам запуска, в связи с чем они могут исполнять команды параллельно.

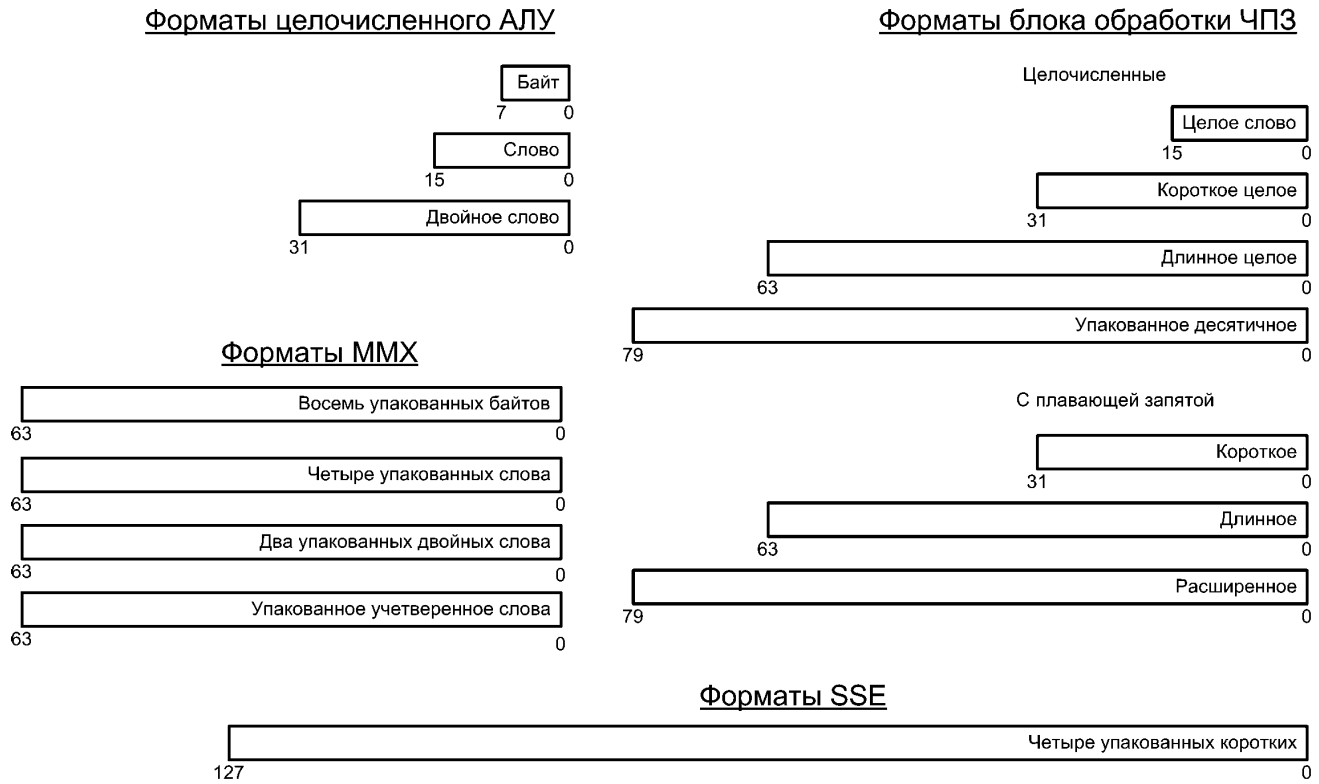


Рис. 16. Числовые форматы процессоров семейства x86

2. Микроархитектура современных суперскалярных процессоров

2.1. Особенности микроархитектуры процессоров NetBurst

Семейство NetBurst было разработано фирмой Intel и впервые реализовано в процессорах Pentium 4 в ноябре 2000 г. По сравнению с процессорами предшествующего семейства P6, в NetBurst был внесен ряд существенных изменений [3–5, 11]:

- использован кэш трасс (trace-cache) команд первого уровня, хранящий декодированные команды в виде последовательных микрокоманд (трасс);
- усовершенствована система декодирования, что совместно с введением кэш трасс позволило устранить «узкое место» конвейера семейства P6;
- усовершенствован механизм переименования и выделения регистров, увеличены их число и разрядность;
- введен механизм ранней спекулятивной диспетчеризации;
- усовершенствованы алгоритмы предсказания направления ветвления;
- удвоена частота работы арифметико-логических устройств;
- изменена система очередей к исполнительным устройствам;
- увеличены размеры буферов чтения и записи;
- изменена структура кэш-памяти данных первого уровня;
- усовершенствован блок шинного интерфейса в связи с применением более производительной системной шины.

Функциональная схема процессоров семейства NetBurst представлена на рис. 17 и соответствует материалам, изложенным в [3]. Следует отметить, что данная схема менее детализирована по сравнению с ранее приведенной на рис. 3 схемой процессоров семейства P6.

Кэш трасс представляет собой специальное адресно-ассоциативное запомнинающее устройство, хранящее декодированные команды x86. Это устройство позволяет многократно использовать результаты декодирования команд цикла и команд обработки строк, а также обеспечивает упаковку микрокоманд в группы для

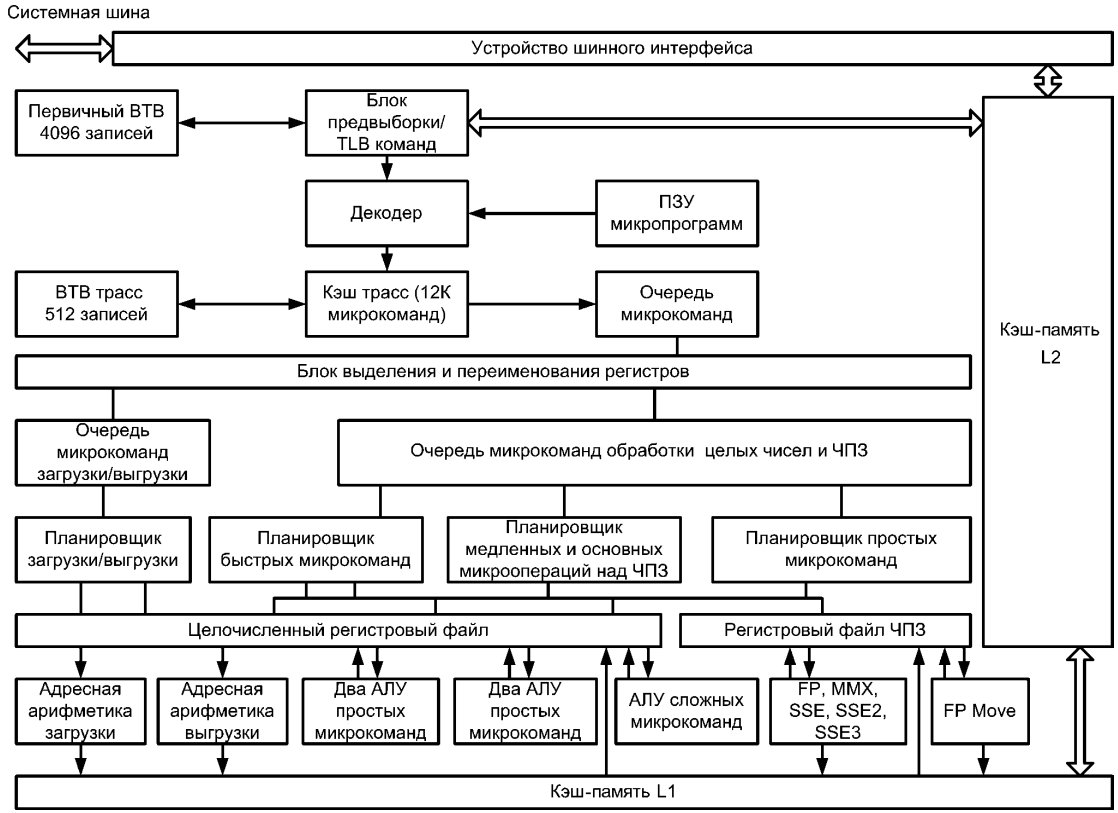


Рис. 17. Функциональная схема процессоров семейства NetBurst

последующей диспетчеризации. Кроме того, становится возможным указание позиций кэш трасс в качестве целевых микрокоманд для инструкций переходов и соответственно использование предсказания направления ветвления по позициям кэш трасс. Для этого используется буфер меток перехода в кэш трасс (Trace Branch Target Buffer — TBTB).

Размер кэш трасс составляет 12 К ячеек для хранения микрокоманд, организованных в 2048 блоков по шесть ячеек. Трасса образуется из нескольких блоков, организованных в двунаправленный список: каждый блок хранит номера следующего и предыдущего блоков трассы. Число блоков в трассе при этом не превышает 64. Чтение кэш трасс выполняется по три ячейки за один такт, т. е. блок читается за два такта. Все микрокоманды одной команды обязательно размещаются в одном блоке, а число микрокоманд для команд перехода ограниченно двумя на один блок.

Существенным отличием микроархитектуры NetBurst от микроархитектуры P6 является иная стратегия выделения и переименования регистров. Для микроархитектуры P6 характерно однозначное соответствие входов буфера переупорядочивания блока ROB регистрам замещения (всего 40), а также наличие архитектурного регистрового файла, обновляемого при отставке команд. В процессорах NetBurst входы блока ROB (всего 126) не соответствуют регистрам замещения (всего предусмотрено 128 целочисленных регистров и 128 для ЧПЗ), а выделяются для микрокоманд независимо. При этом состояние архитектурных регистров сохраняется в последних использованных регистрах замещения (регистры замещения разделены на целочисленный регистровый файл и регистровый файл ЧПЗ).

Блок диспетчеризации NetBurst также претерпел существенные изменения. Очереди и планировщики разделены на две части: очереди и планировщики арифметико-логических микрокоманд, а также очередь и планировщик микрокоманд загрузки/выгрузки. Предусмотрены независимые быстрые и медленные планировщики, выдающие соответственно по две и по одной микрокоманде за один такт, оптимизирующие продвижение операндов к быстрым и медленным исполнительным устройствам. Микропроцес-

соры NetBurst обеспечивают выдачу трех микрокоманд за один такт.

Как было отмечено в разд. 1, в процессорах с микроархитектурой NetBurst используется механизм ранней спекулятивной диспетчеризации, заключающийся в передаче микрокоманды в очереди планировщиков до получения задействованных в них операндов. Ожидание выполняется уже в очередях планировщиков, а подготовленные операнды заносятся в целочисленный и ЧПЗ регистровые файлы (так называемая петля реплея). Благодаря этому удается сократить время исполнения связанных по данным микрокоманд.

Кэш-пямать данных первого уровня имеет малый объем и оптимизирована по быстродействию. Для моделей с 8 Кбайт четырехассоциативной кэш-памяти L1D латентность составляет два такта для целочисленных данных и девять тактов для ЧПЗ, а для моделей с 16 Кбайт восьмиассоциативной кэш-памятью латентность составляет четыре такта для целочисленных данных и 12 тактов для ЧПЗ. Латентность чтения кэш-памяти L2 составляет от семи тактов для моделей с 256–512 Кбайт кэш-памяти L2 до 18 тактов для моделей с 1 Мбайт кэш-памяти L2.

Процессоры с микроархитектурой NetBurst имеют ряд недостатков, обусловленных:

- проблемой алиасинга при обращении к кэш-памяти L1D с адресным расстоянием 64 Кбайт;
- половинной частотой запуска скалярных операций SSE;
- неэффективной реализацией операций сдвига и целочисленного деления;
- высокой латентностью инструкций, использующих флаг переноса;
- малым размером кэш-памяти L1D и высокой латентностью доступа к кэш-памяти L2.

2.2. Микроархитектура процессоров AMD64

Архитектура 64-разрядных процессоров семейства AMD64 (Athlon 64, Opteron 64, Turion 64) во многом схожа с архитектурами предшествующих 32-разрядных моделей AMD, а также процессоров

P6 и NetBurst, описанных ранее. Однако в ней использованы и некоторые интересные аппаратные решения [8–11]. В частности, в состав процессора включены контроллер памяти DDR2 и блок предвыборки, что позволяет оптимизировать запросы к ОЗУ. Кроме того, в AMD64 реализуются однонаправленные магистрали HyperTransport типа «точка — точка» (до 24 Гбайт/с) и быстродействующий коммутатор, что ускоряет межпроцессорный обмен и упрощает построение эффективных вычислительных систем с NUMA-архитектурой (Non Uniform Memory Access) и ccNUMA-архитектурой (Cache Coherent Non Uniform Memory Access). Следует также отметить, что большинство современных процессоров содержат несколько однотипных процессорных ядер на одном кристалле.

Структура процессора K8 семейства 10h показана на рис. 18.

Интерфейсная часть процессорного ядра состоит из контроллера шин HyperTransport, контроллера памяти DDR2, блока хранения и оптимизации запросов к памяти. На одном кристалле процессоров семейства K8 расположены кэш-память команд и данных третьего уровня (L3), кэш-память команд и данных L2, 64 Кбайт двухассоциативной кэш-памяти L1D и кэш-память L1I, хранящая инструкции в частично декодированном формате (определены границы команд, поля команд). При загрузке кэш-память линейки в L1, размер которой составляет 64 байт, следующие 64 байт ОП загружаются автоматически (аппаратная предвыборка в предположении о последовательности адресов).

Особенностью процессоров AMD64 является механизм подготовки микрокоманд к декодированию и диспетчеризации. Перед помещением инструкций в кэш-память L1I они подвергаются частичному декодированию, состоящему в определении границ команд и размера кода операции, а также в выборе декодера, который будет использован при последующей обработке команд (эта информация также записывается в кэш-память L1I). Одновременно происходит предсказание направления ветвления для команд перехода.

В AMD64 реализованы два декодера: VectorPath и DirectPath, получающие на вход 32-байтные блоки команд. Декодер VectorPath способен декодировать длинные команды, состоящие из трех и более микрокоманд, а декодер DirectPath декодирует простые инструк-

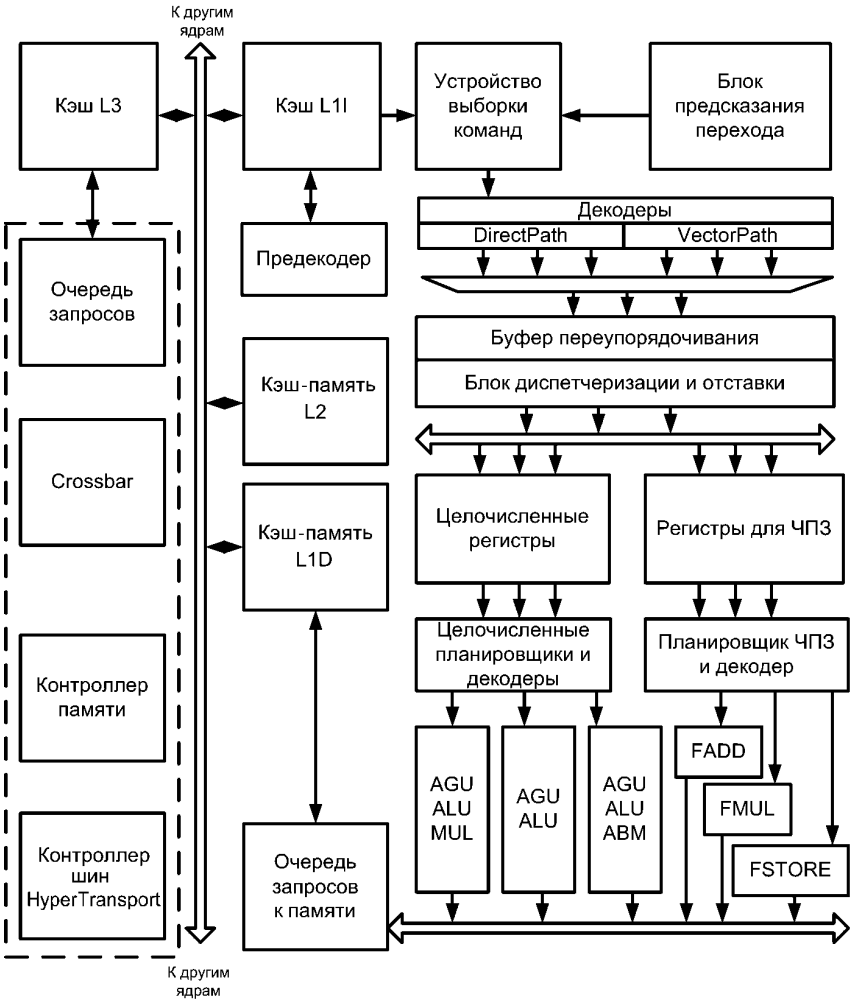


Рис. 18. Структура процессорного ядра K8 семейства 10h

ции, порождающие одну или две микрокоманды. Результатом работы декодеров являются последовательности микрокоманд, располагаемых в буфере переупорядочивания в виде групп по три микрокоманды (называемые в описаниях макрооперациями: *macro-ops*). Каждая группа представляет собой одно сложное задание, одновременно и параллельно обрабатываемое на трех исполнительных

устройствах. Темп декодирования каждого декодера — три группы за один такт, но, так как декодеры не могут выдавать результат одновременно, в буфер переупорядочивания в каждом такте передается не более трех микрокоманд. При формировании групп для связанных по данным микрокоманд возможно неполное их заполнение.

Буфер переупорядочивания хранит сформированные группы (до 24 групп по три микрокоманды) в упорядоченной последовательности и способен выдавать их в зависимости от типа микрокоманд в целочисленный планировщик или в планировщик ЧПЗ. Только после выдачи групп готовых к исполнению микрокоманд в очереди планировщиков происходит окончательное их декодирование и переименование регистров.

Для целочисленных микрокоманд позиция микрокоманды в группе однозначно определяет номер очереди и номер исполнительного устройства (микрокоманда в нулевой позиции распределяется в нулевое АЛУ, в устройство адресной арифметики и т. д.). В отличие от планирования целочисленных операций, планировщик операций над ЧПЗ не использует позиции группы при распределении микрокоманд на исполнительные устройства. Таким образом, исполнение групп микрокоманд может быть переупорядочено, в то время как микрокоманды одной группы одновременно исполняются и удаляются из буфера переупорядочивания (отставляются).

2.3. Микроархитектура суперскалярных процессоров IBM POWER4

За свою многолетнюю историю компания IBM неоднократно опережала конкурирующие фирмы при воплощении передовых идей. В частности, в IBM был разработан первый опытный суперскалярный процессор (до выпуска ЭВМ «Эльбрус 1» в 1978 г., однако результаты опубликованы не были), создан первый двухъядерный микропроцессор и т. п. Выпущенный в 2002 г., процессор POWER4 превзошел по производительности одного процессорного ядра ближайших конкурентов: Pentium 4, AMD Athlon XP и Sun

UltraSPARC III-Cu. Его основное применение — производительные вычислительные системы, серверы pSeries и iSeries.

В 64-разрядных процессорах IBM POWER4 используются сокращенный набор команд и большое число регистров общего назначения (архитектура RISC) [11, 12]. Архитектура процессора POWER4 в большой степени схожа с архитектурой процессоров семейств AMD64 и NetBurst (архитектура CISC): это конвейерный суперскалярный многоядерный микропроцессор.

В процессор POWER4 заложен высокий потенциал для создания SMP-систем. На одном кристалле объединяются два процессорных ядра (рис. 19). Каждое ядро имеет собственные раздельные блоки кэш-памяти L1I (64 Кбайт прямого размещения с контролем по четности) и L1D (неблокируемые двухассоциативные 32 Кбайт с контролем по четности). Для каждого ядра предусмотрено специальное устройство для хранения некэшируемых запросов (*Noncacheable Unit — NCU*), обеспечивающее заданную в программе последовательность запросов чтения и записи. На кристалле также располагаются три независимых блока кэш-памяти L2 (восьмиассоциативные с возможностью коррекции одиночных ошибок и обнаружения двойных ошибок), подключенных к коммутатору. На кристалле также расположена кэш-память тегов L3, контроллер кэш-памяти L3 и контроллер памяти. Кэш-память L3 реализуется на отдельном кристалле.

Четыре процессора POWER4 могут быть объединены в многочиповый модуль (*Multi-chip Module — MCM*), содержащий, таким образом, восемь процессорных ядер. Для целей коммутации используется контроллер фабрики (*Fabric Controller*), соединяющий каждый процессор на многочиповом модуле с тремя другими с помощью трех пар однонаправленных локальных шин «точка — точка». Предусмотрена возможность коммутации четырех многочиповых модулей для создания 32-процессорных SMP-систем с помощью специальных шин связи MCM. Для соединения процессоров с системой ввода/вывода служит 32-разрядная шина GX.

В процессоре также реализованы сервисные блоки: блок мониторинга производительности, блок пошаговой трассировки, блок самоконтроля, управляющий процессор, блок управления началь-

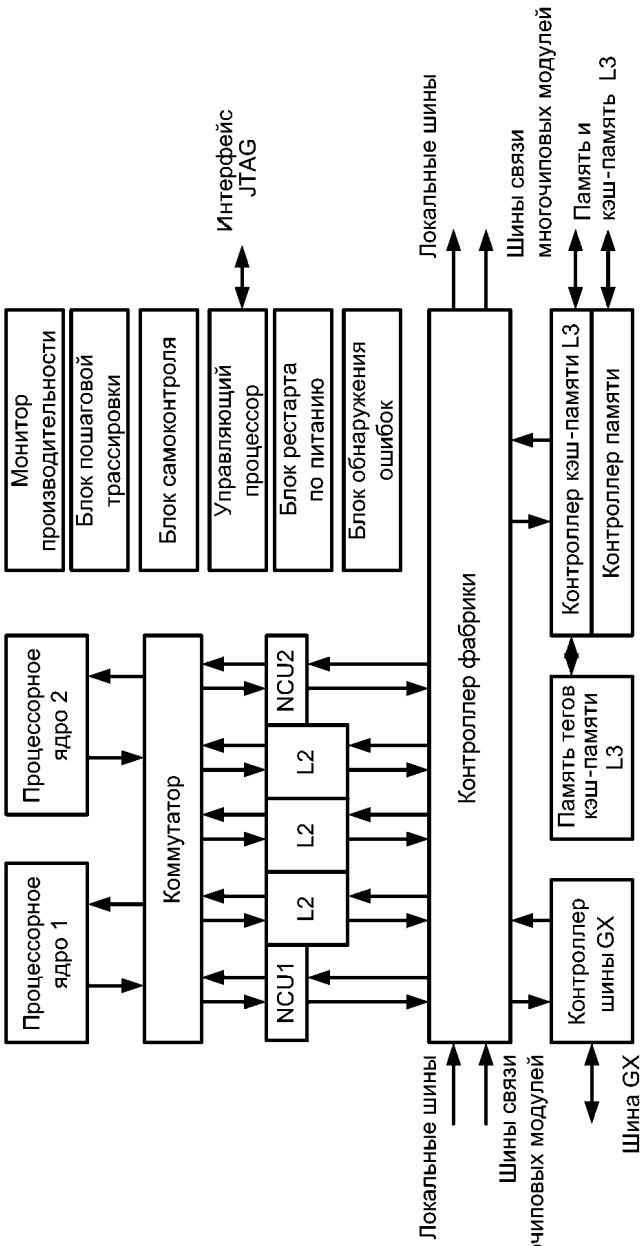


Рис. 19. Структура процессора IBM POWER4

ной загрузкой и рестартом по питанию, блок обнаружения ошибок.

Рассмотрим более подробно структуру суперскалярных процессорных ядер POWER4 (рис. 20).

Каждое процессорное ядро содержит устройство выборки команд, связанное таблицами быстрого преобразования, устройством предсказания направления ветвления и кэш-памятью команд первого уровня. Выборка осуществляется по адресу, находящемуся в *регистре адреса команды*. Если адресат не найден в кэш-памяти L1, происходит обращение в подсистему памяти по физическому адресу, получаемому из логического адреса с помощью таблиц быстрого преобразования. В процессоре реализованы три типа таблиц:

- таблицы быстрого страничного преобразования для команд и данных (*Translation Lookaside Buffer — TLB*);
- таблицы быстрого сегментного преобразования для команд и данных (*Segment Lookaside Buffer — SLB*);
- таблицы быстрого сегментно-страничного преобразования логических адресов в физические (*Effective-to-Real Address Translation*), содержащие результаты предшествующих преобразований по SLB и TLB.

В блоке предсказания направления ветвления реализованы три алгоритма предсказания. Предсказание по таблице истории переходов (*Branch History Table — BHT*), содержащей 16 384 записей, основано на хранении однобитовой истории происшедших переходов. Второй алгоритм основан на использовании вектора глобальной истории (*Global History Vector — GHV*), содержащего 11 бит истории для одиннадцати групп команд перехода. Этот вектор используется в качестве указателя на запись в таблице глобальной истории (*Global History Table — GHT*), содержащей 16 384 однобитных записей об истории происшедших переходов. Третий алгоритм использует 16 384 однобитных записей о том, какой из первых двух алгоритмов предсказания оказался более успешным для конкретной команды перехода. Указанные таблицы модифицируются после отставки команд перехода. В процессоре POWER4 предусмотрены команды, определяющие результат предсказания и изменяющие содержимое указанных таблиц.

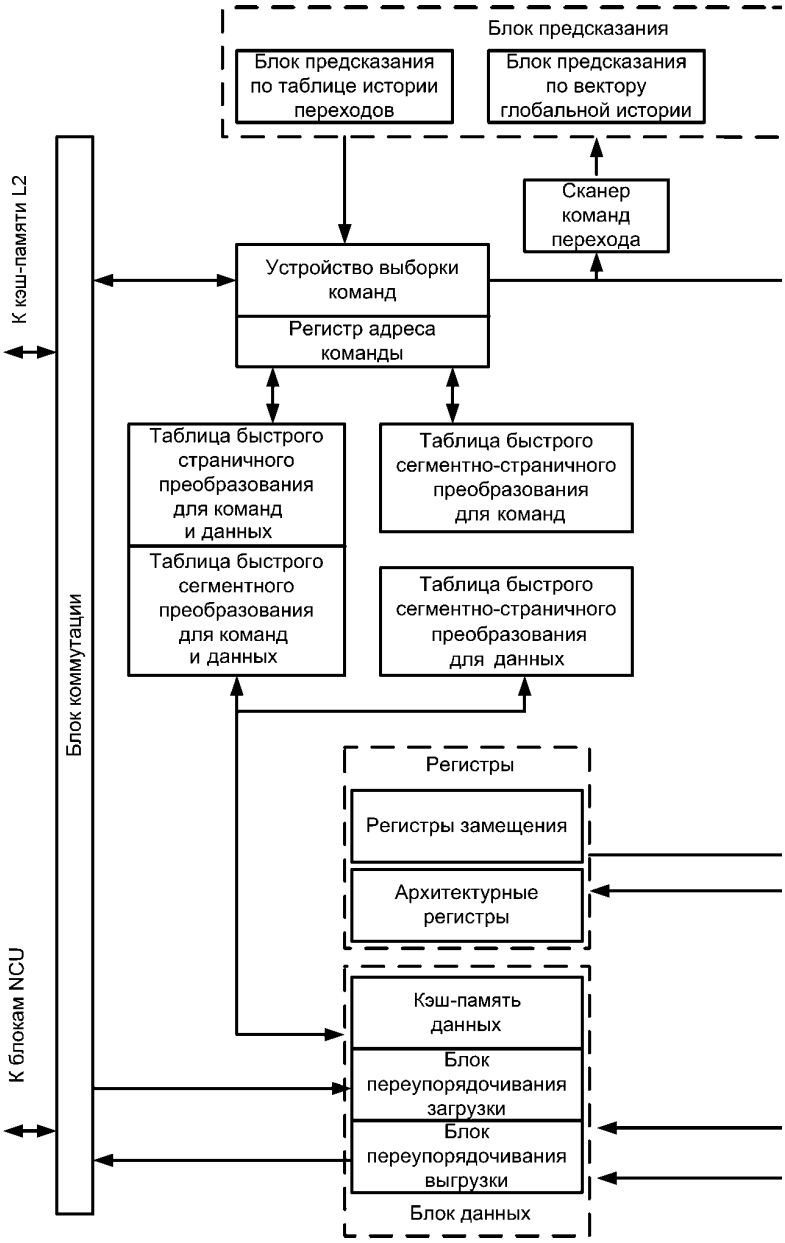
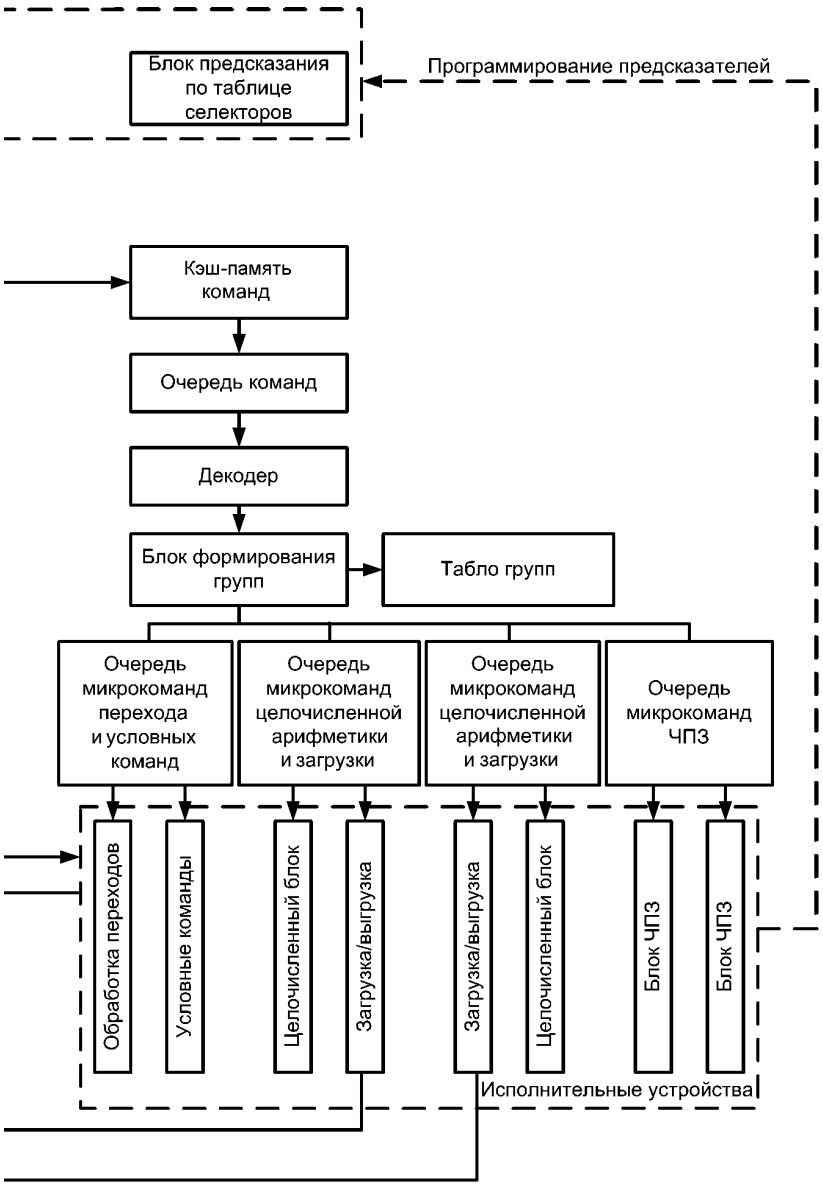


Рис. 20. Структура процессорного



ядра процессоров IBM POWER4

Выбранные из кэш-памяти L1 команды поступают в очередь команд и далее в декодер, формирующий упорядоченные группы микрокоманд. Каждая группа может содержать до пяти микрокоманд. Расположение микрокоманд в группе также соответствует исходной последовательности команд программы. Для обеспечения неупорядоченного исполнения часть архитектурных регистров переименовывается. Однако, если микрокоманда использует архитектурный регистр, не подвергающийся переименованию, генерируемая микрокоманда завершает группу, т. е. исполняется упорядоченно (раньше последующих). Одна позиция в группе предназначена исключительно для команд перехода. Информация о каждой группе хранится в специальном устройстве, называемом *табло групп* (*Group Completion Table — GCT*), оно обеспечивает своевременную отставку групп.

Применяются также дополнительные ограничения на размещение микрокоманд в группах и диспетчеризацию:

- для команды, состоящей из двух микрокоманд (загрузки и исполнения), обе микрокоманды должны быть помещены в одну группу;
- микрокоманды для более сложных команд (более двух микрокоманд) всегда размещаются в начале группы;
- все микрокоманды группы диспетчеризируются и отставляются одновременно;
- диспетчеризация группы возможна при доступности следующих ресурсов: свободная запись в табло групп (20 записей); свободный слот в группе (пять слотов); свободная запись в блоке переупорядочивания запросов чтения и в блоке переупорядочивания запросов записи; свободный регистр замещения (36 РОН, 32 регистра для ЧПЗ, девять регистров условных команд, два регистра-счетчика, один регистр управления операциями ЧПЗ, четыре регистра для обработки чисел с фиксированной запятой).

После диспетчеризации микрокоманды группы направляются в очереди исполнительных устройств для ожидания получения операндов из регистров. Каждое процессорное ядро POWER4 обеспечивает выдачу до пяти микрокоманд за один такт. К моменту диспетчеризации не все микрокоманды группы могут оказаться

готовыми, в результате чего только часть микрокоманд будет исполнена. Группа в этом случае останется неисполненной и будет находиться в очереди микрокоманд до готовности операндов.

Следующие стадии конвейера — исполнение микрокоманд группы, сохранение результатов в регистрах, ожидание завершения исполнения всех предшествующих групп и ожидание завершения всех микрокоманд группы. В процессорном ядре POWER4 предусмотрены четыре очереди к исполнительным устройствам:

- очередь команд к устройству арифметики переходов и к устройству обработки условных команд;
- две очереди к целочисленным АЛУ и к блокам загрузки/выгрузки. Далее микрокоманды загрузки и выгрузки совместно с данными и адресами направляются в блок переупорядочивания запросов на загрузку и в блок переупорядочивания запросов на выгрузку, связанные с кэш-памятью L1;
- очередь к устройствам обработки ЧПЗ.

Отставка выполняется при условии завершения исполнения всех микрокоманд группы и отставки предыдущих групп.

2.4. Микроархитектура суперскалярных процессоров Sun UltraSPARC III

Суперскалярные 64-разрядные микропроцессоры Sun UltraSPARC III были разработаны для реализации производительных серверов и систем с массовым параллелизмом фирмы Sun Microsystems, Inc. (начало продаж 2002 г.). Эти микропроцессоры, как и процессоры POWER4, имеют RISC-архитектуру [13]. Реализуемый в них набор команд описан в стандарте SPARC Instruction Set Architecture Version 9 (ISA V9) и в отличие от набора команд x86 может быть использован сторонними производителями микропроцессоров.

В процессорах Sun UltraSPARC используется архитектурный регистровый файл из 128 регистров, доступ к которому ограничен: в каждый момент времени доступны только 24 последовательных регистра (так называемое регистровое окно). Положение регистрового окна можно изменять с помощью команд, что позволяет

защитить локальные данные процессов и ускоряет обмен данными благодаря передаче параметров через внутренние регистры процессора. В остальном процессоры SPARC-архитектуры соответствуют общим принципам построения суперскалярных процессоров, описанных в разд. 1.

Структура процессорного ядра Sun UltraSPARC III-Cu показана на рис. 21. *Блок системного интерфейса* реализует протокол шины и управляет процедурой поддержания когерентности кэш-памяти процессора. Команды и данные, поступающие с внешней шины, передаются в основную кэш-память команд L1 (четырехассоциативная, 32 Кбайт с контролем по четности) и основную кэш-память данных L1 (четырехассоциативная, 64 Кбайт с контролем по четности). Также используются буферы предвыборки команд (одна линейка, 32 байт) и предвыборки данных (четырехассоциативный, 2 Кбайт), хранящие аппаратно- или программно-предвыбранные некэшируемые команды и данные. Как и в процессорах AMD, в процессоре UltraSPARC III реализован контроллер памяти, выдающий сигналы управления ОП. Это позволяет оптимизировать транзакции чтения и записи. При выгрузке результатов в память используется *буфер выгрузки* и *кэш выгрузки* (четырехассоциативный, 2 Кбайт), позволяющие переупорядочить запросы на запись. Также на кристалле кэш-памяти L2 размещена память тегов кэш-памяти L2 и контроллер кэш-памяти L2. Буфер TLB команд состоит из двух буферов быстрого страничного преобразования: малого 16-входового и большого 128-входового. Буфер TLB данных состоит из трех буферов быстрого страничного преобразования: малого 16-входового буфера и двух 512-входовых буферов.

Блок определения адреса команды получает адрес из блока предсказания направления ветвления, содержащего информацию о 4 К обработанных командах перехода. При этом процессор выбирает не только команды по предсказанному адресу, но и команды в альтернативной ветви программы. Для размещения первого потока команд используется *основной буфер команд*, а второго — *дополнительный буфер команд*. Указанная особенность существенно сокращает время сброса конвейера при неправильно предсказанном переходе (с семи до трех тактов). В состав набора команд

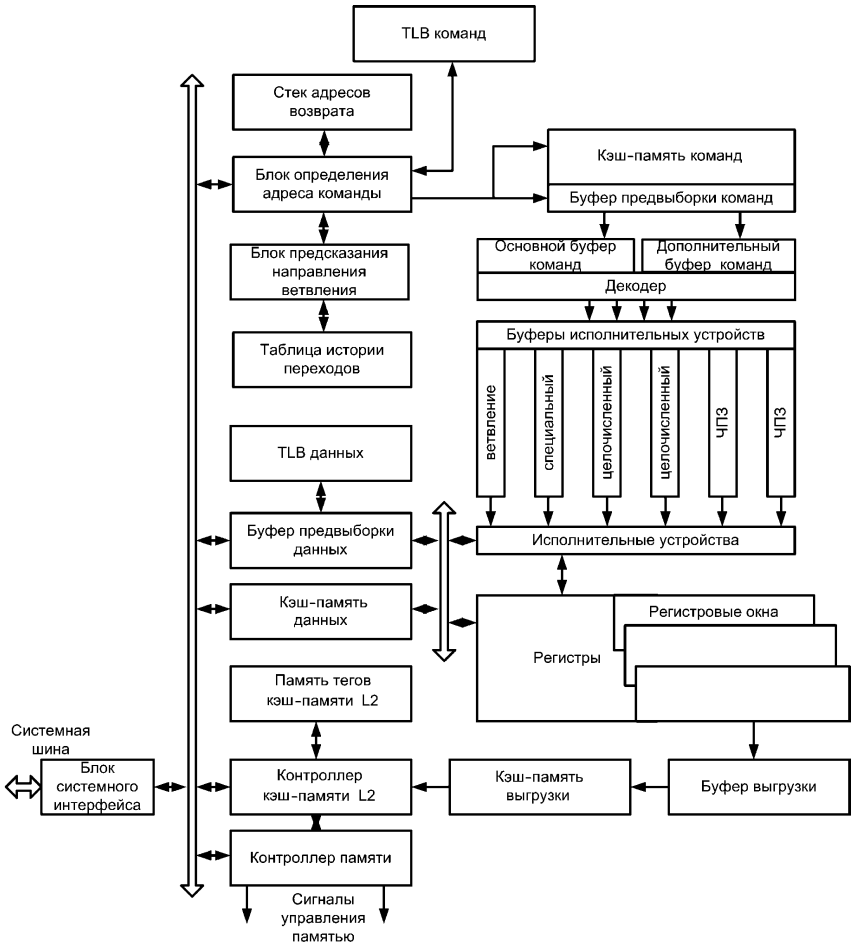


Рис. 21. Структура процессоров Sun UltraSPARC III

процессора включены инструкции поддержки Java-кода, влияющие на предсказание направления ветвления.

Инструкции из буферов команд направляются в декодер, откуда декодированные микрокоманды диспетчеризируются в буферы исполнительных устройств. В каждом такте в буферы и на исполнительные устройства может быть выдано не более четырех микрокоманд. Очереди к исполнительным устройствам разделены на шесть групп: очередь к устройству обработки ветвлений, очередь

специальных инструкций, две очереди к целочисленным АЛУ, две очереди к блокам обработки ЧПЗ.

2.5. Архитектура синтезируемых суперскалярных процессорных ядер MIPS32 74K

Ядро MIPS32 74K является первым коммерчески реализуемым синтезируемым описанием суперскалярного процессорного устройства. Его конечная реализация с частотами до 1 ГГц возможна с использованием технологий ASIC и FPGA. Это позволяет реализовывать на основе ядра MIPS32 74K высокопроизводительные системы цифровой обработки сигналов, размещаемые на одном кристалле. Корпорация MIPS также сопровождает процессорное ядро средствами проектирования, библиотеками функций, системами разработки и отладки программ.

Структура процессорного ядра MIPS32 74K показана на рис. 22. Особенностью данного суперскалярного ядра является возможность расширения функциональных возможностей благодаря применению пользовательских команд и подключению дополнительных исполнительных устройств (технология CorExtend) [14, 15].

Устройство шинного интерфейса обеспечивает пакетную передачу данных по 64-разрядной системной шине с различными частотами синхронизации. В состав ядра также входят раздельные кэш-память команд и кэш-память данных (неблокируемые четырехассоциативные, размер линейки 32 байт, с контролем по четности) с изменяемыми структурами и функциональностью. Размер кэш-памяти команд L1 может быть выбран из 16/32/64 Кбайт, кэш-памяти данных L1 — из 0/16/32/64 Кбайт. Реализованы механизмы блокировки линеек, предвыборки по части виртуального адреса. Кэш-память данных может быть реализована с политиками записи writeback и write-through.

Блок управления памятью (*Memory Manage Unit*) также может быть оптимизирован разработчиком: может быть использован двухассоциативный двухпортовый TLB команд и данных с 16/32/48/64 входами; четырехвходовой TLB команд для 4 Кбайт и 1 Мбайт стра-

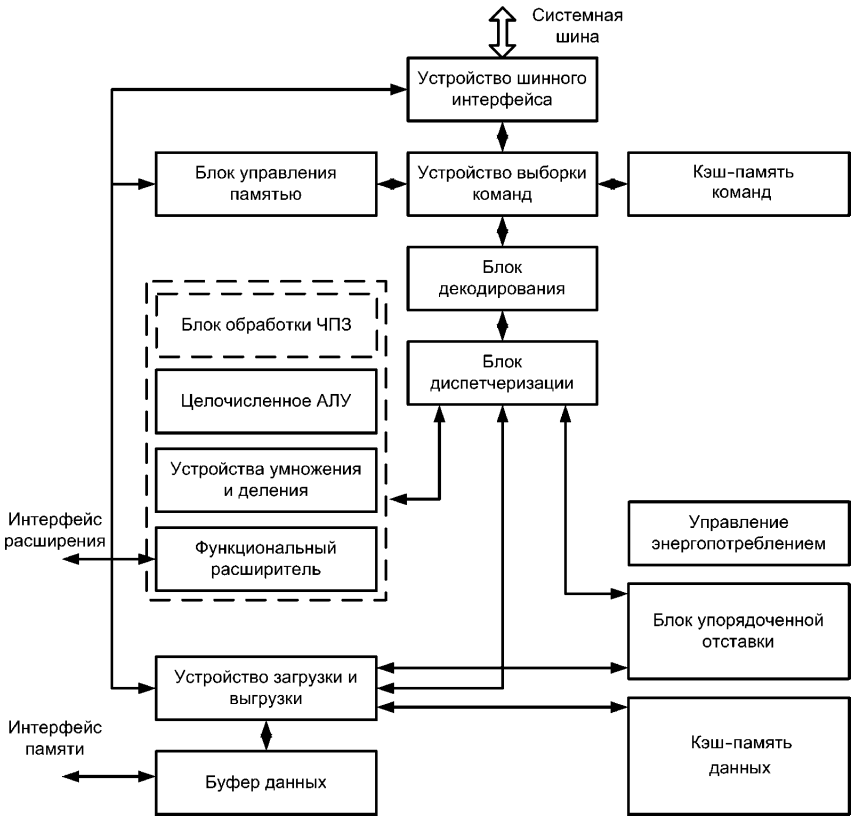


Рис. 22. Структура процессоров семейства MIPS32 74K

ниц памяти; буфер JTLB для 4, 16, 64, 256 Кбайт, 1, 4, 16, 64, 256 Мбайт страниц.

Возможности *устройства выборки команд*:

- выборка четырех команд за один такт;
- аппаратный стек адресов возврата с восемью ячейками;
- мажоритарный блок предсказания направления ветвления на основе трех таблиц истории по 256 записей каждая (структура подобна структуре блока предсказания в процессорах IBM POWER4);
- программирование размера аппаратной предвыборки при кэш-промахах (количество дополнительных линеек для предвыборки: 0/1/2).

Блок диспетчеризации в стандартном варианте состоит из двух очередей: очереди к целочисленному АЛУ и устройству умножения/деления, а также очереди к устройству адресной арифметики и команд перехода. Выполненные команды помещаются в блок упорядоченной отставки. Дополнительно может быть использован блок обработки ЧПЗ. Кроме того, АЛУ выполняет SIMD-команды с числами в форматах 2×16 и 4×8.

В составе процессора также реализованы дополнительные функциональные возможности:

- блок управления энергопотреблением, переводящий процессорное ядро в режим Power-down по команде WAIT;
- программно-управляемый делитель частоты синхронизации;
- поддержка отладки ядра по интерфейсу JTAG (EJTAG Debug 3.1);
- поддержка работы в режиме трассировки;
- поддержка самоконтроля.

Заключение

Суперскалярная обработка основывается на способности процессора выполнять более одной простой операции за один такт, что достигается благодаря дублированию исполнительных устройств. В настоящий момент выпускается несколько десятков коммерчески успешных суперскалярных микропроцессоров, нашедших применение в производительных ЭВМ с массовым параллелизмом, в персональных компьютерах и в серверах. Предпринимаются попытки реализации ядер суперскалярных процессоров на основе программируемых логических интегральных схем, что оказывается чрезвычайно востребованным при цифровой обработке сигналов. Несмотря на достаточно долгую историю, принципы организации суперскалярной обработки, равно как и сами процессоры, успешно применяются и продолжают совершенствоваться.

Литература

1. *Цилькер Б.Я.*, *Орлов С.А.* Организация ЭВМ и систем: Учеб. для вузов. СПб.: Питер, 2004. 668 с.

2. IA-32 Intel Architecture: Software Developer's Manual. Sept. 2005. — http://www.intel.com.

3. Intel 64 and IA-32 Architecture: Optimization Reference Manual. Order Number 248966-016. Nov. 2007. — http://www.intel.com.

4. The Microarchitecture of the Intel Pentium 4 Processor on 90nm Technology / Darrell Boggs, Aravindh Baktha, Jason Hawkins, et al. // Intel Technology Journal. 2004. V. 8, Issue 1. — http://developer.intel.com/technology/itj/index.htm.

5. The Microarchitecture of the Pentium 4 Processor / Glenn Hinton, Dave Sager, Mike Upton, Darrell Boggs, et al. // Intel Technology Journal. 2001. February 12. V. 5 (1).

6. P6 Family of Processors: Hardware Developer's Manual. Order No 244001-001. Sept. 1998. — http://download.intel.com/design/PentiumII/manuals/24400101.pdf.

7. *Шагурин И.И.*, *Бердышев Е.М*. Процессоры семейства Intel P6. Архитектура, программирование, интерфейс. М.: Горячая линия — Телеком, 2000. 248 с.

8. AMD64 Architectur: Programmer's Manual. V. 1–3. Advanced Micro Devices, Sept. 2007. — http://www.amd.com.

9. AMD Athlon Processor x86 Code: Optimization Guide. Publication No 22007. Feb. 2002. — http://www.amd.com.

10. Software Optimization Guide for AMD Family 10h Processors. Publication 40546. Dec. 2007. — http://www.amd.com.

11. *Бессонов О.* Обзор микроархитектур современных десктопных процессоров. Июль 2006. — http://www.ixbt.com.

12. POWER4 system microarchitecture / J. M. Tendler, J. S. Dodson, J. S. Fields, Jr. et al. // IBM J. Res. & Dev. Jan. 2002. V. 46, No 1. P. 5–25.

13. An Overview of UltraSPARC III Cu. UltraSPARC III Moves to Copper Technology // A White Paper. Sun Microsystems. Version 1.1. Sept. 2003 — http://www.sun.com.

14. Architectural Strengths of the MIPS32 74K Core Family / K. R. Kishore, Vidya Rajagopalan, Georgi Beloev and Radhika Thekkath // MIPS Technologies, Inc. Dec. 2007.

15. MIPS32 74Kc Processor Core Datasheet // MIPS Technologies, Inc. Dec. 14, 2007.

Оглавление

Учебное издание

Попов Алексей Юрьевич

Организация суперскалярных процессоров

Редактор *С. А. Серебрякова*
Корректор *Е. В. Авалова*
Компьютерная верстка *М. А. Голуба*

Подписано в печать 31.03.2011. Усл. печ. л. 3,49. Формат $60\times84/16$.
Тираж 100 экз. Изд. № 145. Заказ №

Издательство МГТУ им. Н. Э. Баумана.
Типография МГТУ им. Н. Э. Баумана.
105005, Москва, 2-я Бауманская ул., 5.

Для заметок