

Ю.И. Митропольский

**КОНЦЕПТУАЛЬНЫЙ ПРОЕКТ
МУЛЬТИАРХИТЕКТУРНОЙ
ВЫЧИСЛИТЕЛЬНОЙ
СУПЕРСИСТЕМЫ**

**ТЕХНОСФЕРА
Москва
2016**

ГЛАВА I.

ВВЕДЕНИЕ

В настоящей работе подведены итоги исследований, начатых в начале 90-х годов. Исследования явились продолжением работ по системе «Электроника СС БИС». На каждом этапе ставилась задача разработки оптимальной архитектуры вычислительной суперсистемы для текущего состояния технологической базы. Однако фундаментальные принципы построения системы актуальны и в настоящее время. Целый ряд предложенных концепций опередил зарубежные разработки, и это опережение доходило до 10 лет.

Данный проект является концептуальным, поскольку для перехода на этапы НИР и ОКР в силу известных причин не было никаких условий. Важной особенностью проекта является его свобода от ограничений совместимости, поэтому он сбалансирован и оптимален. Проект может быть основой для разработки отечественной элементной базы и архитектуры отечественных вычислительных суперсистем.

Исследования были поддержаны грантами РФФИ и проводились в рамках проектов ОНИТ РАН.

Автор выражает благодарность большому числу сотрудников, с которыми он имел счастье работать на протяжении более 50 лет.

Далее рассмотрена предыстория и основные этапы исследований по проекту.

I.I. Электронная вычислительная машина БЭСМ-6

Электронная вычислительная машина БЭСМ-6 была разработана в середине 60-х годов и сдана госкомиссии в 1967 г. Главный конструктор академик С.А. Лебедев заложил в основу структуры машины два основных принципа: принцип «водопровода», в настоящее время обычно называемый конвейером, и принцип специализации, обеспечивающий высокую эффективность при выполнении аппаратных или программных функций.

Машина БЭСМ-6 предназначалась для решения крупных научно-технических задач, что, естественно, отразилось как на ее архитектуре, так и на выборе системы элементов и конструкции [1].

Элементная база машины включала диодные логические схемы и усилители на переключателях тока с подвешенным источником питания. Важными особенностями системы элементов являлись высокая скорость переключения и очень высокая нагрузочная способность как по входу, так и по выходу. Диодные и усилительные схемы размещались в специальных блоках, которые, в свою очередь, устанавливались в стойку с двух сторон.

Система синхронизации обеспечивала возможность функционирования конвейера на тактовой частоте 10 МГц, что использовано в большинстве схем, в частности в арифметическом устройстве и в устройстве управления. На следующем уровне темп конвейера определялся циклом работы буферной памяти, который равен трем тактам. Этот цикл соответствует и максимальному темпу поступления команд. В арифметическом устройстве использовался асинхронный конвейер, темп получения результатов зависел от типа операции и от операндов. Средний темп составлял 10 тактов, что соответствует производительности 1 млн операций с плавающей запятой в секунду (1 Mflops).

Для согласования пропускных способностей процессора и оперативной памяти применялись расслоение памяти и неадресуемая буферная память, работающая в режиме кэш-памяти. Ее объем составлял 8 команд и 8 слов данных.

Важной особенностью машины были аппаратные и программные средства для обеспечения мультипрограммного режима. К ним относятся виртуальная адресация памяти со страничной организацией, система прерывания, наличие нескольких режимов выполнения команд в процессоре и соответствующие программы операционной системы.

При реализации подсистемы ввода-вывода ставилась задача обеспечения высокой пропускной способности при обмене с устройствами памяти на внешних магнитных носителях и обслуживания достаточного числа электромеханических устройств ввода и вывода. В машине было реализовано 7 быстрых направлений обмена (в современных терминах – 7 селекторных каналов) и набор медленных направлений, аппаратура для которых ограничивалась минимальным набором согласующих элементов и схем связи этих элементов с процессором. Функционирование медленных направлений (образующих мультиплексный канал) обеспечивалось программами работы с каждым конкретным типом устройства.

Указанные выше аппаратные средства обеспечили создание много-пользовательской операционной системы. За время эксплуатации машины было разработано несколько вариантов операционных систем, а также трансляторы с автокода и распространенных языков высокого уровня.

I.2. Система обработки данных АС-6

Установка и эксплуатация БЭСМ-6 в вычислительных центрах, где выполнялась обработка больших объемов данных, поступающих от большого числа абонентов, в частности в Центре управления полетами, послужила стимулом создания системы АС-6. В этих центрах узким местом являлось небольшое число внешних устройств и низкая пропускная способность подсистемы ввода-вывода БЭСМ-6. Разработка началась с создания аппаратуры сопряжения для БЭСМ-6 (отсюда и название – АС-6). На первом этапе ставились задачистыковки БЭСМ-6 с АС-6, которая должна была обеспечить подключение большого числа телеграфных и телефонных каналов, каналов приема телеметрической информации, а также увеличение объема памяти на магнитных дисках и существенное увеличение числа периферийных устройств. Однако по мере накопления опыта по использованию оборудования первого этапа стало очевидно, что в системе необходимы более мощные средства для обработки данных и, главное, наличие возможности наращивания системы за счет подключения дополнительных машин и устройств. Все эти обстоятельства привели к постановке задачи разработки многомашинной системы с развитыми средствами реконфигурации.

В основу реализации системы легли идеи специализации подсистем и устройств и унификации в рамках системы каналов обмена.

Кроме БЭСМ-6 в систему входили центральный процессор АС-6, периферийная машина ПМ-6, дополнительные устройства оперативной памяти, контроллеры магнитных дисков, контроллер приема телеметрической информации. Все эти устройства объединялись в систему в качестве абонентов канала 1-го уровня. Этот канал предназначался для передачи сообщений, содержащих 50-разрядное слово и 23-разрядный адрес.

Периферийная подсистема состояла из периферийных машин ПМ-6 и ряда контроллеров-мультиплексоров, подключенных к ПМ-6 с помощью канала 2-го уровня. Канал 2-го уровня предназначался для передачи сообщений, состоящих из произвольного числа однобайтовых посылок. Для осу-

ществления коммутации каналов использовались коммутаторы канала 2-го уровня. Общее число подканалов – 256. Основные контроллеры, рассчитанные на 32 подканала, – это мультиплексоры телефонных и телеграфных каналов и мультиплексор преобразования сопряжения, обеспечивавший подключение устройств, имеющих выход на интерфейс ввода-вывода ЕС ЭВМ.

Программное обеспечение системы АС-6 состояло из операционных систем БЭСМ-6, ЦП АС-6, ПМ-6, соответствующих систем программирования, тестовых и обслуживающих программ.

Система АС-6 с 1973 г. находилась в опытной эксплуатации, при этом продолжались работы по ее развитию. В 1975 г. она использовалась при проведении работ по программе совместного советско-американского проекта «Аполлон – Союз». Сдача системы в полном объеме была проведена в 1979 г.

В системе АС-6 были впервые реализованы новые идеи, явившиеся основой разработок суперЭВМ и фундаментальных исследований по архитектуре перспективных вычислительных систем. Прежде всего необходимо отметить следующие особенности:

- АС-6 – это неоднородная многомашинная вычислительная система;
- проблемная ориентация ЦП АС-6 на решение задач по управлению сложными объектами и эффективную трансляцию;
- функциональная специализация периферийной машины ПМ-6 и других вспомогательных устройств;
- специализация внутрисистемных каналов.

По мере создания и эксплуатации системы стало очевидным несоответствие новых архитектурных идей и возможностей элементной базы. В целях дальнейшего развития этого направления в 1973 г. был разработан проект системы БЭСМ-10, в котором на основе задела, полученного при создании АС-6, и использования высокоскоростных интегральных схем типа ЭСЛ планировалось создание перспективной вычислительной системы. Однако этот проект не был поддержан Министерством радиопромышленности СССР.

I.3. Вычислительная система «Электроника СС БИС»

Продолжение работ в этом направлении было осуществлено под руководством заместителя главного конструктора БЭСМ-6 и главного

конструктора АС-6 академика В.А. Мельникова. В соответствии с совместным решением Министерства электронной промышленности СССР и Академии наук СССР в 1978 г. была поставлена задача создания системы с предельной производительностью на основе проведения широкого фронта исследований по микроэлектронике, оптоэлектронике и другим направлениям. Система предназначалась для решения наиболее крупных задач в области ядерной физики, метеорологии, геологии, авиастроения, космонавтики, микроэлектроники и др.

Разработка суперкомпьютерной системы «Электроника СС БИС-1» базировалась на том научном багаже, который был накоплен при создании БЭСМ-6 и АС-6 [2, 3, 4]. Однако для достижения производительности, на два порядка величины большей, чем в этих машинах, было необходимо освоение нового технологического уровня и разработка соответствующей ему архитектуры.

В рамках создания системы были разработаны и освоены следующие компоненты и узлы [5]:

- высокочастотные интегральные схемы типа ЭЛС с задержкой на один логический уровень 0,5 нс и матричные БИС;
- БИС памяти типа ЭСЛ и КМОП;
- керамические без выводные корпуса и контактирующие устройства для них;
- многослойные высокочастотные печатные платы;
- микрокабель;
- высокочастотные соединители с контактными парами, имеющими согласованный импеданс;
- блоки и стойки вторичного электропитания;
- низкотемпературные тепловые трубы и тепловые разъемы;
- стойки фреонового и водяного охлаждения;
- установки функционального контроля.

В первоначальном проекте системы рассматривались возможности включения в ее состав следующих проблемно-ориентированных подсистем:

- основная машина с векторно-конвейерным процессором;
- матричная машина;
- машина для логической обработки данных.

Кроме того, рассматривались возможности включения следующих функционально-специализированных подсистем:

- периферийная машина;
- контроллер внешней полупроводниковой памяти;
- контроллер дисковой памяти;
- внешние машины;
- управляющие машины.

С учетом имевшихся ресурсов и первоочередных задач было принято решение отложить разработку матричной, логической и периферийной машин. В качестве внешних и управляющих машин использовались ЭВМ типа «Электроника 79», «Электроника 82» [6].

В центральном процессоре и контроллерах внешней полупроводниковой и дисковой памяти использовались схемы типа ЭСЛ среднего уровня интеграции и матричные БИС. Корпуса ИС устанавливались в контактирующие устройства, которые также обеспечивали их прижим к теплоотводящей трубке с фреоном. В каждом из 50 блоков процессора могло быть установлено до 144 БИС. Общее число вентилей – около 1 миллиона. Тактовая частота 100 МГц. В оперативной памяти также были использованы схемы ЭСЛ. БИС памяти и ИС управления устанавливались на тепловых трубках, закрепленных в тепловых разъемах блоков, которые, в свою очередь, контактировали с тепловой плитой, охлаждаемой фреоном. Во внешней полупроводниковой памяти были использованы схемы типа КМОП.

При выборе архитектуры центрального процессора рассматривался вариант ЦП АС-6, развитый в проекте БЭСМ-10, а также изучались суперкомпьютеры Cray 1 фирмы Cray Research и Cyber 205 фирмы CDC. Достижение максимальной производительности было возможно только при использовании синхронных конвейерных схем, обеспечивающих получение по крайней мере одного результата в такт. Система команд ЦП АС-6 была ориентирована на асинхронные конвейерные схемы, содержала большой набор форматов команд и данных, что усложняло ее реализацию на синхронных конвейерных схемах. Наиболее перспективной была признана архитектура Cray-1, что подтвердилось в дальнейшем. Однако архитектура процессора основной машины системы «Электроника СС БИС-1» отличалась от Cray-1 рядом усовершенствований. Так, удалось повысить темп выполнения операции вычисления обратной величины до 1 такта вместо 3, обеспечить одновременное выполнение векторных и скалярных операций с плавающей запятой за счет добавления соответствующих функциональных устройств, увеличить объем буфера команд. Пиковая производитель-

нность процессора равна 250 Mflops. Система команд процессора трехадресная, ориентирована на использование адресуемых скалярных, векторных и адресных регистров.

Подсистема внешней полупроводниковой памяти отличалась наличием интеллектуального контроллера, предназначенного для реализации различных методов доступа к внешней памяти со стороны основной машины. По заданию – директиве от основной машины программа специализированного процессора параллельно с управлением пересылкой данных осуществляла формирование адресов этих данных во внешней памяти по заданному алгоритму. Возможно вычисление адресов по любому закону или выборка их из списка. В отличие от аналогичных устройств, в которых память работает по алгоритмам дискового накопителя (Cray X-MP Solid-state Disk), в данном случае пересылке в оперативную память подлежали только полезные данные, что приводило к экономии оперативной памяти и снижению загрузки канала обмена с основной машиной. Наличие двух портов в контроллере позволяло построить двухпроцессорную систему с производительностью 500 Mflops [7, 8].

Подсистема дисковой памяти в двухпроцессорной системе обеспечивала подключение до 8 контроллеров с общим объемом памяти 20 Гбайт. Контроллер обеспечивал обмен между оперативной памятью основной машины и управляющим устройством накопителя, контроль и исправление ошибок с помощью кода Рида – Соломона.

Программное обеспечение состояло из операционных систем основной и внешних машин, систем программирования на языках макроассемблера, «Фортран 77», «Паскаль», «Си». Компиляторы включали средства оптимизации программ с учетом особенностей векторной архитектуры (векторизация циклов и планирование потока команд).

В 1991 г. были проведены испытания системы «Электроника СС БИС-1», изготовлены и наложены 4 образца, началась их установка у заказчиков. В том же году был разработан проект системы «Электроника СС БИС-2», направленный на создание многопроцессорной системы с производительностью до 10 Gflops. Кроме многопроцессорных основных машин планировалось включить в систему мониторные машины для управления системой и подготовки задач, а также подсистему с массовым параллелизмом.

В программе фундаментальных исследований в области вычислительной техники, элементной базы и программного обеспечения отделения информатики, вычислительной техники и автоматизации РАН, разрабо-

танной в 1992 г., планировались исследования архитектуры неоднородных сверхвысокопроизводительных вычислительных систем, включающих суперЭВМ с традиционной векторно-конвейерной архитектурой, подсистему с массовым параллелизмом и специализированные процессоры, исследования методов распределения и распараллеливания вычислений на таких системах, а также комплексные исследования элементно-конструкторской базы суперЭВМ, в том числе систем питания и охлаждения, и исследования методов автоматизированного проектирования, наладки, диагностики, тестирования и эксплуатации суперЭВМ.

В рамках этой программы планировалось создание неоднородной системы «Электроника СС БИС-3». В системе «Электроника СС БИС-3» предусматривалось объединение до 64 процессоров с архитектурой, совместимой с системой «Электроника СС БИС-1», со специализированной подсистемой, имеющей архитектуру массового параллелизма. Общая производительность системы – более 100 GFlops.

Однако в 1993 г. было принято решение о прекращении работ. Естественно, что это привело к разрушению всех коллективов, организованных за более чем 10 лет работы. Были утеряны научные и технологические заделы по многим направлениям, в том числе работы по решению больших задач на векторно-конвейерной машине.

Опыт разработки, производства и наладки системы «Электроника СС БИС-1» показал необходимость широкой кооперации академических институтов и промышленности. Только благодаря этой кооперации удалось организовать разработку и производство всех комплектующих, что выполнялось одновременно с разработкой и подготовкой производства всех устройств системы и с разработкой программного обеспечения [9, 10].

I.4. Исследования по неоднородным вычислительным суперсистемам

После прекращения работ по системе «Электроника СС БИС» исследования по архитектуре и возможности реализации неоднородных вычислительных суперсистем были продолжены. В этих исследованиях в основу был положен комплексный подход с учетом как проблем, связанных с аппаратной реализацией системы, так и проблем, связанных с созданием программного обеспечения.

Концепции построения неоднородных вычислительных суперсистем и предложения по архитектуре и подходам к их реализации были опубликованы в 1995 г. [11]. Впервые была предложена архитектура неоднородной суперсистемы, основанной на тесном взаимодействии процессоров для скалярной, векторной и параллельной обработки, которая дает возможность обеспечения высокой эффективности при решении больших задач, содержащих фрагменты с различными формами параллелизма [12].

В результате исследований по применению перспективных ультрабольших интегральных схем была впервые разработана и опубликована в 1997 г. архитектура векторного модульного масштабируемого унипроцессора, обеспечивающего выполнение десятков и сотен операций в один такт [13, 14]. Впервые была предложена структура масштабируемого процессора, состоящего из модулей обработки и коммутации. Набор последовательно соединенных модулей обеспечивает предельную производительность при выполнении длинной цепочки векторных операций или небольших программ. Параллельное включение таких наборов позволяет пропорционально увеличить производительность. Отличительной особенностью процессора является использование принципа близкодействия и отсутствие логически сложного управления.

В 2003 г. опубликованы результаты следующего этапа исследований, результатом которых стал проект мультиархитектурной вычислительной суперсистемы, в которой для фрагментов задач с параллелизмом на уровне данных используются векторные мультипроцессоры, для фрагментов задач с параллелизмом на уровне программ – скалярные мультипроцессоры, а для выполнения функций управления вычислениями и распределением ресурсов – мониторно-моделирующая подсистема [15]. Система обладает свойствами масштабирования на нескольких уровнях.

В 2005 г. опубликована статья, в которой впервые сформулирована и обоснована новая парадигма для суперкомпьютеров – мультиархитектура вычислительных суперсистем [16].

На первом этапе исследований рассматривалась система, состоящая из тесно объединенных векторного процессора и мультипроцессорной системы на основе микропроцессоров. Затем с учетом развития технологии и элементной базы исследовались подходы создания масштабируемого мультиконвейерного векторного процессора [17]. В 2008 г. был разработан проект многоуровневой масштабируемой мультиархитектурной вычислительной си-

стемы [18]. Были проведены исследования и разработка сетевой структуры системы, в том числе сети управления, межузловой сети и сети памяти [19].

В настоящее время объектом исследований является мультиархитектурная суперсистема, при этом ставится цель обеспечения максимальной производительности как за счет рационального и экономного использования аппаратных средств, согласования архитектуры и форм параллелизма в программах, так и за счет оптимизации математического и программного обеспечения. Масштабируемость системы на всех уровнях позволяет оптимизировать аппаратные средства в соответствии с потребностями прикладных программ.

1.5. Зарубежные системы

Наиболее важными и распространенными в вычислительных центрах для научно-технических расчетов были системы фирмы CDC – CDC 6600 (1964 г., 3 млн операций в секунду) и CDC 7600 (1969 г., 10 млн оп./с). Отличительными особенностями этих систем являются простая 3-адресная система команд, наличие набора специализированных конвейерных функциональных устройств, за счет параллельной работы которых могла быть достигнута предельная производительность. Также следует отметить наличие периферийных процессоров, освобождающих центральный процессор от управления вводом-выводом [20].

Десятилетие с 1976 г. по 1985 г. прошло при полном превосходстве векторно-конвейерных суперсистем фирмы Cray. В системе Cray-1 (1976 г.) использовалась 3-адресная система команд, аналогичная системе команд машин CDC 6600 и CDC 7600, дополненная векторными командами с прямой адресацией векторных регистров [21]. В режиме зацепления каждый такт выполнялись 2–3 операции с плавающей запятой, производительность достигла 160 Mflops. Фирма выпустила несколько серий многопроцессорных векторных систем, в частности Cray X-MP, Cray Y-MP [22]. 8-процессорный вариант Cray-2 с производительностью 1,9 Gflops занимал первое место с 1985 г. по 1990 г. [23].

Векторно-конвейерные системы выпускали и японские фирмы NEC, Hitachi и Fujitsu. В 1990 г. система NEC SX-3/44R (23 Gflops) вышла по производительности на первое место.

Однако к середине 90-х годов рынок суперкомпьютеров, зависящий от государственных заказов, резко сократился в связи с окончанием

холодной войны. Сохранялся только спрос на самые большие системы стоимостью от 1 до 5 млн долларов. Возник рынок «малых» суперкомпьютеров – от 100 тыс. до 1 млн долларов. Лидером в этом секторе была фирма Silicon Graphics. Кроме того, резкий рост производительности микропроцессоров и использование их в мультипроцессорных и мульти-машинных высокопроизводительных системах привел к снижению цен. В результате банкротства фирма Cray Computer была поглощена фирмой Silicon Graphics. Широко было распространено мнение, что микропроцессоры-киллеры вытеснят векторные суперкомпьютеры.

Переоценка ситуации произошла в 2002 г., когда японская система Earth Simulator, построенная фирмой NEC на 5000 однокристальных векторных процессорах, превысила по производительности систему фирмы IBM в 5 раз и достигла производительности 40 Tflops. Степень обеспокоенности американских специалистов показывает тот факт, что эту систему назвали computenic, т.е. «компьютер + спутник», имея в виду воздействие запуска советского спутника в 1957 г. [24].

Была возрождена фирма Cray, которая разработала ряд векторных систем с использованием собственных кристаллов и мультичипных конструкций. Система Cray-X1 достигла производительности 52,4 Tflops [25]. Однако разработки на микропроцессорах занимают лидирующие позиции. Исключением являются векторные мульти машинные системы фирмы NEC [26].

Если говорить о самых производительных системах, то следует отметить, что в 2004 г. фирмой IBM была разработана скалярная система на специальных микропроцессорах – Blue Gene/L с производительностью 92 Tflops. Эта система лидировала до 2007 г., когда ее версия достигла производительности почти 600 Tflops [27].

В марте 2006 г. фирма Cray объявила о планах создания архитектуры, объединяющей несколько типов процессоров на единой платформе. В ноябре 2007 года фирма Cray объявила о выпуске семейства суперкомпьютеров Cray XT5 и XT5h (hybrid – неоднородный) [28]. Основная вычислительная мощность в Cray XT5h обеспечивается векторными узлами Cray X2. Обе системы при соответствующем масштабировании могут обеспечить реальную производительность более 1 Pflops. В дальнейшем сообщений о выпуске этих систем не было. Фирма продолжала разработку и производство систем на микропроцессорах.

В 2008 г. лидером стала система IBM Roadrunner, которая с производительностью 1,37 Pflops стала первой универсальной системой, пре-

одолевшей рубеж в 1 Pflops [29]. Она также была первой гибридной системой, в которой использовались 8-ядерные неоднородные кристаллы Cell и 2-ядерные микропроцессоры AMD Opteron. В состав кристалла Cell входил процессор с архитектурой Power и 8 синергетических (векторных) сопроцессоров, объединенных кольцевой сетью.

В 2008 г. на первое место вышла скалярная система Cray Jaguar на многоядерных микропроцессорах Opteron – 2,3 Pflops [30]. В 2010 г. ее опередила китайская система Tianhe-1A – 2,57 Pflops, построенная на микропроцессорах Xeon x5670 фирмы Intel и графических процессорах фирмы NVIDIA [31].

В 2011 г. система K computer японской фирмы Fujitsu достигла пиковой производительности 10,51 Pflops. Система выполнена на 8-ядерных микропроцессорах Sparc64-VIIIfx, изготовленных фирмой Fujitsu [32].

В июне 2012 г. ее опередила система Sequoia из серии систем Blue Gene/Q фирмы IBM с производительностью 16,3 Pflops, выполненная на 16-ядерных микропроцессорах PowerPC A2 [33]. В системе использовалось рекордное число ядер – 1 572 864, имеется 5D-сеть.

В ноябре 2012 г. на первое место вышла система Titan – Cray XK7 фирмы Cray с производительностью 17,6 PFlops [34]. Система построена на 16-ядерных микропроцессорах Opteron 6274 и графических процессорах NVIDIA K20x.

В июне 2013 г. китайский суперкомпьютер Tianhe-2 с производительностью 33,86 Pflops на пакете программ LINPACK вышел на первое место [35]. Каждый из 16 тысяч вычислительных узлов состоит из двух процессоров Intel Ivy Bridge Xeon и трех сопроцессоров Xeon Phi. В системе 3,1 миллиона ядер. Кристалл Xeon Phi фирмы Intel включает 61 векторный процессор, каждый из которых может обрабатывать в векторном режиме операнды шириной 512 бит, что эквивалентно 16 операциям с плавающей запятой с 32-разрядными operandами или 8 операциям с 64-разрядными operandами [36].

I.6. Содержание концептуального проекта

Во второй главе проанализированы основные принципы создания вычислительных систем. Наиболее важным аспектом при этом является комплексный подход, при котором разработка архитектуры, аппаратных

средств и программного обеспечения, включая системные и прикладные программы, должна выполняться параллельно.

В третьей главе рассмотрена архитектура системы. В ее основе лежат концепции мультиархитектуры, взаимной адаптации архитектуры и программ, проблемной ориентации основных вычислительных средств, функциональной специализации вспомогательных вычислительных средств, специализации внутрисистемных сетей и иерархического построения системы.

В четвертой главе описаны архитектура и система команд масштабируемой основной машины. Набор модулей для построения процессоров позволяет компоновать структуры, ориентированные на различные формы параллелизма. Количественный состав модулей позволяет получать различные степени параллелизма в рамках одного процессора. Приведены архитектура и система команд отдельных модулей и процессоров на основе их объединения. Следует отметить, что описание системы команд носит эскизный характер, кодировка операций приведена для иллюстрации.

В пятой главе описаны архитектура и система команд функционально-специализированных машин, предназначенных для обеспечения функционирования мониторно-моделирующей подсистемы, сети памяти, сети управления, межузловой сети и периферийной подсистемы. Функционально-специализированные процессоры, входящие в состав системы, существенно отличаются от масштабируемых процессоров прежде всего из-за сокращения числа форматов данных за счет исключения операций с плавающей запятой, а также благодаря более простым схемам выполнения операций и меньшей глубине конвейерных схем. Основой для построения функционально-специализированных машин является базовый процессор. Индивидуальные функции, необходимые для той или иной машины, реализуются в виде модулей расширения.

В шестой главе проведено сравнение проекта с зарубежными разработками. Показано, что на всех этапах проведения исследований имел место концептуальный приоритет, достигавший в ряде случаев 10 лет.

В заключении предложены этапы для реализации проекта и рассмотрены перспективы развития.

ГЛАВА 2.

ОСНОВНЫЕ ПРИНЦИПЫ

2.1. Комплексный подход

Основой методологии разработки, изготовления и использования системы является комплексный подход, основанный на параллельной и взаимосвязанной разработке элементной базы, конструкции, методов проектирования, схемной документации, архитектуры и программного обеспечения, а также создания инструментов для обеспечения взаимной адаптации аппаратных и программных средств на всех этапах разработки, комплектации, настройки и управления системой в целом и ее отдельных компонентов. При этом ставится задача повышения эффективности работы как аппаратных, так и программных ресурсов. Повышение производительности системы зависит от достижения высокой тактовой частоты процессоров, от числа параллельно работающих логических схем и степени их загруженности полезной обработкой данных. Достижение максимальной эффективности, высокой степени загруженности аппаратных средств и снижения накладных расходов возможно только при учете взаимозависимости всего комплекса аппаратных средств, архитектуры системы, системного и прикладного программного обеспечения. Создание высокоэффективной системы для ограниченного класса задач и конкретного уровня технологии без жестких экономических ограничений вполне возможно. Однако при создании систем, ориентированных на широкий класс задач и достаточно длительное время их использования, а также на развитие системы и ее масштабирование по мере появления новых технологических возможностей, возникает целый ряд проблем, требующих новых подходов ко всем аспектам создания и использования вычислительных систем.

2.2. Взаимосвязь технологии и архитектуры

Целью повышения эффективности за счет согласования архитектуры и схемотехники является сокращение паразитных потерь на вспо-

могательных и организационных операциях и увеличение благодаря этому удельного веса собственно обработки. Кроме того, нельзя переоценить важность согласования работы памяти и процессора (процессоров), а также согласования работы центральной и периферийной частей системы.

Основным вопросом для суперсистем является использование возможностей увеличения объема оборудования в целях повышения параллелизма на аппаратном уровне. При этом необходим выбор формы параллелизма и уровня в системе, где будет использована данная форма. Имеется две основные формы параллелизма на аппаратном уровне, это использование множества одинаковых или разнородных параллельно функционирующих устройств, объединенных в параллельную структуру, и использование различных функционально-специализированных устройств, объединенных в конвейерную структуру. Обе формы параллелизма использовались при создании однопроцессорных скалярных машин, например для расслоения памяти и совмещения операций в процессоре. Прежде всего это параллелизм на уровне выполнения команд, получивший широкое распространение и развитие в общедоступных микропроцессорах. Кроме этого, необходимо упомянуть параллелизм программ-нитей, или multithreading, часто называемый в русской литературе мультитредовой структурой, однако, с нашей точки зрения, термин «микромультипрограммирование» лучше соответствует сути, так как отражает мультипрограммирование на уровне микроструктуры процессора. Рассмотрим особенности использования этих форм параллелизма в суперсистемах.

В параллельных структурах объектами объединения являются функциональные устройства, процессоры или машины. С точки зрения повышения эффективности желательно наличие высокой степени локализации обработки данных в каждом из указанных объектов и сокращение обмена данными между объектами. Последнее особенно важно при увеличении числа параллельных объектов, поскольку при этом усложняется система коммутации и увеличиваются задержки при передаче данных между объектами.

В конвейерных структурах степень параллелизма обработки данных зависит от числа станций конвейера, соответствующих числу подфункций, на которые может быть разбита функция. Эффективность работы таких структур зависит от наличия больших объемов данных для обработки и средств их доставки на вход конвейера. Следует подчеркнуть, что при

увеличении числа станций, т.е. длины конвейера, и сохранении их загруженности эффективность увеличивается, поскольку скорость передачи данных не зависит от числа станций, а зависит только от быстродействия логических схем и расстояния между соседними станциями.

Большое влияние на эффективность параллельных структур имеет топология размещения отдельных схем, функциональных устройств, модулей и процессоров.

Оценка энергозатрат на различные функции позволяет оценить те или иные архитектурные и аппаратные решения. В работе [37] приводятся следующие данные по расходу энергии:

- обработка 64-разрядных операндов – 20 пДж;
- умножение с фиксированной запятой – 0,5 пДж;
- умножение с плавающей запятой – 25 пДж;
- пересылка данных в кристалле – 20–250 пДж;
- пересылка данных вне кристалла – 300–500 пДж;
- пересылка данных в системе – 1 нДж;
- выполнение команд вне очереди - планирование выдачи команд – 2 нДж;
- обращение к динамической памяти – 16 нДж.

Эти данные еще раз подтверждают необходимость физической концентрации вычислительных ресурсов, упрощения схем управления выдачей и выполнением команд, сокращения пересылки данных и обращения к памяти.

2.3. Формы параллелизма в программах и в архитектуре

Хотя большинство алгоритмов и программ ориентированы на последовательные вычисления, часто можно выделить некоторые их свойства, создающие условия для параллельной обработки. Прежде всего это касается обработки больших массивов данных, это так называемый параллелизм на уровне данных. С наибольшей эффективностью эта форма параллелизма используется в векторных машинах. Другая форма параллелизма – параллелизм на уровне задач, имеет место в программах, разбиваемых на большое число независимых или слабо связанных подзадач.

Для решения задач на параллельных системах алгоритмы и программы подвергаются адаптации к преобладающей в данной системе форме параллелизма на уровне аппаратуры и архитектуры. Например, задачи с параллелизмом на уровне данных при решении на системе с чисто параллельной структурой разбиваются на отдельные части для обработки фрагментов большого массива, при этом необходима специальная программа длястыковки и объединения результатов работы отдельных частей программы. Для эффективного решения на векторной машине программ с параллелизмом на уровне задач требуется объединение этих задач в целях лучшей векторизации программы. Большие трудности возникают при переносе задач с одной системы на другую, особенно если в этих системах преобладают разные формы параллелизма.

Очевидно, что в суперсистемах должны быть использованы обе формы параллелизма, и только от их оптимального взаимодополнения зависит конечная эффективность. На уровне функциональных устройств естественно использовать параллельные структуры для выполнения параллельной обработки разрядов слова. Наилучшим решением для функциональных устройств является применение синхронного конвейера. При разработке архитектуры процессоров для суперсистем необходимо обеспечить возможности как построения достаточно длинных цепочек функциональных устройств, так и организации параллельной и независимой работы большого числа устройств. На уровне объединения процессоров или машин основными проблемами являются построение распределенной иерархически организованной памяти, создание эффективной системы коммутации и аппаратно-программной структуры для оптимизации обмена данными между уровнями памяти.

В такой системе возникают новые проблемы, связанные с выявлением формы параллелизма того или иного участка программы и дальнейшей его оптимизацией. Следующей проблемой является распределение ресурсов, планирование выполнения программы и собственно управление ее выполнением.

В перспективе развитие концепции мультиархитектуры должно привести к проектированию и изготовлению систем под конкретные задачи, что станет экономически оправданным при определенной степени автоматизации соответствующих процессов. В данном случае необходимо упомянуть системы с перестраиваемой структурой. Эти системы при всей их привлекательности для многих приложений обладают суще-

ственным недостатком, особенно важным применительно к суперсистемам. Это высокий уровень накладных расходов на коммутацию при перестройке и связанное с этим снижение производительности.

2.4. Принципы параллелизма и локальности данных

Исследование форм параллелизма, обеспечивающих максимальную производительность, и исследование методов применения этих форм являются основой создания вычислительных систем предельной производительности. Достижение экзафлопсной производительности возможно только при использовании и согласовании всех форм параллелизма как на уровне программ, так и на уровне архитектуры и аппаратных средств. В первую очередь необходимо исследование форм параллелизма, обеспечивающих максимальную производительность на аппаратном уровне, а затем исследование методов применения этих форм на уровне архитектуры, системных и прикладных программ. Такой подход, связанный с приоритетом физической реализации, позволяет с максимальной эффективностью использовать все возможности аппаратуры, в том числе при внедрении новых технологий и приборов. Однако при этом потребуются дополнительные усилия, связанные с отказом от совместимости с системами предыдущего поколения и с созданием новых вычислительных методов и программ.

Принцип взаимной адаптации предполагает наличие двух взаимосвязанных процессов. Первый процесс включает разработку архитектуры и аппаратных средств, наилучшим образом соответствующих формам параллелизма, применяемым в программах, а также создание новых архитектурных и аппаратных подходов. Второй процесс включает создание оптимизированных программ для систем с новой архитектурой, а также исследование новых вычислительных методов и алгоритмов. На следующем этапе возможен новый цикл взаимовлияния.

Обеспечение локальности данных является еще одним эффективным инструментом повышения производительности. В больших системах обычных средств обеспечения локальности данных недостаточно. В исследуемой системе используется иерархическая система управления данными, в которой важным инструментом является использование об-

менно-редактирующих машин для обмена между уровнями иерархии памяти. В результате возможна оптимизация перемещения данных и сокращение во времени обмена и вычислений.

2.5. Развитие технологии и совместимость

К числу противоречивых тенденций, возникающих при создании новых систем, относится совместимость программного обеспечения при переходе на новый технологический уровень и при изменении или расширении круга задач. Первым примером разрешения проблемы совместимости было создание системы IBM/360 [38]. На первом этапе было предложено 6 совместимых моделей, использующих единую операционную систему и имеющих диапазон производительности в 50 раз. По мере развития системы с 1964 года до настоящего времени было создано 7 основных вариантов архитектуры и очень большое число моделей. Основные изменения архитектуры были связаны с увеличением объема адресуемой памяти, совершенствованием виртуальной памяти – увеличение объема и введение нескольких адресных пространств, с введением мультипроцессирования, векторной обработки, расширения парка периферийного оборудования. Наконец, в последней версии системы – IBM z/ARCHITECTURE – введены 64-разрядные регистры данных, а также 64-разрядные команды, виртуальные и физические адреса, что соответствует объему памяти $1,8 \times 10^{19}$ (16 EXAB) [39]. В результате фирме удалось сохранить использование системы в узком секторе корпоративных систем, а в сфере персональных компьютеров, серверов и суперкомпьютеров были использованы другие архитектуры.

Еще одним примером длительного использования архитектуры является x86 фирмы Intel. В течение 30 лет эта архитектура прошла путь от простого 4-разрядного процессора до 60-ядерного векторного процессора. Первоначально повышение производительности достигалось в основном за счет повышения тактовой частоты и увеличения разрядности. На следующем этапе возможности повышения тактовой частоты резко сократились, однако за счет повышения степени интеграции стало возможно использовать целый набор механизмов повышения параллелизма на

уровне выполнения команд. При этом удельный вес аппаратных средств собственно процессора заметно сократился.

Следующим шагом использования большого объема оборудования на кристалле было объединение нескольких стандартных процессоров (так называемые multi-core), в настоящее время обычно число процессоров не превышает 16. Конструкции из большого числа сравнительно простых процессоров получили название many-core. К таким конструкциям относится и последняя разработка фирмы – специализированный кристалл, включающий 60 векторных процессоров, каждый из которых может обрабатывать 512 параллельных разрядов, или 8 64-разрядных слов, или 16 32-разрядных слов [36].

Развитие суперкомпьютеров с середины 70-х до начала 90-х годов началось с машины Cray 1, основанной на идее достижения максимальной производительности одного векторно-конвейерного процессора [21]. Затем было создано несколько моделей, в том числе и мультипроцессорных [22].

Вслед за разработками фирмы Cray ряд фирм выпустили собственные варианты векторных суперкомпьютеров. Это сопровождалось широким распространением новых архитектурных и программных идей. Кроме того, расширился фронт исследований по технологическим проблемам, в частности связанным с вопросами построения систем питания и охлаждения. К числу разработанных векторно-конвейерных машин относится и уже упоминавшаяся отечественная суперкомпьютерная система «Электроника СС БИС-1». В рамках выпуска системы было разработано большое число новых технологий – сверхбыстродействующие матричные СБИС, СБИС памяти большой емкости, высокочастотные печатные платы, разъемы и микрокабель, фреоновая система охлаждения, тепловые трубки, новые источники питания и др. [9].

Система IBM/360 не оказала заметного влияния на развитие суперкомпьютеров, что нельзя сказать о микропроцессорах. Дешевые и достаточно производительные микропроцессоры, которые были названы «киллерами», вытеснили все другие подходы к созданию систем и фактически направили развитие в ложном направлении, в результате чего все усилия были сосредоточены на разработке микропроцессоров для персональных компьютеров, ориентированных на повышение скорости решения однопотоковой задачи за счет параллелизма выполнения команд и автоматического обеспечения локальности данных.

2.6. Новые подходы к разработке архитектуры, аппаратуры и программного обеспечения

Новые архитектуры для решения больших задач нужны были давно, но их развитие было фактически приостановлено в целях обеспечения совместимости с микропроцессорами и их программным обеспечением. Только в настоящее время в связи с исследованиями по экзафлопсным системам стало преобладать мнение, что необходимы новые подходы к архитектуре и отказ от жесткой совместимости. В течение последнего времени большую популярность приобрели т.н. акселераторы – специализированные процессоры в основном для задач с параллелизмом данных, используемые как дополнение к стандартным микропроцессорам, на которых выполняются скалярные вычисления и работает операционная система. В качестве акселераторов широко используются модифицированные графические процессоры фирмы NVIDIA со специальной программной системой CUDA. Фирма Intel разработала свой акселератор – многоядерный векторный процессор, основанный на совместимости с архитектурой x86. Эти подходы в определенной степени обеспечили повышение производительности, их можно рассматривать как промежуточный этап.

Один из радикальных подходов был предложен на основе наших исследований по мультиархитектурным вычислительным суперсистемам, которые явились продолжением работ по системе «Электроника СС БИС-1» и по проекту системы «Электроника СС БИС-2» [11–19].

Основные концепции этого подхода, впервые опубликованные в 1995 г. и развитые в последующие годы, следующие:

- унипроцессор с максимальной производительностью, имеющий модификации с различными формами параллелизма, и мультимодульная масштабируемая машина;
- мультиархитектурная структура на всех уровнях вычислительной подсистемы и взаимная адаптация архитектуры и прикладных программ;
- управляемая локализация данных;
- иерархическая структура системы;
- мониторно-моделирующая подсистема;
- функционально-специализированные сети и подсистемы;

- функционально-специализированные процессоры;
- эффективное масштабирование.

Наши исследования показали, что разработку вычислительных систем надо начинать с архитектуры масштабируемых суперсистем. Модульность таких систем позволит выделить или специально разработать компоненты и для менее производительных систем, включая серверы, персональные системы и мобильные устройства. В системе используются самостоятельные процессоры, поэтому потребности использования стандартных микропроцессоров нет.

Важен аспект обеспечения возможности включения в систему подсистем с новой архитектурой, в том числе подсистем на новой элементной базе. Благодаря такому подходу обеспечение совместимости не будет тормозом для развития, как это происходит в настоящее время. Одним из факторов для обеспечения этой возможности сохранения совместимости при смене технологии является отделение мониторно-моделирующей подсистемы, в которой реализуются в том числе функции операционной системы, от других подсистем. При этом возможны локальные технологические решения, используемые сначала в подсистемах с новой архитектурой, переносимые на следующих этапах развития и на другие подсистемы. Благодаря такому подходу эволюционность развития не будет вступать в противоречие с необходимостью глобальных изменений при смене технологии.

2.7. Основные принципы построения системы

Мультиархитектура системы, т.е. наличие нескольких типов основных проблемно-ориентированных вычислительных машин и подсистем, обеспечивает ее адаптивность при решении самых различных прикладных задач. В зависимости от формы параллелизма задачи или части большой задачи выбирается вычислительная подсистема с соответствующей архитектурой. Основные машины имеют модульную структуру. На основе выбора набора модулей и масштабирования создается архитектура конкретной масштабируемой основной машины. Модульность конструкции масштабируемой машины обеспечивает достижение максимальной производительности унипроцессора за счет конвейерной

структуры как отдельных модулей, так и их объединений в виде цепочек и параллельной работы цепочек. Модульная конструкция позволяет иметь основные машины с различной архитектурой.

Функциональная специализация вспомогательных процессоров, машин и сетей повышает эффективность выполнения как основных вычислительных задач и связанных с ними перемещений данных, так и задач операционной системы, управления сетевой структурой и подсистемой ввода-вывода.

Наличие мониторно-моделирующей подсистемы, предназначеннной для выполнения обычных функций операционной системы – трансляция и загрузка задач, позволяет осуществлять распределение ресурсов и управление вычислительным процессом, а также новых функций моделирования и анализа отдельных частей задач на предмет формы параллелизма. В состав мониторно-моделирующей подсистемы входит иерархическая структура управляющих машин, связанных сетью управления между собой и с другими подсистемами и машинами. Такая структура обеспечивает работу распределенной операционной системы.

Специальная подсистема управления пересылкой данных по заданию прикладной программы, входящая в состав иерархической сети памяти, а также включение общей памяти и обменно-редактирующей машины на каждом уровне иерархии системы позволяют резко повысить эффективность за счет снятия с основных вычислительных машин нагрузки по пересылке и редактированию массивов данных.

Межузловая сеть, предназначенная для построения горизонтальной структуры обмена сообщениями между взаимодействующими программами, выполняемыми в масштабируемых основных машинах, объединенных в вычислительные узлы, обеспечивает высокую эффективность взаимодействия за счет применения сетевых машин и развитой системы коммутации.

Распространение принципа близкодействия на все уровни перемещения данных обеспечивает сокращение затрат энергии и времени.

Возможность создания заказной архитектуры для группы прикладных задач возникает за счет модульности конструкции и масштабирования при наличии системы автоматизации крупномодульного проектирования и производства.

Новые подходы к архитектуре системы потребуют не только новых подходов к созданию распределенной операционной системы и программных средств сетевой структуры, но и новых концепций создания

прикладных программ, включая новые вычислительные методы, методы управления данными и методы распараллеливания.

Повышение эффективности вычислений может быть достигнуто за счет объединения в единой системе вычислительных ресурсов, ориентированных на разные формы параллелизма. Были проведены исследования мультиархитектурной вычислительной системы, в которой вычисления распределены между векторными и скалярными масштабируемыми процессорами, которые могут объединяться в мультипроцессорные модули различной комплектации. Вычислительная система состоит из вычислительной подсистемы, включающей все вычислительные ресурсы, архивной памяти, периферийной подсистемы и мониторно-моделирующей подсистемы.

Дополнительными ресурсами повышения производительности являются оптимизация подготовки и распределения задач и оптимизация подготовки и распределения данных для выполнения этих задач.

В мультиархитектурной системе для обеспечения эффективности необходимо оптимизировать (с учетом архитектуры масштабируемых машин) решение задачи на всех этапах – от выбора численного метода, алгоритма, трансляции и распределения частей задачи. Другой аспект связан с обеспечением подкачки данных в оперативную память масштабируемого процессора. В настоящее время в большинстве случаев обеспечение локальности данных, т.е. своевременной подкачки, выполняется за счет многоуровневой кэш-памяти. Однако для многих задач это приводит к большим потерям. Необходимо иметь средства планирования перекачки данных в соответствии с алгоритмом задачи. Для реализации этих функций необходима аппаратно-программная подсистема, которая, с одной стороны, независимо от программиста выполняет функции операционной системы, а с другой – при тесном взаимодействии с программой обеспечивает указанные функции оптимизации.

Многоуровневая структура системы в отличие от одноуровневой при большом числе процессоров позволяет более эффективно локализовать фрагменты задачи на ограниченном числе ядер-процессоров. На разных уровнях системы – масштабируемый процессор, вычислительный узел или мультикомпьютер – могут присутствовать процессоры с различной архитектурой. Данные особенности системы необходимо учитывать при создании сети управления, структуры этой сети и распределенной аппаратно-программной структуры мониторно-моделирующей подсистемы.

Все уровни иерархии памяти и сети памяти также являются объектами управления со стороны мониторно-моделирующей подсистемы. При этом указанная выше оптимизация подготовки и распределения данных должна быть согласована со структурой памяти на всех уровнях, в частности с наличием обменно-редактирующих машин (интеллектуальных контроллеров обмена).

Другими объектами управления являются сеть межузлового обмена и сетевые процессоры, управляющие этой сетью. В данном случае речь не идет о прямом управлении со стороны мониторно-моделирующей подсистемы, поскольку межузловой обмен инициируется вычислительными программами. Однако для повышения эффективности этого обмена часто взаимодействующие между собой программы желательно размещать в соседних или близко расположенных машинах и вычислительных узлах. Размещение программ является функцией мониторно-моделирующей подсистемы, которая выполняется при взаимодействии с программистом.

Составной частью сети памяти являются средства ввода-вывода. С одной стороны, они тесно взаимодействуют через сеть памяти с устройствами памяти разных уровней, а с другой стороны, такие средства управления, как периферийные процессоры и устройства обмена (каналы ввода-вывода), сами являются частью мониторно-моделирующей подсистемы, поскольку они выполняют некоторые функции операционной системы.

Исследования и разработка сети управления и мониторно-моделирующей подсистемы должны выполняться комплексно с учетом следующих факторов:

- архитектуры, состава и конфигурации вычислительной подсистемы;
- особенностей программирования в мультиархитектурной среде;
- новых требований к прикладным программам;
- новых требований к операционной системе.

2.8. Требования к прикладным программам

Мультиархитектурная вычислительная система должна обеспечивать выполнение любой программы на предусмотренном в системе языке программирования. Однако наличие в системе масштабируемых

процессоров с различной архитектурой приводит к необходимости выделения отдельных частей программы, форма параллелизма в которой соответствует определенной архитектуре. Аналогично настройка может иметь место при построении вычислительных узлов, мультикомпьютеров, вычислительных комплексов. Задача автоматического выделения таких частей программы является достаточно сложной, а во многих случаях невыполнимой без участия автора прикладной программы. Кроме того, на всех этапах создания оптимальной прикладной программы необходимо учитывать особенности архитектуры системы. К числу этих этапов, прежде всего, следует отнести выбор численных методов, алгоритма вычислений, распределения памяти и собственно программирование. Ясно, что роль программиста и его влияние на оптимизацию программ на всех этих этапах их создания нельзя преуменьшать. Таким образом, необходима взаимная адаптация, т.е. настройка аппаратных средств в соответствии с особенностями программ и разработка вычислительных методов, алгоритмов и программ для эффективного распараллеливания на имеющейся вычислительной системе.

Наиболее высокая производительность может быть достигнута при максимальной загрузке конвейерных устройств в системе. Это может быть достигнуто за счет увеличения удельного веса векторизуемых программ, использования сложных векторных вычислений, обеспечивающих работу длинных цепочек векторных операций, и конвейеризации скалярных вычислений. Загрузка конвейерных устройств и процессора в целом зависит также от своевременной подкачки данных в соответствующую память. Участие программиста при этом должно состоять в планировании обмена на разных уровнях иерархии памяти и задании формы размещения данных. Одной из эффективных особенностей системы является возможность редактирования данных в процессе обмена. Наличие специализированных обменно-редактирующих процессоров в контроллерах обмена между соседними уровнями памяти с прямым доступом позволяет в процессе обмена управлять адресацией данных по заданному алгоритму, а не только по последовательным адресам. Как правило, задание на алгоритм обмена выдает прикладная задача.

Возможности масштабирования системы распространяются на все ее уровни – масштабируемые процессоры, вычислительные узлы, мультипроцессоры. Если программист учитывает эти возможности при на-

писании программы, то степень ее оптимизации может возрасти. В данном случае возникают противоречивые обстоятельства. С одной точки зрения любая программа должна выполняться на разных системах, отличающихся степенью масштабирования и комплектацией. С другой – программа, написанная с учетом параметров конкретной системы, будет выполняться значительно эффективнее. Разрешение этого противоречия зависит от потребности в степени оптимизации программ. Для больших задач должна достигаться наивысшая степень оптимизации программ, поэтому программист в данном случае должен иметь возможность влиять на выбор соответствия между программой и конфигурацией системы путем соответствующей настройки программы. Для других задач такая возможность может не использоваться.

Выполнение любой программы предваряется ее разбиением на фрагменты для обеспечения мультипоточного режима (режима микромультипрограммирования). При автоматическом режиме такой фрагментации необходимо обеспечить сохранение контента, требующегося для продолжения прерванной программы. Объем контента зависит от числа различных регистров и объема локальной памяти, содержимое которой может быть изменено при переключении программ. В целях сокращения сохраняемого контента при прерывании можно отмечать те места в программе, где объем контента сведен к минимуму (например только счетчик команд). Это может быть достигнуто, если программист разбивает свою программу на микромодули, каждый из которых начинается с новой загрузки данных, а заканчивается после завершения использования загруженных данных.

Еще одним инструментом повышения эффективности программ может быть явное выделение участков программы, содержащих только операции обработки данных, и участков, содержащих команды переходов. Такая возможность может быть использована для увеличения объема предварительно выбранных команд на участках первого типа и сокращения этого объема на участках второго типа.

Была исследована структура модуля скалярной обработки, ориентированного на конвейерный режим выполнения групп взаимосвязанных скалярных операций. Для реализации такой возможности необходима фрагментация программ со скалярной обработкой и выделение скалярных конвейерных нанопрограмм. Эта фрагментация также может задаваться программистом в явном виде.

Таким образом, для обеспечения высокой эффективности выполнения программ желательно в факультативном для программиста режиме ввести следующие механизмы:

- явного задания формы параллелизма для программного модуля или группы модулей;
- явного задания границ для фрагментации программ;
- явное разделение в программах функций обмена (подкачка данных) и вычислений;
- явное выделение в программах фрагментов безусловных вычислений и фрагментов управления ходом программы;
- явное задание выделения скалярных конвейерных нанопрограмм.

2.9. Основные новые особенности программного обеспечения

Для достижения высокой производительности при решении больших задач программное обеспечение системы должно реализовывать ряд функций, зависящих от архитектуры системы. При этом следует подчеркнуть, что оптимальная реализация этих функций не может быть полностью выполнена собственно операционной системой (по крайней мере на начальном этапе ее создания). Как уже было сказано, желательно наличие явного указания в прикладной программе на те или иные ее особенности. Однако если прикладной программист имеет дело с одной программой, в том числе и состоящей из большого числа модулей, то операционная система управляет работой большого числа таких программ и непосредственно управляет всеми ресурсами системы.

Анализ формы параллелизма выполняется как по явному указанию в программе, так и автоматически или полуавтоматически в интерактивном режиме. В результате программа разделяется на разделы с разными формами параллелизма. Эти разделы подготавливаются для последующей трансляции и загрузки в соответствующую часть вычислительной подсистемы. На данном этапе операционная система оценивает также масштаб задачи и другие параметры в целях определения наиболее подходящей для данного раздела части вычислительной подсистемы.

При наличии в системе специализированных процессоров выявляются разделы программы, предназначенные для этих процессоров. Эти

разделы также подготавливаются для последующей трансляции и загрузки в соответствующую часть вычислительной подсистемы.

В целях сокращения потерь на ожидание данных в операционной системе необходимо наличие функций организации программного макроконвейера, обеспечивающего совмещение операций пересылки массивов данных и вычислений. Исходная информация для формирования этого макроконвейера может быть в прикладной программе при явном задании обмена или может быть получена при анализе программы – автоматическом или полуавтоматическом.

Функция фрагментации программ в целях загрузки отдельных модулей в память различных частей вычислительной подсистемы, а также в целях обеспечения микромультипрограммного режима может выполняться операционной системой в соответствии с явным разбиением программ на микромодули, выполняемым прикладным программистом, или автоматически или полуавтоматически в интерактивном режиме. При этой фрагментации необходимо учитывать структуру и параметры вычислительной подсистемы, в том числе многоуровневую организацию.

В результате согласованных действий по подготовке программ и данных на этапе их загрузки должен быть сформирован пакет программы, состоящий из фрагмента (отрезка) программы и данных для выполнения этого фрагмента программы. Такое пакетирование программ направлено на максимальное повышение эффективности работы вычислительной подсистемы.

Одним из наиболее важных аспектов подготовки фрагментов программ и данных является достижение высокой степени локализации, что резко сокращает объем непроизводительного обмена информацией.

Достижение высокой эффективности при этом зависит от формы планирования выполнения программ. Желательно, чтобы в основном при трансляции использовалось статическое планирование.

2.10. Снижение потребляемой энергии

В настоящее время снижение потребляемой энергии обеспечивает саму возможность создания суперсистемы. Разработка и эксплуатация системы должны сопровождаться обеспечением этого фактора на всех этапах.

Основными инструментами для снижения потребляемой энергии по существу являются все рассмотренные выше архитектурные и про-

граммные механизмы повышения эффективности, меры по оптимизации конструкции и использования схемотехники на основе принципа близкодействия, а также программируемая функция отключения части оборудования при распределении программ.

Основными инструментами для снижения потребляемой энергии являются:

- взаимная адаптация архитектуры машин и прикладных программ и обеспечение более эффективного использования аппаратуры;
- простота системы команд процессоров основных и функционально-специализированных машин;
- сокращение потерь при передаче сигналов за счет использования схемотехники на основе близкодействия;
- сокращение паразитных обращений к памяти за счет оптимизации локализации данных на уровне аппаратных средств, системных и прикладных программ;
- сокращение паразитных перемещений массивов за счет использования в сети памяти обменно-редактирующих машин и соответствующих программ;
- оптимизация прикладных программ и эффективность распределенной программно-аппаратной системы распределения ресурсов и управления программами.

2.11. Основные отличительные особенности проекта

В заключение перечислим основные отличительные особенности проекта.

Неоднородность, мультиархитектура, взаимная адаптация архитектуры и программ, локализация данных.

Специализированные основные и вспомогательные вычислительные машины для суперсистем.

Специализация сетевой структуры. Иерархическая сеть памяти и редактирование данных. Горизонтальная межузловая сеть.

Мониторно-моделирующая подсистема. Новые требования к системным и прикладным программам. Распределенная операционная система. Сеть управления.

ГЛАВА 3.

АРХИТЕКТУРА СИСТЕМЫ

3.1. Отличительные особенности проекта мультиархитектурной системы

Основные проблемы создания высокопроизводительных вычислительных систем нашли отражение в различных аспектах и подходах, разработанных в проекте мультиархитектурной вычислительной суперсистемы. Мультиархитектура как парадигма для вычислительных систем основана на взаимной адаптации технологии, аппаратных средств, архитектуры, системного и прикладного программного обеспечения и вычислительных методов. В отличие от адаптивного суперкомпьютинга фирмы Cray, предполагающего адаптацию архитектуры к программам, взаимная адаптация обеспечивает более высокую эффективность и стимулирует создание новых вычислительных методов и моделей параллельного программирования.

В системе необходимо несколько типов процессоров с различными формами параллелизма. Для достижения высокой производительности, эффективности и малого потребления энергии их схемотехника должна быть достаточно простой, а также она должна быть построена с учетом принципа близкодействия. Наиболее рациональным является широкое использование конвейерных структур. Как известно, такие структуры являются основой векторных процессоров, в скалярных процессорах их эффективность сильно зависит от оптимизации программы.

Построение процессоров должно предусматривать их крупномодульную структуру, при которой любой процессор может быть скомпонован из унифицированных модулей. В зависимости от набора и числа тех или иных модулей возможно построение процессора с заданными характеристиками. Этот принцип важен со многих точек зрения – проектирования, изготовления, тестирования и, что самое важное, масштабирования процессоров.

3.2. Многомодульная масштабируемая основная машина

Одной из целей указанного подхода является многократное повышение производительности уипроцессора, выполняющего одну программу.

Так, унипроцессором является многомодульный масштабируемый процессор. Он является основой построения многомодульной масштабируемой основной машины, предназначеннной для выполнения прикладных программ. В состав процессора входят модули векторной обработки (МВО), которые программно, логически и топологически могут быть объединены в цепочку. Если программа содержит сложные векторные функции или наборы связанных между собой векторных вычислений, то длина цепочки векторных операций, выполняемых параллельно, может достигать десятков (рис. 3.1).

Кроме параллелизма за счет зацепления векторных операций возможна и вторая форма параллелизма – параллельная работа нескольких конвейеров. Для эффективной работы в данном случае необходимо разбиение длинных векторов на отрезки, каждый из которых выполняется на самостоятельной конвейерной ветке.

Модуль МВО содержит несколько параллельных функциональных устройств. Их параллельная работа зависит от степени оптимизации программы с точки зрения обеспечения готовности данных для выполнения параллельных операций. В состав модуля входит буферная память, адресуемые скалярные и векторные регистры. Входной коммутатор обеспечивает передачу данных от адресуемых и входных регистров на вход функциональных устройств, а выходной – от функциональных устройств на вход адресуемых и выходных регистров.

Выходные регистры модуля соединяются с входными регистрами соседнего модуля, что обеспечивает выполнение цепочки векторных операций. Для управления цепочкой модулей на ее вход подается векторная мультикоманда, содержащая набор команд для цепочки модулей МВО. В каждом модуле первая команда передается в регистр команды для выполнения, а остальная часть мультикоманды передается в соседний модуль. Такая конвейерная схема выдачи векторной мультикоманды обеспечивает необходимый темп запуска операций в модулях. Эта схема не зависит от числа модулей в цепочке.

Модули векторной обработки являются независимыми векторными процессорами и при включении в их состав блока управления могут выполнять собственную векторную программу. Эта форма параллелизма возможна, если удастся разбить программу на независимые фрагменты. Дополнительным источником параллелизма может являться макроконвейер таких программных фрагментов.

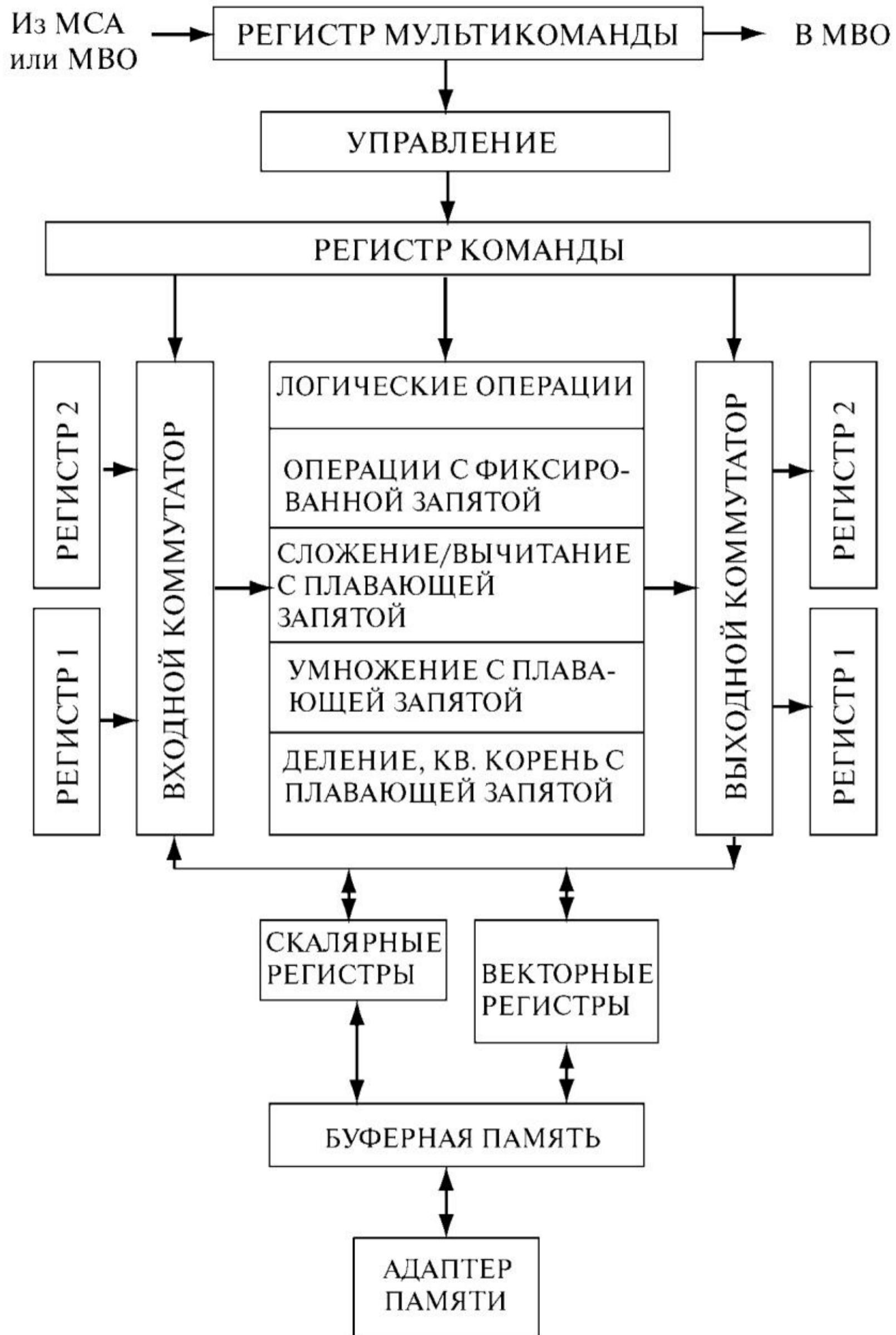


Рис. 3.1. Модуль векторной обработки

Производительность, измеряемая в числе результатов операций с плавающей запятой в такт, для модуля векторной обработки в пределе может достигать суммарного числа функциональных устройств в модуле.

3.2.1. Векторные и скалярные мультикоманды

Для управления цепочкой модулей векторной обработки и выполнения скалярных операций используется модуль скалярно-адресный (МСА) (рис. 3.2). Кроме выполнения скалярной части программы этот модуль выдает векторную мультикоманду, состоящую из команд для каждого модуля векторной обработки, в первый модуль векторной обработки.

Модули скалярно-адресные, управляющие ветвями модулей векторной обработки, в свою очередь, управляются со стороны модуля диспетчерского управления (МДУ), единственного в унипроцессоре (рис. 3.3). Этот модуль выполняет операции управления пересылками данных, передачи управления, а также выдачи скалярных мультикоманд в модули скалярно-адресные. В каждом модуле, кроме набора функциональных устройств и схем управления, имеются адресуемая регистровая память и буферная память. Из модуля МДУ выдается скалярная мультикоманда для нескольких модулей МСА аналогично выдаче векторной мультикоманды из МСА.

Указанных модулей достаточно для построения различных векторных и скалярных процессоров. Возможно использование и других специализированных модулей. Примером такого модуля может быть модуль скалярной обработки, в котором набор конвейерных устройств используется для параллельного выполнения небольших отрезков скалярных программ, где последовательность операций обеспечивает загрузку функциональных устройств, а наличие независимых отрезков программ обеспечивает запуск операции в каждом устройстве каждый такт. Возможно введение узкоспециализированных модулей для задач определенного класса.

Кроме того, в состав так называемой масштабируемой основной машины (рис. 3.4) входит оперативная память с адаптером сопряжения и буфер директив. Оперативная память с одной стороны связана с модулями процессора, для чего используется коммутатор типа 2D-тора. Буфер директив обеспечивает связь процессора с сетью управления.

Забегая вперед, отметим, что по сети памяти выполняются загрузка программ и пересылка данных для машины. По сети управления в буфер

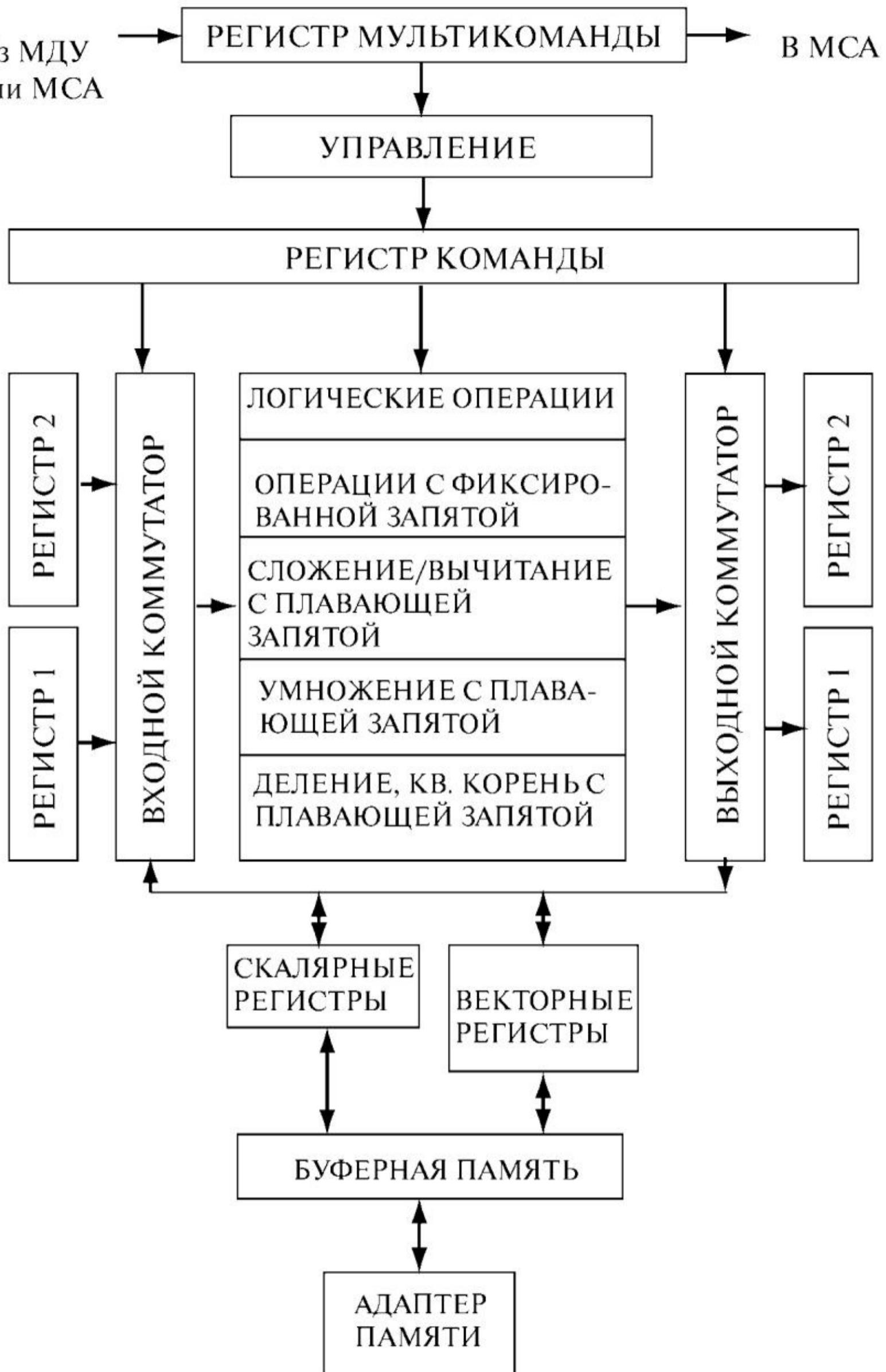


Рис. 3.2. Модуль скалярно-адресный

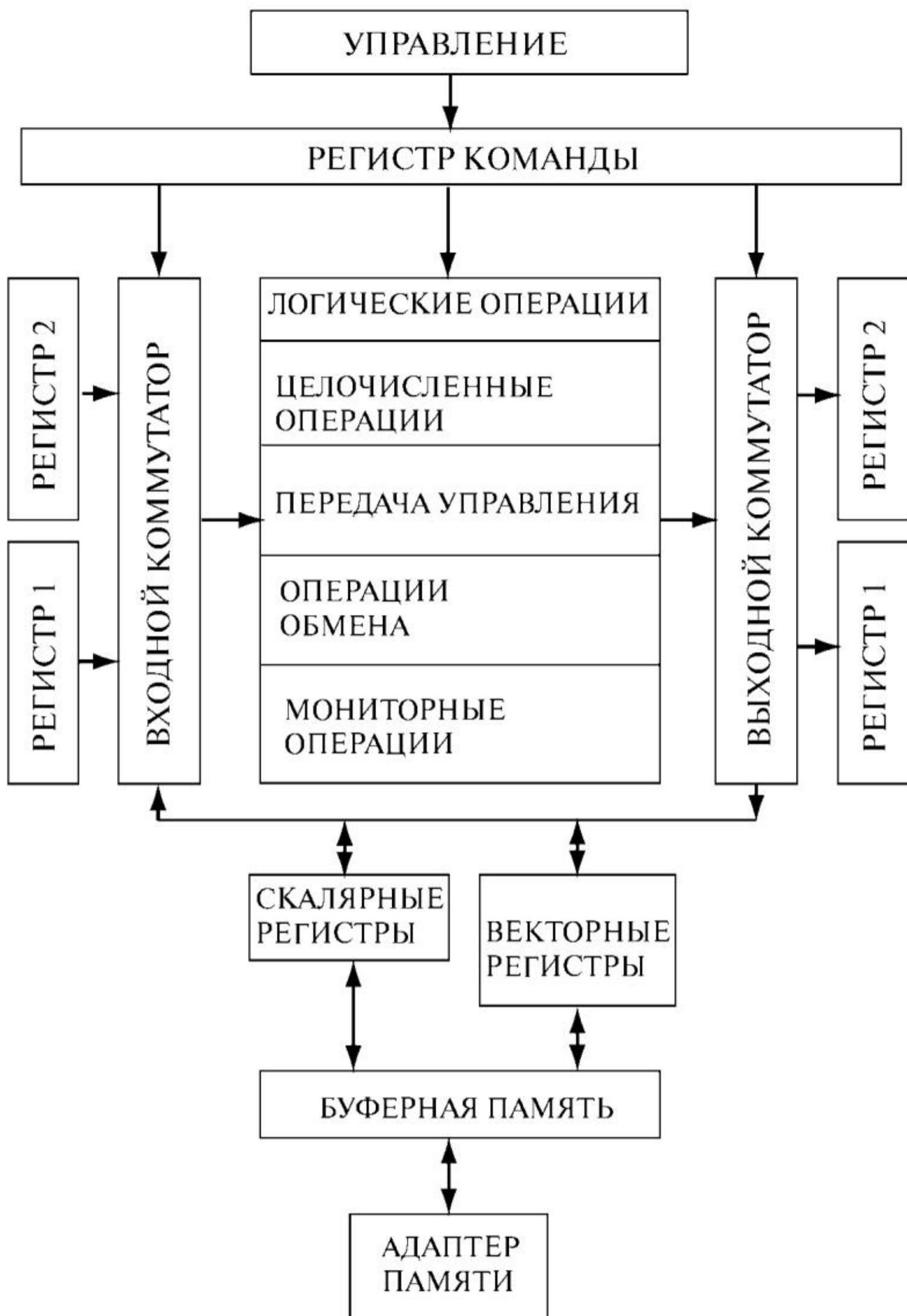


Рис. 3.3. Модуль диспетчерского управления

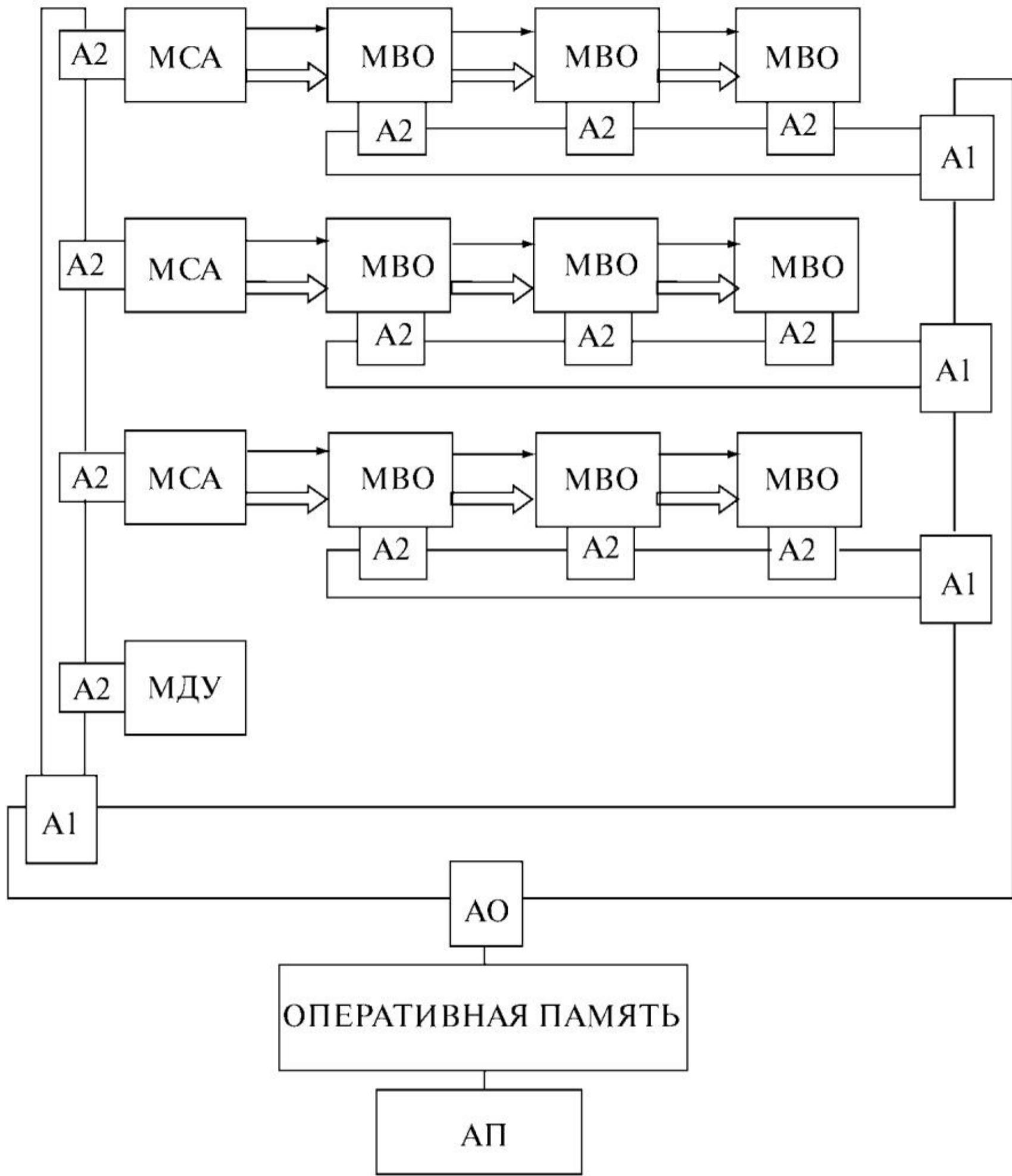


Рис. 3.4. Масштабируемая основная машина

директив поступают директивы для запуска программ машины. Сама машина формирует директивы и помещает их в соответствующий буфер при выполнении мониторных команд и при прерываниях.

Производительность масштабируемой основной машины в пределе равна суммарной производительности всех модулей.

3.2.2. Локализация данных

Обеспечение высокой производительности в многомашинной системе на основе масштабируемых процессоров возможно только при разрешении проблемы, связанной с более низкой по сравнению с процессорами пропускной способностью памяти. Для разрешения этой проблемы необходимо стремиться к локальности данных. Возможны следующие механизмы для достижения этой цели:

- аппаратный вероятностный механизм, основанный на использовании одного или более уровней кэш-памяти;
- программный механизм оптимизации программ на уровне трансляции и управления данными в рамках операционной системы;
- аппаратно-программный механизм, основанный на планировании пересылки данных на уровне прикладной программы и наличии соответствующих аппаратных и программных средств. Этот механизм должен также включать возможность интерактивного взаимодействия прикладного программиста с программами оптимизации во время отладки программы.

Выбор того или иного механизма зависит от характера программ, структуры данных и масштабов системы. Так, имеется большая разница в подходах между векторизуемыми программами и программами со скалярной обработкой. По-видимому, необходимо сочетание разных механизмов. Существенную роль играет выбор структуры системы и структуры памяти.

В масштабируемой машине, ориентированной на обработку больших массивов данных на длинных цепочках модулей векторной обработки, неэффективно применение кэш-памяти. В каждом модуле должна быть память, состоящая из адресуемых скалярных и векторных регистров, и буферная (локальная) память. В состав масштабируемой машины входит оперативная память. Необходима предварительная загрузка данных в оперативную память в соответствии с планированием вычислений со стороны прикладной задачи. Для более эффективного выполнения такой пересылки данных необходимо использование аппаратно-программных средств, основанных на использовании специализированных обменно-редактирующих машин, выполняющих программно-управляемый обмен, исходные данные для которого поступают от прикладной задачи. Таким образом, программист получает возможность управления перемещением данных и повышения степени их локализации.

3.2.3. Масштабируемая векторная машина

Основным назначением масштабируемой векторной машины является выполнение программ с явным параллелизмом на уровне данных. Такие программы должны быть векторизованы и ориентированы на выполнение сложных векторных операций.

В масштабируемой векторной машине используются две основные формы параллелизма:

- выполнение векторных операций с помощью большого числа конвейерных функциональных устройств, при этом параллелизм обеспечивается как за счет выполнения независимых операций, так и за счет зацепления векторных операций, в частности при выполнении сложных векторных операций;
- параллельное выполнение групп векторных операций с помощью отдельных наборов устройств для выполнения векторных операций.

Архитектура, методология программирования, структура аппаратных средств, схемотехника и конструкция масштабируемой векторной машины ориентированы на достижение максимальной производительности при выполнении программ с указанными свойствами. Схемотехника и конструкция процессора основаны на принципе близкодействия, при реализации которого обеспечивается передача данных между соседними элементами, благодаря чему минимизируются потери на распространение сигналов. Основой схемотехники являются синхронные конвейерные схемы, используемые как в функциональных устройствах, так и в схемах управления и обмена. Для достижения высокой производительности необходимо обеспечить параллельную работу большого числа функциональных устройств. Для этого могут быть использованы цепочки устройств и параллельные ветви из таких цепочек. Наилучшие условия с точки зрения загрузки устройств и достижения высокой производительности возникают при выполнении векторных операций и цепочек этих операций при вычислении сложных векторных функций.

3.2.4. Масштабируемая конвейерная скалярная машина

Наиболее эффективным схемотехническим приемом повышения производительности аппаратных средств является использование конвейерных структур, в особенности синхронных конвейерных схем. Они широко используются как в векторных, так и в скалярных процессорах,

как на уровне выборки, дешифрации и выдачи команд, так и при выполнении собственно операций обработки. Если в векторных процессорах проблема загрузки процессора решается достаточно просто в силу независимости обработки элементов вектора, то в скалярных процессорах зависимость последовательных операций друг от друга, например использование результата операции в качестве операнда следующей команды, приводит к приостановке подачи operandов на вход конвейерного устройства и эффективность его использования падает.

Известно много различных методов преодоления этих проблем. На программном уровне возможна оптимизация программы в целях сокращения времени ожидания завершения операции в конвейере. На аппаратном уровне используется выполнение команд не в порядке их размещения в программе, а в зависимости от наличия ресурсов для выполнения команды. Это фактически простейший вариант управления выполнением программы со стороны данных (т.н. data flow). Реализация этого метода связана с резким увеличением оборудования для обеспечения корректного выполнения программы.

Аналогичные проблемы возникают при ветвлении программ. В таких случаях для исключения приостановки конвейера выполняется предсказание направления перехода и осуществляется выборка команд для возможного ветвления. Реализация данного метода также связана с большим объемом дополнительного оборудования.

Известны методы повышения степени параллелизма за счет параллельного выполнения двух или более команд и за счет параллельной работы нескольких арифметико-логических устройств. К аппаратным методам повышения параллелизма относится суперскалярная организация процессора, при которой выполняются выборка, дешифрация и выдача одновременно нескольких команд. Обычно такой подход эффективен для процессоров с RISC-архитектурой. При этом выполняется динамическое планирование. Успех данного метода зависит от потенциального параллелизма в программе, т.е. необходима некоторая оптимизация программы на статическом уровне. К другому подходу относится структура процессора с очень длинным командным словом (VLIW). При этом аппаратура должна обеспечивать параллельное выполнение всех команд в таком слове. В этом случае необходимы статическое планирование и подготовка программы, состоящей из длинных командных слов. Успех здесь также зависит от

оптимизации программы за счет группировки нескольких независимых операций.

Следующей проблемой в скалярных процессорах является влияние задержек при обращении к памяти. Для сокращения этих задержек используются адресуемые регистры или кэш-память. Адресуемые регистры позволяют на уровне статического планирования обеспечить подготовку данных. Кэш-память позволяет динамически выделять те данные, обращение к которым наиболее вероятно. В современных микропроцессорах имеется два или три уровня кэш-памяти. Однако вследствие более быстрого роста быстродействия логических схем по сравнению с памятью указанные методы не обеспечивают необходимого темпа поступления данных из памяти. Одним из подходов преодоления указанных задержек является т.н. multithreading. При этом программа разбивается на «легкие» процессы (threads – буквально нити), использующие единое адресное пространство, т.е. при переключении процессов не выполняется смена таблиц для преобразования виртуальных адресов в физические. На аппаратном уровне обеспечивается быстрое переключение процессора с одного такого процесса на другой с одновременным переключением контента процессов. Для хранения контента используется быстрая память, не создающая задержек для процессора. Подкачка этой памяти для неактивных процессов происходит параллельно с работой процессора. В процессорах с несколькими параллельно работающими функциональными устройствами возможно параллельное выполнение нескольких процессов. Такая организация называется simultaneous multithreading. Указанные английские термины относятся и к программным, и к аппаратным средствам. В русской литературе, как правило, используется фонетическая калька – мультитредовая организация. С нашей точки зрения, более адекватным термином может быть микромультипрограммирование. Подобный термин встречался и в англоязычной литературе на ранних этапах развития этих идей.

Все указанные методы повышения производительности используются в стандартных микропроцессорах. При этом следует подчеркнуть, что стремление сохранить совместимость с процессорами серии x86 и повысить степень параллелизма работы процессора в ряде случаев привело к непропорциональному росту объема оборудования, что, в свою очередь, увеличивает длину межсоединений, снижает тактовую частоту и повышает потребляемую энергию, из-за чего снижается возможная

плотность размещения схем в многопроцессорных системах. Поэтому для масштабируемого конвейерного скалярного процессора, используемого в мультиархитектурной вычислительной суперсистеме, нужен новый подход к повышению производительности.

На схемотехническом уровне этот подход должен быть основан на принципе близкодействия, т.е. схемы, от которых зависит скорость работы, должны по возможности располагаться рядом. Функциональные устройства должны быть построены на синхронных конвейерных структурах. Целью при построении всех других схем должна быть максимальная загрузка функциональных устройств на реальных программах. Естественно, это также связано с созданием специальной методики программирования.

Что касается подходов к программированию, при создании суперсистемы следует исходить из принципа приоритета достижения наибольшей производительности программ после их оптимизации и трансляции даже за счет увеличения трудоемкости написания и отладки прикладных программ. Для массовой вычислительной техники характерен приоритет снижения трудоемкости программирования часто в ущерб производительности.

Целью при разработке масштабируемого конвейерного скалярного процессора может служить следующая гипотетическая модель. Вычислительные ресурсы процессора представляют собой цепочку из большого числа функциональных устройств, а все остальные ресурсы – управление, память и программы – служат для обеспечения непрерывной работы всех функциональных устройств. За счет близкодействия может быть достигнута максимальная тактовая частота, а производительность будет равна i операций в такт, где i – число устройств в цепочке.

Естественно, что одна программа при наличии взаимозависимости между соседними операциями не может обеспечить требуемую загрузку конвейера. Поэтому необходимо на вход конвейерной цепочки устройств каждый такт выдавать данные для новой независимой программы. Число таких программ для загрузки всего конвейера зависит от его длины, т.е. от числа устройств в цепочке. Новая программа, зависимая от предыдущей, может быть запущена только после завершения этой предыдущей программы. В промежутке между запусками этих двух зависимых программ может быть запущено столько программ, сколько однотактных ступеней содержится в цепочке. Если предположить, что

во всех устройствах имеется по s ступеней конвейера, то число программ в цепочке будет равно $n \cdot s$.

Указанные программы должны содержать набор команд, которые могут быть выполнены в цепочке функциональных устройств. Естественно, что к таким командам относятся только арифметические команды. Все остальные типы команд – логические, переходы, обращение к памяти и управляющие – должны выполняться в специальном управляющем процессоре (модуле). Этот же модуль должен выдавать набор команд для цепочки. Этот набор будем называть скалярной мультикомандой. С другой стороны, этот набор является отрезком программы, который будем называть нанопрограммой. В общем случае в нанопрограмму может входить такая последовательность арифметических команд, которая целиком не может быть включена в скалярную мультикоманду. В этом случае на основе нанопрограммы формируется несколько скалярных мультикоманд. Процедуру переключения нанопрограмм соответственно будем называть наномультипрограммированием.

Указанный подход отличается от микромультипрограммирования тем, нанопрограмма состоит из их мультикоманд, которые ориентированы на конкретную аппаратную реализацию в виде цепочки функциональных устройств, а не на универсальное АЛУ. Кроме того, момент переключения нанопрограмм и выдачи мультикоманд определяется аппаратно и, как правило, должен выполняться каждый такт. Цели этих методов разные, хотя и имеется некоторое сходство. Следует подчеркнуть, что нанопрограммы существенно отличаются от «легких» процессов, используемых при микромультипрограммировании.

Несмотря на то что нанопрограммы выдаются несколькими процессорами, они могут относиться как к одной программе, так и к различным программам. При наличии нескольких специальных управляющих процессоров (модулей) данная структура может рассматриваться как мультипроцессорная. Для параллельного выполнения нанопрограмм возможно включение двух или более цепочек функциональных устройств, связанных с единым устройством для хранения нанопрограмм.

Основные команды управляющего процессора должны быть однотактными. Для выполнения одиночных арифметических команд необходимо предусмотреть их наличие в системе команд. При этом за счет оптимизации программ влияние большого времени их выполнения

может быть сведено к минимуму. Выполнение команд обращения к памяти должно совмещаться с собственно обработкой.

Отличие от структуры с длинным командным словом заключается в том, что длинное командное слово содержит ряд независимых команд, которые могут выполняться параллельно на нескольких устройствах. Мультикоманда состоит из ряда последовательных команд, которые, как правило, взаимосвязаны друг с другом, т.е. результат одной команды может быть операндом следующей.

Отличительной чертой процессора является наличие последовательно соединенных функциональных устройств и локальной памяти для операндов и результатов операций. Группа таких устройств вместе с памятью и схемами управления может быть объединена в некоторый унифицированный модуль скалярной обработки (МСО). В минимальной конфигурации в состав модуля должны входить устройство сложения-вычитания чисел с плавающей запятой и устройство умножения-деления чисел с плавающей запятой. Учитывая тот факт, что основным режимом работы цепочки устройств является выполнение последовательных и связанных между собой команд, желательно обеспечить такой же режим и в самом модуле, т.е. возможность последовательной работы указанных устройств, при этом последовательность их работы должна задаваться программой. В состав модуля также входят адресуемые регистры, буферная память, входной и выходной коммутатор данных и схемы приема команд из соседнего модуля, выдачи команд в следующий модуль и схемы управления (рис. 3.5).

Основная часть программы процессора выполняется в модуле МДУ. Кроме того, используется модуль МСА для выполнения основной части программы, формирования скалярных мультикоманд и выдачи их в модуль МСО. Модуль МСА обеспечивает распараллеливание программы и обеспечения полной загрузки цепочки модулей скалярной обработки. Для лучшего согласования скоростей работы модулей МСА и модуля МСО выдача скалярных мультикоманд в модуль МСО должна выполняться из специального буфера мультикоманд. В процессе выполнения программы в модуле МСА формируются одиночные или группы скалярных мультикоманд, которые помещаются в указанный буфер мультикоманд. Выборка мультикоманд из буфера осуществляется аппаратно и обеспечивает выборку мультикоманды каждый такт, благодаря чему обеспечивается загрузка цепочки модулей МСО.

Другим фактором обеспечения такой загрузки является своевременная подготовка данных и размещение их в буферной памяти модулей

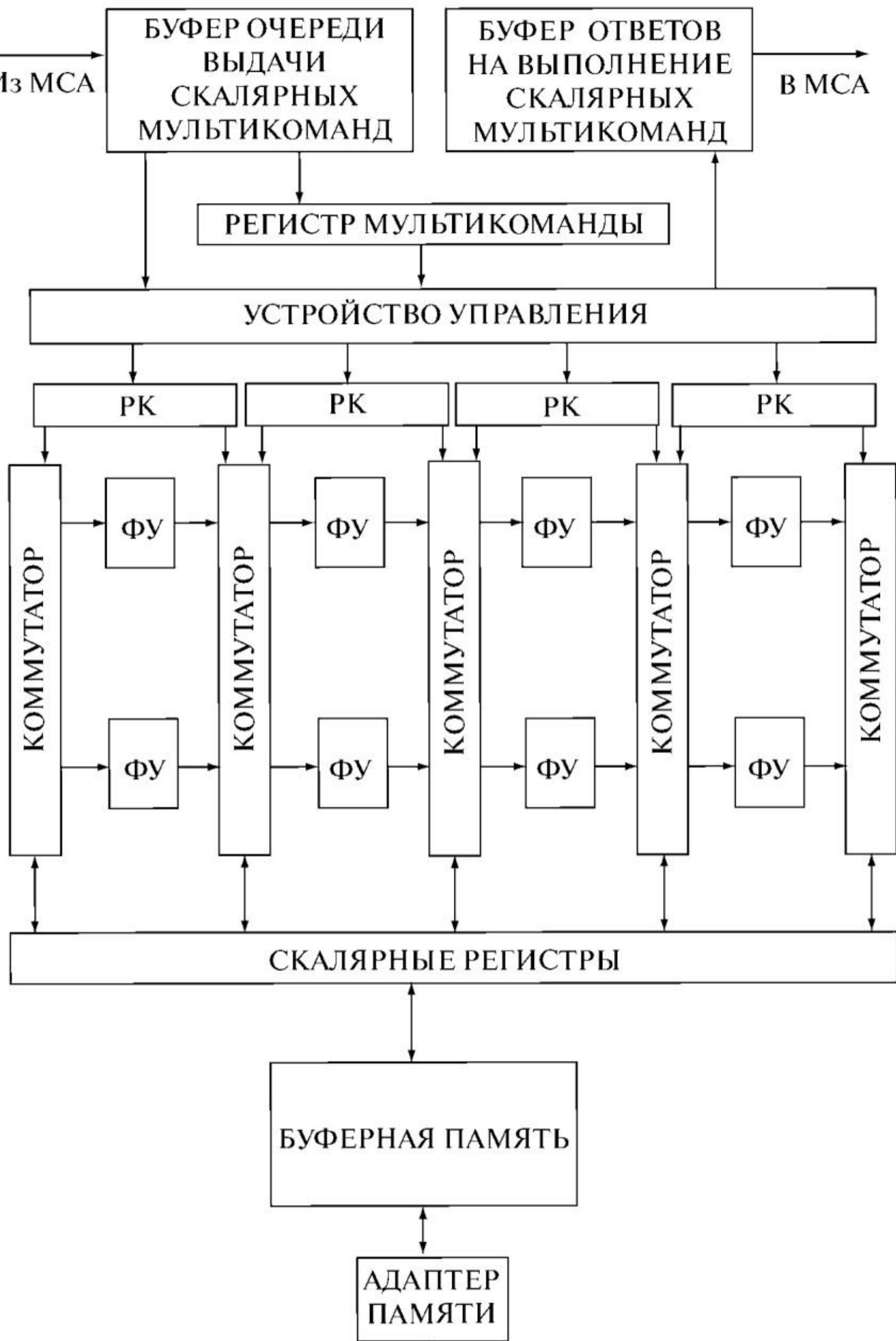


Рис. 3.5. Модуль скалярной обработки

МСО. Аппаратной составляющей в данном случае является контроллер обмена между оперативной памятью процессора и модулями МСО, управляемый со стороны модуля МДУ. Программной составляющей, обеспечивающей загрузку модулей МСО, является специальная подготовка программ, при этом те данные, которые необходимы для выполнения мультикоманд, должны помещаться в буферную память модулей МСО с опережением по отношению к выдаче соответствующих мультикоманд. При этом для каждой мультикоманды и соответствующей команды модуля МСО должна выделяться своя область индивидуального контента в буферной памяти, т.е. буферная (локальная) память модуля разбивается на группы, при этом каждая запускаемая мультикоманда соответствует своей группе данных в памяти. Информация об этом соответстии помещается во все модули вместе с выдачей или до выдачи мультикоманды.

Информация об окончании нанопрограммы должна помещаться в специальный буфер возврата в основную программу.

Масштабируемая конвейерная скалярная машина (рис. 3.6) может иметь разные конфигурации и уровни масштабирования. Во-первых, в каждом процессоре может быть разное число модулей МСА, во-вторых, в цепочке может быть разное число модулей МСО, в состав которых, в свою очередь, может входить разное число функциональных устройств. Таким образом может быть обеспечено распараллеливание ветвей одной программы и выполнение последовательности скалярных операций обработки различной длины.

3.2.5. Масштабируемые мультиархитектурные машины

Для задач, в которых имеется очень тесное взаимодействие между векторными вычислениями и скалярными вычислениями с применением описанных выше методов, основанных на применении модуля МСО, целесообразно в процессоре иметь параллельные ветви для векторной и скалярной обработки. В каждой ветви модуль МСА связан либо с цепочкой модулей МВО, либо с модулями МСО.

3.2.6. Масштабируемые специализированные машины

Для реализации алгоритмов для узкого класса задач возможно создание специализированных модулей. Включение таких модулей в состав соответствующей масштабируемой машины позволяет построить специализированную масштабируемую машину.

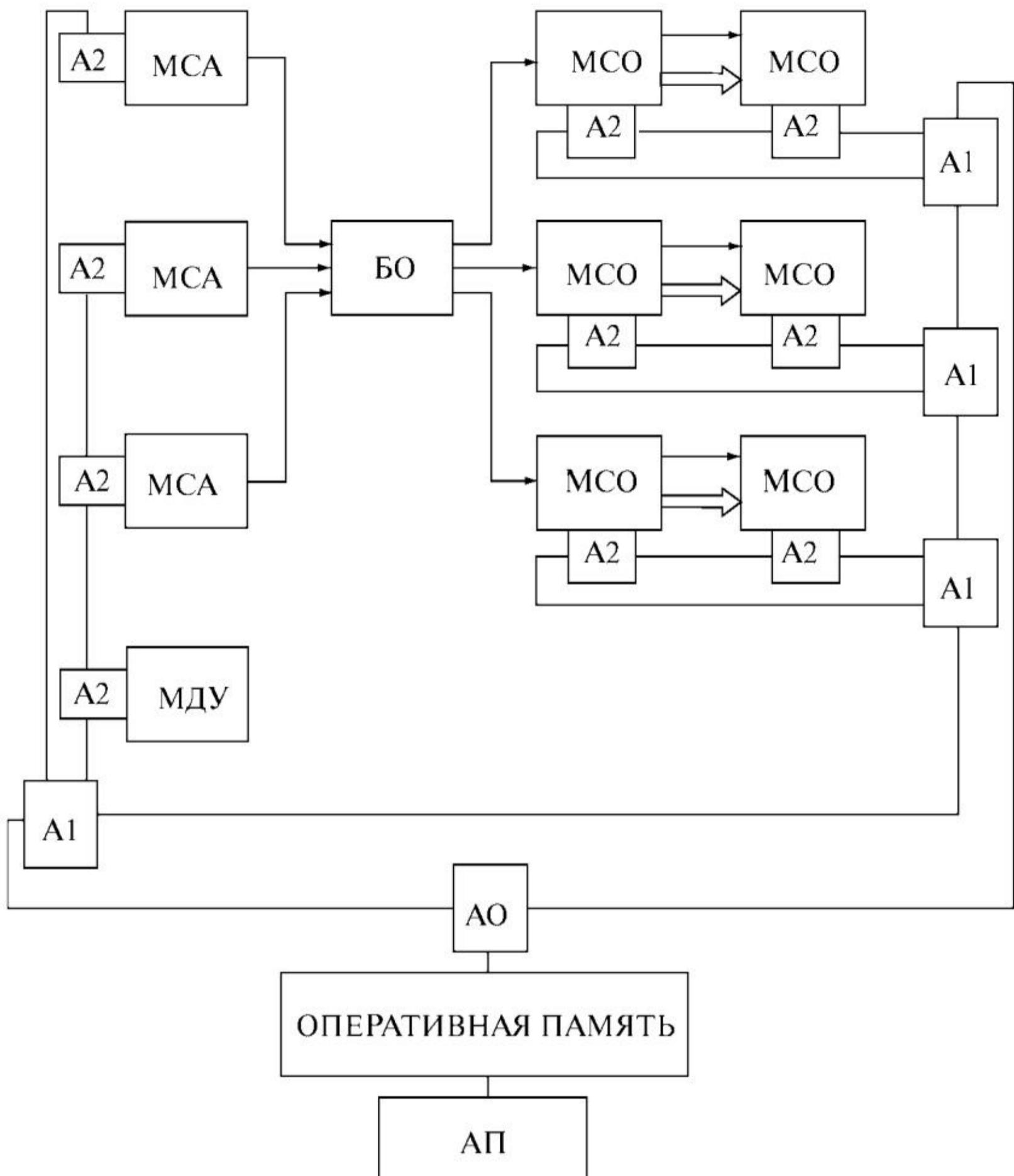


Рис. 3.6. Масштабируемая скалярная конвейерная машина

3.3. Сетевая структура системы

Сетевая структура системы соответствует основным функциям, связанным с взаимодействием программ, выполняемых на различных машинах, с обменом с памятью и с управлением прикладными программами. Указанные функции определяют структуру сети, конфигурацию межсоединений и коммутации, архитектуру специализированных машин и особенности программного обеспечения.

3.3.1. Сеть памяти

Сеть памяти системы, предназначенная для управления пересылкой данных, построена по иерархическому принципу. В основе ее организации лежит многоуровневая коммутация. На каждом уровне коммутации к одному объекту более высокого уровня подключается определенное число объектов более низкого уровня. Число этих объектов равно коэффициенту коммутации. Оптимальным является вариант, при котором число уровней иерархии вычислительных средств и сети памяти и число уровней коммутации совпадают. На каждом уровне объединяются вычислительные ресурсы – абоненты, у каждого из которых имеется собственная память. В свою очередь, объединяющий их ресурс также имеет собственную память. Собственную память объединяемых ресурсов будем называть абонентской памятью, а собственную память объединяющего ресурса – центральной памятью данного уровня. Коммутатор, соединяющий центральную память с абонентскими памятьями, управляет интеллектуальным контроллером, или обменно-редактирующей машиной. Программы этой машины позволяют управлять адресацией данных при обмене между центральной и абонентской памятью. Наиболее естественным режимом является последовательная адресация (изменение адреса на единицу) со стороны абонентской памяти и задаваемая программой формула вычисления адресов со стороны центральной памяти, например, с постоянным шагом. Однако возможны и варианты двухстороннего программного вычисления адресов, при этом адаптер абонентской памяти должен также иметь интеллектуальный контроллер, формирующий адреса при обмене. Обмен между абонентскими памятьми осуществляется через центральную память. Таким образом, на каждом уровне иерархии системы имеется общая память, с одной стороны, доступная всем объектам на данном уровне, а с другой – с помощью обменно-редактирующей машины связанная с общей памятью более высокого уровня. На некоторых уровнях в состав общей памяти, кроме памяти с прямым доступом, может входить и дисковая память с соответствующим контроллером.

Функции сети памяти:

- загрузка программ, данных и директив во все машины системы.
- Загрузка программ и данных в функционально-специализированные

машины осуществляется напрямую из адаптера сети памяти – в обход обменно-редактирующей машины;

- программно-управляемое редактирование данных в процессе обмена между центральной памятью (общей памятью более высокого уровня) и абонентской памятью (общей памятью более низкого уровня) на выходе центральной памяти;
- программно-управляемое редактирование данных в процессе обмена между центральной памятью (общей памятью более высокого уровня) и абонентской памятью (общей памятью более низкого уровня) на входе абонентской памяти;
- пересылка управляющей информации для управления и контроля пересылки информации.

Многоуровневая коммутация обеспечивает низкий коэффициент ветвления на каждом уровне, а при редактировании данных невозможна работа двух или более потоков данных, поэтому возможно применение коммутации каналов, благодаря чему обеспечивается достижение высокой пропускной способности. В качестве коммутаторов может быть использована структура типа «толстое дерево» или 2D.

3.3.2. Сеть управления

В большой системе может быть объединено и параллельно использовано для решения единой задачи весьма значительное число машин. При этом объединении должны быть обеспечены такие функции системы, как пересылка данных для выполнения программ, обмен информацией между машинами в рамках выполнения единой программы и выполнение функций операционной системы, таких как трансляция и загрузка программ, распределение ресурсов, планирование и управление вычислительным процессом. В мультиархитектурной системе к этому добавляются функции анализа частей задачи для выявления формы параллелизма и соответствующей подготовки задачи. Для анализа задач возможно применение специализированных моделирующих машин, их программное обеспечение на первом этапе должно осуществлять интерактивное взаимодействие с прикладным программистом.

В связи с необходимостью реализации последней функции, а также в связи с проблемной ориентацией основных машин функции опера-

ционной системы должны выполняться на выделенных для этой цели машинах, при этом архитектура таких управляющих машин может быть упрощена за счет исключения операций с плавающей запятой, сокращения числа типов данных, применения конвейерных структур с малым числом станций и т.п. Сеть управления и набор управляющих и моделирующих машин вместе с соответствующим программным обеспечением являются мониторно-моделирующей подсистемой.

Механизмом управления являются директивы, размещаемые в специальных областях оперативной памяти, называемых буфером директив. Они помещаются в буфер в порядке естественной очереди и поступают на вход процессора. Директивы предназначены для следующих функций:

- запуск программы машины;
- выдача информации об окончании программы;
- сохранение информации при прерывании процессора и запуск программы с точки прерывания.

Взаимодействие основных и функционально-специализированных машин сопровождается формированием и выдачей следующих директив:

- директивы для запуска программ основных машин поступают от управляющих машин вычислительного узла;
- директивы для управляющих машин поступают от управляющих машин более высокого уровня;
- директивы от основных машин об окончании программ поступают в управляющую машину своего вычислительного узла;
- директивы для запуска программ сетевых машин поступают от управляющих машин вычислительного узла;
- директивы для запуска программ обменно-редактирующих машин поступают от управляющих машин своего уровня.

Сеть управления осуществляет пересылку директив и запись их в буфер директив, размещаемый в оперативной памяти каждой машины. Поэтому для сети управления можно использовать инфраструктуру сети памяти без применения обменно-редактирующих машин. Как было указано выше, обменно-редактирующие машины используются только при обмене данными между общими памятью соседних уровней. Пересылка директив и другой управляющей информации происходит без участия обменно-редактирующих машин. В этом случае может быть использован режим коммутации сообщений.

3.3.3. Межузловая сеть

Межузловая сеть предназначена для обеспечения взаимодействия программных модулей, выполняемых в различных вычислительных узлах. Эти программные модули также взаимодействуют с сетью управления и с сетью памяти. Основной проблемой является оптимальное построение взаимодействия программных модулей с сетями в целях повышения эффективности работы аппаратных средств. Дополнительными факторами, усложняющими решение данной проблемы, являются неоднородность системы, ее мультиархитектура и сложная иерархическая структура.

Хотя межузловая сеть имеет структуру с горизонтальными связями и логически не является иерархической, конструктивные особенности больших систем, состоящих из элементов разных уровней – модулей, блоков и стоек, вынуждают на физическом уровне иметь иерархическую структуру сети. Так, связи в пределах стойки достаточно короткие и потери на передачу сообщений минимальны. Связи между стойками отличаются между собой, естественно, они зависят от удаления стоек друг от друга. Для повышения эффективности целесообразно, например, для таких удаленных соединений использовать высокоскоростные линии, в частности оптоволоконные.

Для реализации межузловой сети в состав каждого вычислительного узла включена сетевая машина, связанная с коммутатором, входящим в состав коммутационной структуры межузловой сети.

Коммутационная структура межузловой сети зависит от масштабов системы. Для малых систем, содержащих десятки вычислительных узлов, достаточна коммутационная структура типа 2D-тора с 4-портовыми коммутаторами в узлах. Для средних и крупных систем, содержащих сотни или тысячи вычислительных узлов, наиболее эффективной является коммутационная структура типа 3D (тор). Для очень больших систем, содержащих более нескольких десятков или сотен тысяч вычислительных узлов, необходимо применение двухступенчатой коммутационной структуры типа 5D или 6D и соответствующие многомерные коммутаторы.

В межузловой сети имеется большое число абонентов и возможна одновременная пересылка большого числа сообщений, поэтому должна использоваться коммутация сообщений, предполагающая наличие на входе и выходе абонентов и узлов сети буферной памяти для сообщений.

3.4. Иерархическая структура системы

Система при значительных масштабах должна иметь многоуровневую структуру и состоять из объектов, объединяющих оптимальное число объектов более низкого уровня. В результате возможно сокращение потерь на пересылку данных, упрощение систем коммутации и реконфигурации. Предлагается использовать следующие уровни иерархии в системе:

- масштабируемая основная машина, состоящая из 2–1000 модулей;
- вычислительный узел, состоящий из 2–16 машин;
- мультиархитектурный мультикомпьютер, состоящий из 16–128 узлов;
- мультиархитектурный вычислительный комплекс, состоящий из 1–16 мультикомпьютеров;
- вычислительная подсистема, состоящая 1–16 комплексов.

Таким образом, минимальная комплектация – это основная машина, включающая 2 модуля, максимальная – вычислительная подсистема, включающая до 0,5 млрд модулей. При тактовой частоте в модулях 2 ГГц система может иметь производительность 1 exaflops.

Вычислительный узел ВУ (рис. 3.7) состоит из набора масштабируемых основных машин (МОМ), управляющей машины (УМ), общей памяти с подключенной к ней обменно-редактирующей машиной (ОРМ), сетевой машины (СМ), коммутатора (К) и адаптера сети памяти (АП). Для объединения машин, входящих в состав вычислительного узла, используется коммутатор типа 2D-тора.

На следующих уровнях – мультикомпьютера (МК) (рис. 3.8), мультиархитектурного вычислительного комплекса (ИВК) (рис. 3.9), вычислительной системы (рис. 3.10) кроме объединяемых объектов предыдущего уровня имеются управляющие машины, устройства общей памяти с обменно-редактирующими машинами, коммутаторы и адаптеры сети памяти. На некоторых уровнях может присутствовать дисковая память с дисковым контроллером (ДК), периферийная машина (ПМ) и внешние устройства (ВУ). На уровне вычислительной системы имеется центральная управляющая машина (ЦУМ), моделирующая машина (ММ), периферийная подсистема (ППС), управляемая периферийными машинами, архивная память (АП), средства локальной сети (ЛС) и системы отображения (СО) (рис. 3.11).

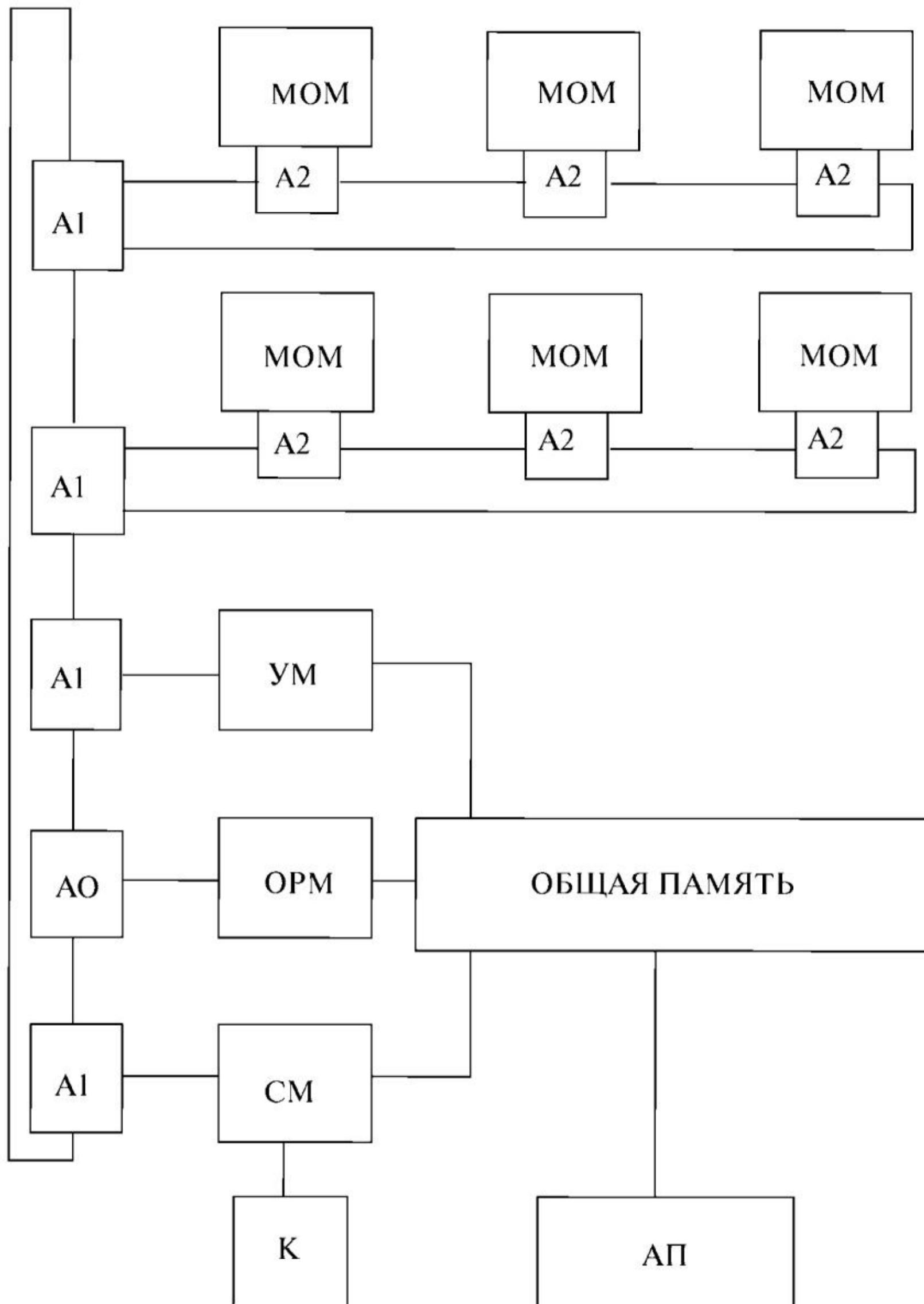


Рис. 3.7. Вычислительный узел

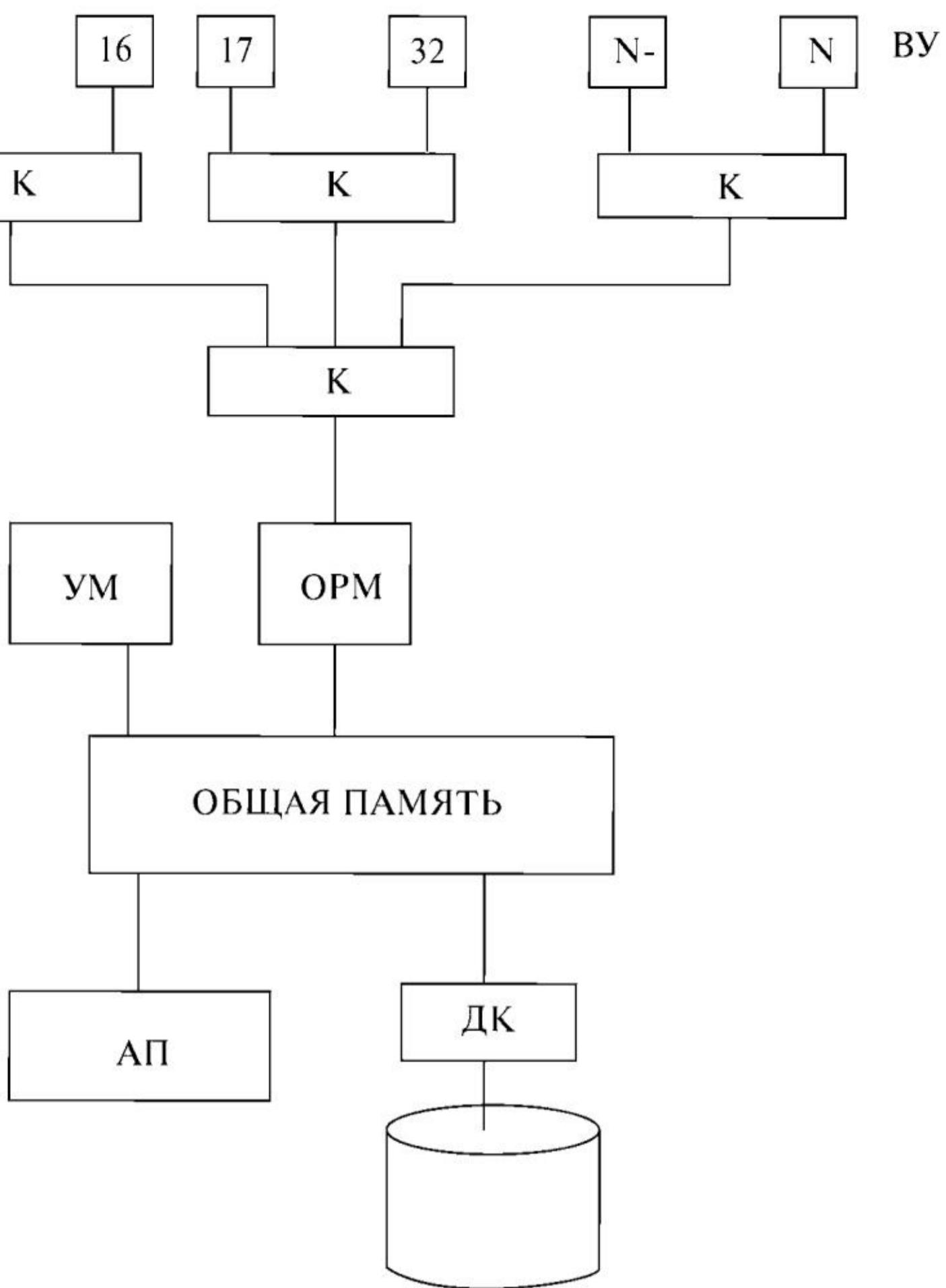


Рис. 3.8. Мультикомпьютер

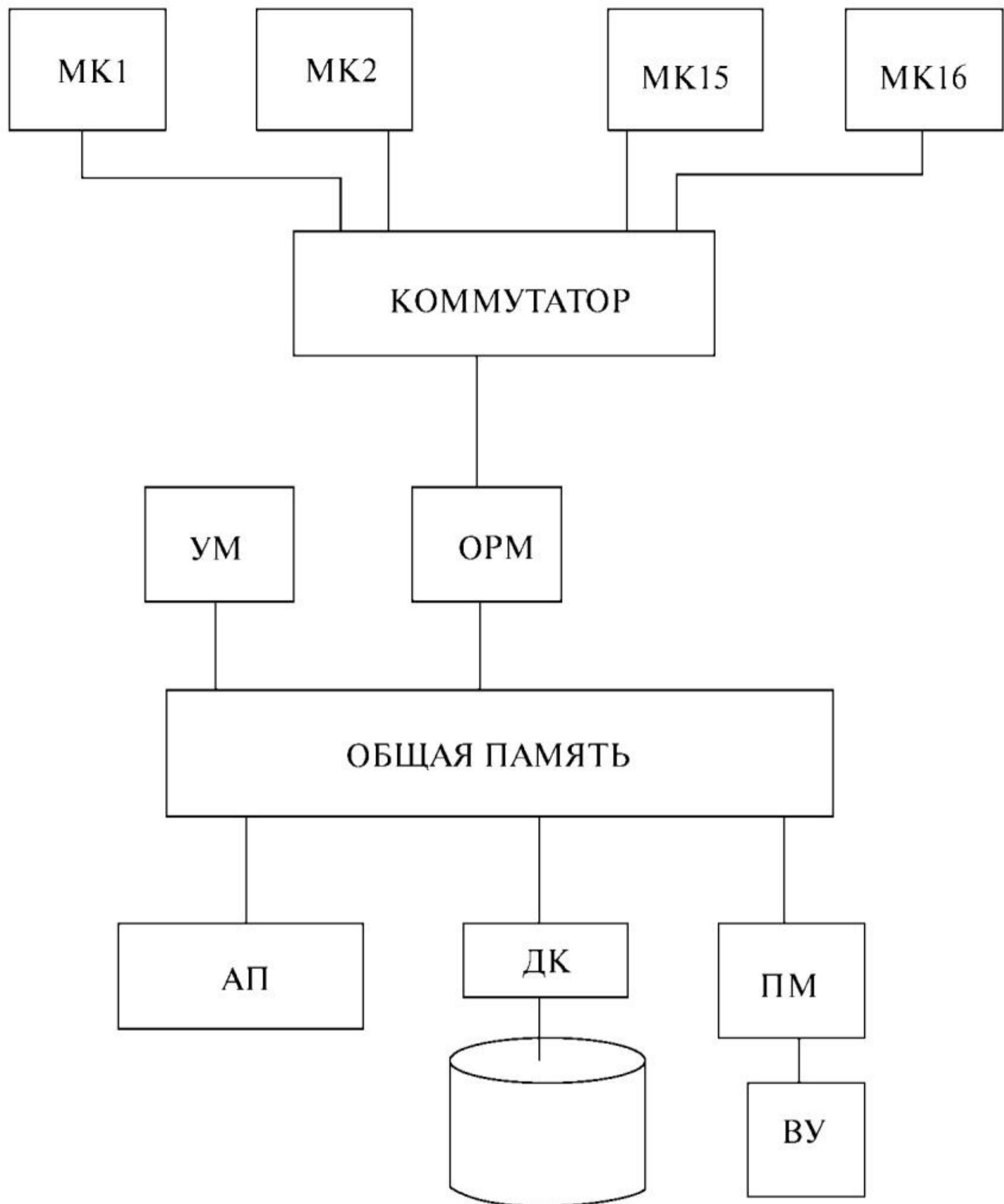


Рис. 3.9. Мультиархитектурный вычислительный комплекс

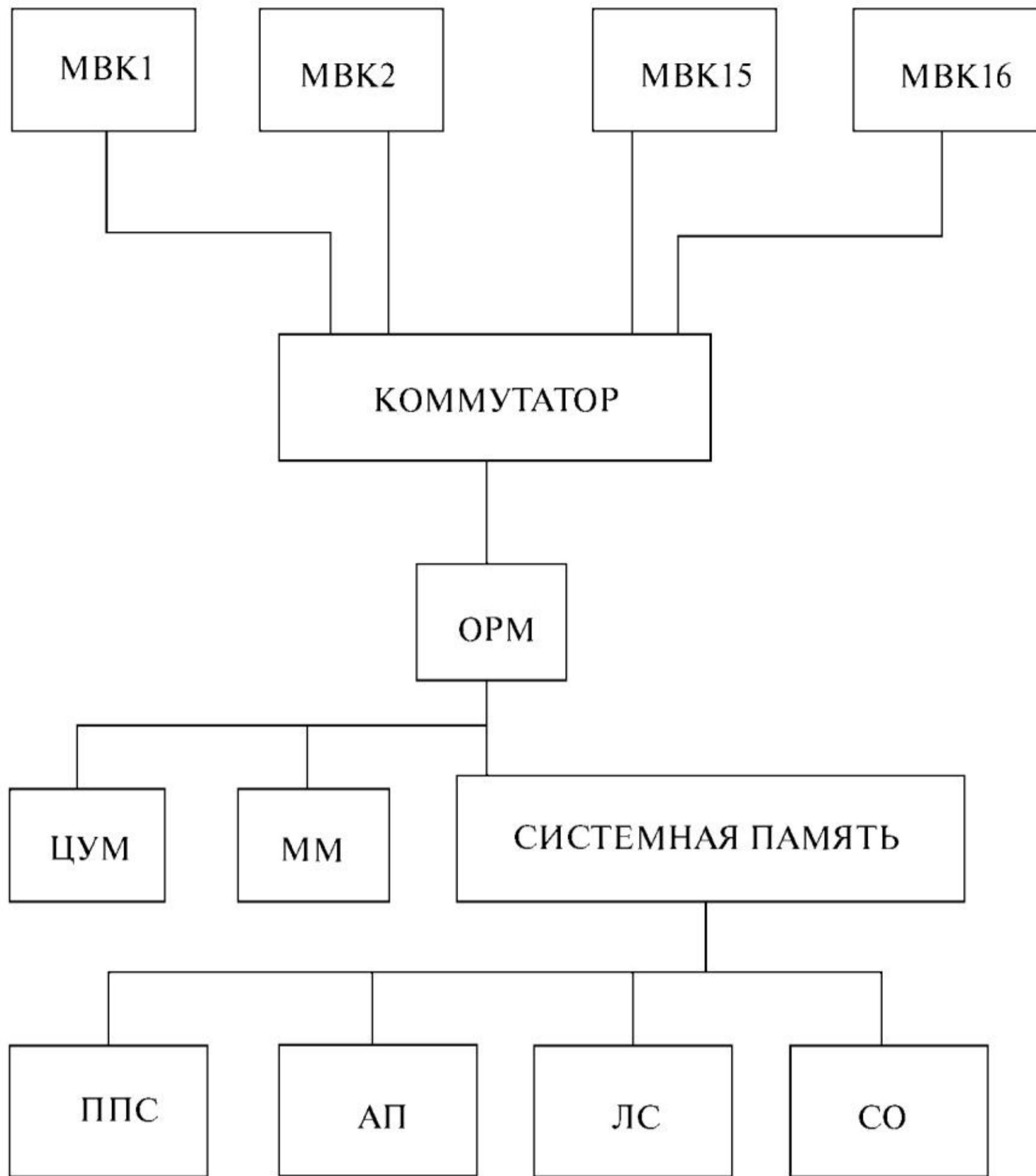


Рис. 3.10. Мультиархитектурная вычислительная система

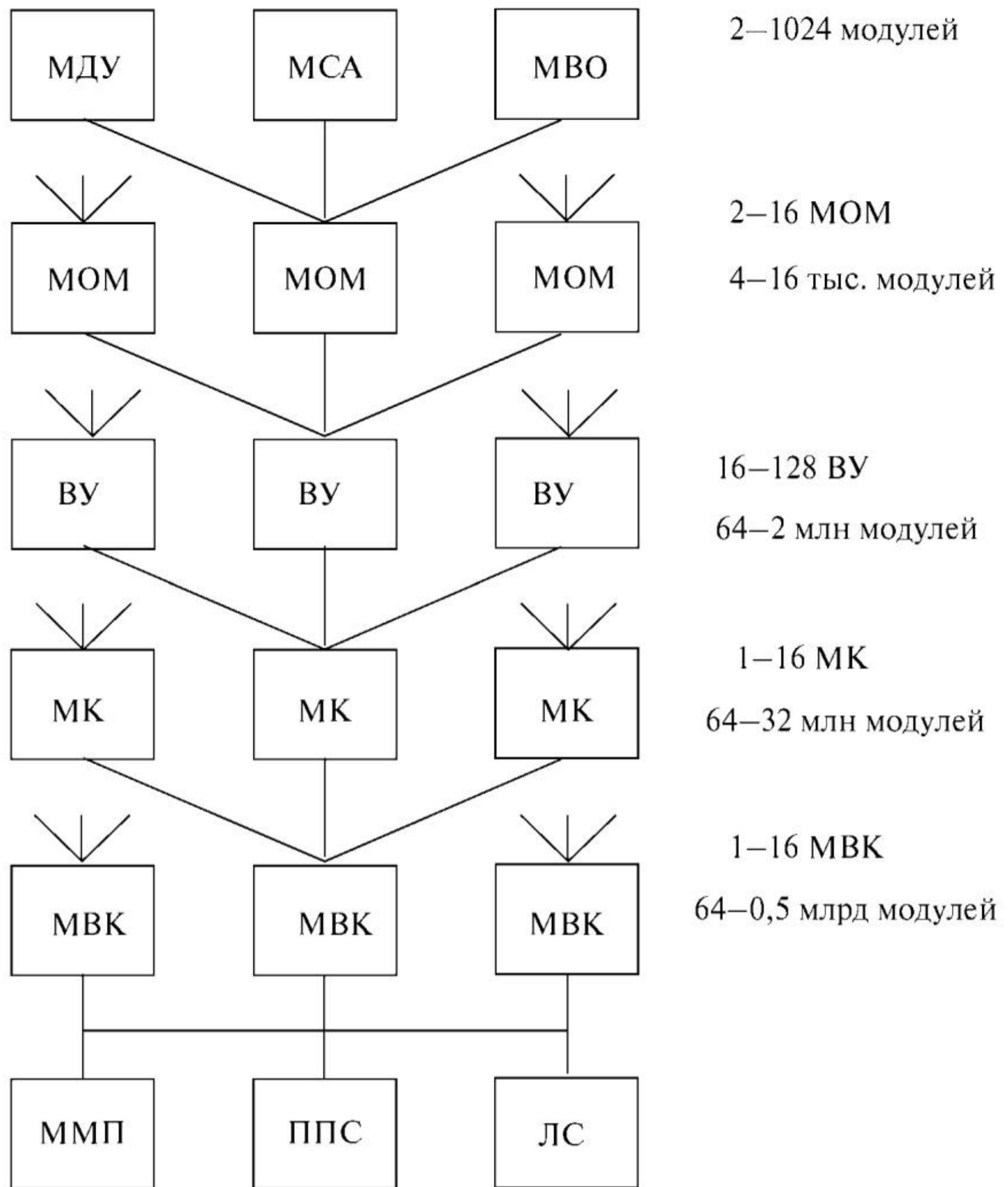


Рис. 3.11. Иерархическая структура системы, сети памяти и сети управления

3.5. Мониторно-моделирующая подсистема, сеть управления и система программирования

Мониторно-моделирующая подсистема выполняет функции подготовки задач, выявления формы параллелизма при интерактивном взаимодействии с прикладным программистом, трансляции программ, фрагментации, загрузки данных и программ и управления выполнением программ.

В мультиархитектурной системе трансляция программ отличается от обычной практики прежде всего из-за наличия процессоров с различной архитектурой. Кроме того, необходимо учитывать тот факт, что в системе могут присутствовать масштабируемые основные машины, отличающиеся наличием разного числа модулей и ориентированных на разную степень параллелизма в рамках одного процессора. На первом этапе трансляции целесообразно привести программу в форму, удобную для ее анализа на следующих этапах. На этапе трансляции необходим анализ формы параллелизма каждого модуля программы и решение вопроса о выборе типа процессора для его выполнения.

На этапе проектирования и конфигурирования системы в соответствии с предполагаемым набором задач могут быть выбраны параметры масштабируемой основной машины, такие как число тех или иных специализированных модулей, а также параметры вычислительного узла и подсистем на более высоких уровнях.

Очевидно, что наиболее эффективно могут быть использованы преимущества мультиархитектурной системы, если они учитываются на этапе выбора вычислительного метода и алгоритма для каждой части задачи. Для реализации такого подхода в языковые средства должны быть введены инструменты управления со стороны программиста. Кроме того, необходимы удобные интерактивные механизмы для взаимодействия между программными средствами подготовки задачи и программистом. Это особенно важно для больших задач, решаемых в течение длительного времени. Корректировка этих программ может быть осуществлена по мере накопления опыта этой эксплуатации.

В условиях вычислительной системы с большим числом процессорных ядер и сложной иерархией процессорных объединений и памяти

необходимость разбиения программы на отдельные фрагменты имеет несколько причин. Во-первых, фрагменты должны предназначаться процессорам с соответствующей архитектурой и уровнем масштабирования. Во-вторых, фрагменты должны обладать свойством внутренней локальности, т.е. для их выполнения может быть выделена тесно связанная группа аппаратных компонентов, например масштабируемая основная машина или вычислительный узел. В-третьих, фрагментация необходима для реализации режима микромультипрограммирования (multithreading), обеспечивающего загрузку процессора при больших задержках по обращению к памяти.

Проблема распределения программ и данных на локальном уровне пересекается с проблемой фрагментации программ. На более высоких уровнях эта проблема связана, с одной стороны, с организацией макроконвейера из отдельных фрагментов задачи, а с другой – с обеспечением оптимальной загрузки всех ресурсов системы.

После распределения программ и данных в процессе выполнения программ должна производиться загрузка программ и данных. Для этой цели должен быть сформирован унифицированный механизм, состоящий в формировании для каждого фрагмента программы пакета, в который входит собственно оттранслированный текст программы и необходимые для ее выполнения данные. Для запуска программы операционная система выдает соответствующую директиву масштабируемой основной машине.

3.6. Обеспечение мультипрограммного режима

В системе должна быть использована единая виртуальная память. Желательно предоставление программисту нескольких адресных пространств, т.е. наличие сегментации памяти. Для преобразования виртуальных адресов в физические используется страничная организация. Многоуровневая организация памяти должна поддерживаться соответствующей аппаратно-программной подсистемой. В целях повышения эффективности на нижнем уровне в масштабируемом процессоре применяется упрощенная система преобразования адресов за счет переключения крупных блоков памяти. На более высоких уровнях преобразование

адресов выполняется соответствующими аппаратными средствами и программами управляющих машин.

Система прерывания в масштабируемых основных машинах связана только с реакцией на ошибки и обращение к монитору по целому ряду причин – окончание фрагмента, переход на служебную программу в управляющей машине и выдача сообщения в межузловую сеть. Потери на прерывания должны быть сведены к минимуму за счет уменьшения числа прерываний и исключения или минимизации объема запоминаемых данных при прерывании.

Система прерывания в управляющих машинах обеспечивает реакцию на все сигналы как от абонентов своего уровня, так и от абонентов более высокого уровня. Поскольку программы управляющих машин входят в состав операционной системы, может быть использован механизм применения команд разрешения прерывания, что исключает сохранение контента программы программой прерывания.

Поскольку все взаимодействие между фрагментами программ выполняется под управлением программ управляющей машины, то синхронизация процессов фактически выполняется на программном уровне в управляющей машине. Эти программы содержат информацию о запуске фрагментов и планировании их выполнения. Одним из элементов такого планирования является организация макроконвейера из программ-фрагментов.

Размеры сети управляющих машин соответствуют масштабу вычислительной системы. При сокращении этих масштабов возможно сокращение числа уровней вычислительной системы и сети управляющих машин, при этом должно происходить соответствующее перераспределение указанных функций.

Архитектура и система команд управляющих машин всех уровней должна быть идентичной, а программное обеспечение всей совокупности управляющих машин является распределенной операционной системой, при этом управляющие машины разных уровней выполняют программы соответствующей части операционной системы.

ГЛАВА 4.

АРХИТЕКТУРА И СИСТЕМА КОМАНД МАСШТАБИРУЕМОЙ ОСНОВНОЙ МАШИНЫ

4.1. Масштабируемая основная машина

При построении масштабируемых процессоров целесообразно использовать крупномодульный принцип благодаря чему, во-первых, упрощается масштабирование внутри процессора и, во-вторых, появляется возможность использовать некоторые модули в различных процессорах. Аналогично могут быть построены процессоры управляющих машин и другие функционально-специализированные процессоры.

Модули, из которых может быть скомпонован любой масштабируемый процессор, являются минимальными структурами для построения подсистемы. Их архитектура с учетом выбора некоторых параметров инвариантна по отношению к реализации и технологии, что обеспечивает сохранение совместимости при изменении технологии. В свою очередь, модули состоят из IP-блоков, что облегчает их проектирование.

В целях обеспечения масштабируемости и специализации процессоров, а также для реализации принципа близкодействия при построении процессоров необходимо предусмотреть наличие следующих особенностей масштабируемых процессоров:

- построение цепочек из большого числа векторных конвейерных функциональных устройств;
- построение нескольких параллельных цепочек векторных конвейерных функциональных устройств;
- выполнение группы последовательных и связанных между собой скалярных операций на цепочке скалярных конвейерных функциональных устройств;
- возможность включения в состав процессоров специализированных функциональных устройств;

- включение в состав машины оперативной памяти, связанной со всеми устройствами машины;
- включение в состав модулей локальной памяти;
- обеспечение средств построения вычислительного узла на основе объединения основных машин.

В соответствии с вышеизложенными принципами целесообразно построение следующего набора модулей и соответствующее распределение функций между ними.

Модуль диспетчерского управления (МДУ) предназначен для выполнения основной управляющей части программы, в том числе для выполнения операций для реализации функций операционной системы, управления обменом между оперативной памятью машины и другими модулями и управления выполнением тех частей программы, которые предназначены для других модулей.

Модуль скалярно-адресный (МСА) предназначен для выполнения команд основной вычислительной части программы, в том числе команд обработки скалярных operandов и адресов. Команды для модуля МСА поступают из модуля МДУ в виде одной или набора групповых мультикоманд для группы модулей МСА.

Модуль векторной обработки (МВО) предназначен для выполнения группы векторных команд, объединенных в виде векторной мультикоманды, которая выдается в цепочку модулей МВО из модуля МСА. Эта группа векторных команд может быть предназначена для выполнения цепочки связанных между собой векторных операций или для выполнения самостоятельных векторных операций в каждом модуле МВО.

Модуль скалярной обработки (МСО) предназначен для выполнения группы связанных скалярных операций, объединенных в виде скалярной мультикоманды, которая выдается в модуль МСО из модуля МСА. Эта группа операций включает последовательность связанных между собой операций, выполняемых на конвейерных функциональных устройствах. Структура модуля МСО обеспечивает одновременное выполнение нескольких таких групп операций в конвейерном режиме.

Модуль специализированный предназначен для выполнения узко-специализированных функций. В системе возможно наличие нескольких типов таких модулей.

В состав машины входит оперативная память, которая, с одной стороны, связана с помощью адаптера, подключенного к обменно-редак-

тирующей машине, с общей памятью вычислительного узла, а с другой – с буферными памятыми всех модулей процессора. Обмен между оперативной памятью основной машины и общей памятью вычислительного узла осуществляется под управлением обменно-редактирующей машины вычислительного узла параллельно с работой основной машины.

В состав машины также входит устройство управления буфером директив. Директивы, поступающие в буфер из управляющей машины вычислительного узла, предназначены для запуска или остановки программ в модуле МДУ.

Для объединения компонентов основной машины используются внутренние сети и средства межсоединений.

Распределение функций в сети памяти основано на иерархической природе сети памяти; наличие на каждом уровне иерархии общей памяти и обменно-редактирующей машины приводит к тому, что загрузка оперативной памяти основных машин и общей памяти на уровне вычислительного узла, мультикомпьютера и мультиархитектурного вычислительного комплекса выполняется под управлением управляющих машин программами обменно-редактирующих машин. При этом, в частности, программы основных машин работают параллельно с указанной загрузкой, а в системе команд основных машин исключены команды пересылки данных из общей памяти в оперативную память.

4.2. Состав и топология масштабируемой основной машины

Архитектура масштабируемой основной машины инвариантна по отношению к аппаратной реализации. Однако следует рассмотреть возможные варианты в зависимости от возможностей технологии и от потребностей решаемых задач. С точки зрения конструкции и топологии имеются следующие варианты реализации основной машины.

1. Основная машина состоит из нескольких БИС. Данный мультичипный вариант возможен при использовании устаревшей технологии или для макетирования, например, на основе ПЛИС. В исключительных

случаях машина с очень большим числом моделей может размещаться на нескольких БИС.

2. Основная машина размещается на одной БИС. Диапазон числа модулей – от 2 до 1000. Максимальное число модулей зависит от уровня технологии. Архитектура машины не должна ограничивать число модулей.

Возможны следующие варианты архитектуры масштабируемых машин:

- скалярная машина;
- мультискалярная машина;
- векторная машина;
- мультивекторная машина;
- мультиархитектурная машина.

3. На одной БИС размещается несколько основных машин. Такая структура возможна при объединении простых скалярных и векторных машин, состоящих из небольшого числа модулей. Если такая БИС содержит часть основных машин узла, то в ее состав должны быть включены такие элементы вычислительного узла, как сеть памяти узла и сеть управления. Эта структура может быть названа подузлом.

4. На одной БИС или на другой единой конструкции размещается вычислительный узел. Данная структура предполагает объединение определенного числа основных машин, обменно-редактирующей, управляющей и сетевой машины узла. Что касается общей памяти вычислительного узла, то в зависимости от параметров и возможностей технологии общая память может размещаться на той же БИС или на других БИС. В последнем случае необходимы соответствующие адAPTERЫ. Для реализации данного варианта потребуется, по-видимому, создание новых технологических и конструктивных решений.

4.3. Организация памяти

Основная машина имеет следующие уровни памяти:

- оперативная память основной машины;
- буферная память модулей;
- регистровая память модулей;
- входные и выходные регистры модулей МВО.

Общая память вычислительного узла связана с оперативной памятью каждой основной машины с помощью обменно-редактирующей машины.

Управляющая машина вычислительного узла осуществляет управление загрузкой программ и данных в оперативную память основной машины путем выдачи соответствующих директив в обменно-редактирующую машину.

4.4. Директивы

Директивы принимаются в буфер директив из управляющей машины. Они предназначены для выполнения таких функций операционной системы, как установка управляющих регистров – регистров преобразования адресов, регистра маски прерывания, регистра реального времени и т.п., а также запуска программы основной машины.

Основная машина при выполнении ряда команд в модуле МДУ, а также при обнаружении специальных ситуаций формирует и выдает в управляющую машину ответную директиву, поступающую в буфер директив управляющей машины.

Формат директив

КД	НП	А	Д
8	16	8	32

КД – код директивы;

НП – номер программы;

А – адрес;

Д – данные.

В зависимости от типа директивы содержимое полей А и Д имеет различное назначение.

Директивы, выдаваемые из управляющей машины, имеют следующие коды:

- 0 – пуск программы основной машины;
- 1 – загрузка регистров состояния;
- 2 – загрузка указателя массива данных;
- 3 – загрузка регистров преобразования адресов;
- 4 – загрузка регистра прерывания.

Директивы, выдаваемые из основной машины, имеют следующие коды:

- 0 – завершение программы основной машины;
- 1 – обращение к монитору со стороны программы основной машины;
- 2 – ошибка при выполнении программы основной машины.

4.5. Функциональные устройства

В состав модулей входит тот или иной набор функциональных устройств, необходимых для выполнения команд данного модуля.

Унификация устройств необходима для обеспечения крупноблочного проектирования. К числу основных функциональных устройств относятся:

- устройство выполнения логических операций;
- устройство сдвига;
- устройство вычисления числа единиц и номера старшей единицы;
- устройство сложения-вычитания с фиксированной запятой;
- устройство умножения с фиксированной запятой;
- устройство сложения-вычитания с плавающей запятой;
- устройство умножения с плавающей запятой;
- устройство деления с плавающей запятой;
- устройство вычисления квадратного корня с плавающей запятой.

Устройства построены на основе синхронных конвейерных схем и могут быть использованы для реализации как скалярных, так и векторных операций.

4.6. Система команд масштабируемого процессора

Система команд любого масштабируемого процессора является совокупностью систем команд входящих в него модулей. Эти системы команд модулей по кодировке операций не зависят друг от друга. Однако в целях возможности построения одномодульного унипроцессора, совместимого с одним из масштабируемых процессоров, коды операций для различных модулей не должны пересекаться.

4.6.1. Форматы данных

Основными форматами данных являются:

- 64-разрядные числа с плавающей запятой;
- 64-разрядные числа с фиксированной запятой;
- 64-разрядные слова;

- 32-разрядные адреса (целые числа).

Набор форматов данных в разных модулях различен.

В состав модулей входят по одному или по два набора следующих адресуемых регистров:

- 256 адресных регистров;
- 256 скалярных регистров или регистров слов;
- 256 векторных регистров. Каждый вектор может содержать до 256 64-разрядных слов или скалярных чисел.

В состав модулей входит адресуемая буферная память с максимальной длиной адреса 32 разряда (4 триллиона слов).

Параметры объема регистровой и буферной памяти при реализации могут иметь меньшую величину.

4.6.2. Форматы команд

КОП	i	j	k
8	8	8	8

Команды имеют 32-разрядный формат, при этом старшие 8 разрядов отведены для кода операции, три 8-разрядных поля i, j, k служат для адресации регистров. В некоторых командах указанные поля объединяются, а код операции расширяется за счет поля i.

4.7. Система команд модуля диспетчерского управления

4.7.1. Модуль диспетчерского управления

Модуль диспетчерского управления (МДУ) принимает директивы от мониторной машины, которая предварительно помещает их в буфер директив. Данный буфер работает по алгоритму простой очереди (FIFO). В результате считывания директивы из буфера выполняется запуск программы, загруженной в оперативную память процессора или в буферную память модуля.

Основные функции модуля состоят в активизации программы процессора, подкачки его оперативной памяти и буферной памяти модулей (программ и данных), запуске программ в модулях МСА, приеме признаков результата из МСА, выдаче результатов в промежуточную память процессора и в оперативную память вычислительного комплекса, синхронизации работы МДУ и МСА.

Этот модуль является единственным для процессора.

В состав модуля МДУ входят набор функциональных устройств, входной и выходной коммутаторы, 256 скалярных 64-разрядных регистров, 256 адресных 32-разрядных регистров, буферная память, буфер директив со схемами приема и выдачи директив, регистр команды и схемы управления, регистр выдачи команд в МСА и схемы приема признаков результата из МСА, контроллеры управления обменом для промежуточной и оперативной памяти.

Для управления обменом между общей памятью вычислительного узла и оперативной памятью основной машины в состав вычислительного узла входит обменно-редактирующая машина. Эта машина осуществляет преобразование математических адресов в физические адреса общей памяти и управляет обменом, при этом в программе должны быть указаны физические адреса оперативной памяти основной машины. При обмене массивом задается положение массива в оперативной памяти, при этом подразумевается, что последовательные элементы массива находятся в соседних ячейках памяти. При адресации элементов массива в общей памяти возможны различные варианты. К ним относятся вычисление адреса следующего элемента путем сложения адреса предыдущего элемента с фиксированной величиной смещения или с вычисляемой величиной смещения. Возможна также работа по списку адресов.

В модуле МДУ выполняются следующие группы операций:

- обработка скалярных и адресных operandов;
- обмен между адресуемыми регистрами и буферной памятью модуля;
- управление обменом между оперативной памятью основной машины и буферными памятыми всех ее модулей;
- управление обменом между оперативной памятью основной машины и буферной памятью вычислительного модуля;
- редактирование данных во время обмена с оперативной памятью основной машины (функции команд «сборка-разборка»);

- безусловные и условные переходы;
- мониторные операции;
- команды запуска программ в модуле скалярных и адресных операций.

В состав модуля МДУ также входят вспомогательные устройства – контроллер обмена с оперативной памятью и устройство приема и выдачи директив.

4.7.2. Мониторные команды

КОП			
8	8	8	8

Основные мониторные операции:

- 00 – отсутствие операции;
- 01 – выход из программы по ошибке;
- 02 – нормальное завершение программы;
- 03 – обращение к монитору;
- 06 – установка маски прерывания;
- 07 – установка регистра реального времени;
- 08 – загрузка регистров преобразования адресов.

4.7.3. Выдача команд в модуль МСА

КОП	АБП	<длина>	АМ
8	8	8	8

По адресу <АБП> в адресном регистре содержится адрес слова в буферной памяти, в котором находятся первые две 32-разрядные команды. В поле <длина> указывается число команд, пересылаемых в МСА. В поле АМ указывается номер модуля МСА, для которого предназначены команды.

- 0A – выдача группы команд в модули скалярно-адресной обработки.

4.7.4. Команды загрузки непосредственных данных

КОП	А	НД
8	8	16

- 04 – загрузка содержимого поляjk в младшие разряды регистра Ai.

4.7.5. Команды передачи управления

КОП	//////////	АП
8	8	16

КОП	//////////	//////////	<АП>
8	8	8	8

Переходы осуществляются в пределах программного пакета, размещаемого в буферной памяти программ модуля МДУ. В поле АП указывается адрес перехода, или в поле <АБП> содержится адрес адресного регистра, содержащего адрес перехода. Имеется полный набор операций условного и безусловного перехода.

Основными операциями являются:

- 10 – безусловная передача управления по прямому адресу;
- 11 – безусловная передача управления по косвенному адресу;
- 12 – безусловная передача управления по прямому адресу с возвратом;
- 13 – безусловная передача управления по косвенному адресу с возвратом;
- 18 – условная передача управления по нулевому коду в нулевом адресном регистре;
- 19 – условная передача управления по ненулевому коду в нулевом адресном регистре;
- 1A – условная передача управления по положительному коду в нулевом адресном регистре;
- 1B – условная передача управления по отрицательному коду в нулевом адресном регистре;
- 1C – условная передача управления по нулевому коду в нулевом скалярном регистре;
- 1D – условная передача управления по ненулевому коду в нулевом скалярном регистре;
- 1E – условная передача управления по положительному коду в нулевом скалярном регистре;

- 1F – условная передача управления по отрицательному коду в нулевом скалярном регистре.

4.7.6. Команды обмена

4.7.6.1. Обмен между оперативной памятью масштабируемого процессора и буферными памятыми модулей

КОП	<НАБП>	<НАОП>	//////////
8	8	8	8

Выполняется запись данных в буферную память модуля из оперативной памяти процессора или считывание данных из буферной памяти и запись их в оперативную память.

Начальный адрес массива в буферной памяти содержится в адресном регистре, заданном в поле <НАБП>, вместе с адресом модуля АМ.

//////////	АМ	НАБП
8	24	32

Начальный адрес массива в оперативной памяти содержится в адресном регистре, заданном в поле <НАОП>, вместе с величиной смещения.

смещение	НАОП
32	32

Адрес первого слова вычисляется как сумма адреса в регистре <НАОП> и смещения.

Основными операциями являются:

- 20 – запись в буферную память модуля МДУ из оперативной памяти процессора;
- 21 – считывание из буферной памяти модуля МДУ и запись в оперативную память процессора;
- 22 – запись в буферную память модуля МСА из оперативной памяти процессора;

- 23 – считывание из буферной памяти модуля МСА и запись в оперативную память процессора;
- 24 – запись в буферную память модуля МВО из оперативной памяти процессора;
- 25 – считывание из буферной памяти модуля МВО и запись в оперативную память процессора;
- 26 – запись в буферную память модуля МСО из оперативной памяти процессора;
- 27 – считывание из буферной памяти модуля МСО и запись в оперативную память процессора.

4.7.6.2. Обмен между скалярным регистром и буферной памятью

КОП	ACP	АБП
-----	-----	-----

8 8 16

КОП	ACP	//////////	<АБП>
-----	-----	------------	-------

8 8 8 8

В командах записи или считывания скалярного операнда адрес скалярного операнда указан в поле АСР, а адрес в БП указан либо в поле АБП (только при адресации младшей части памяти объемом 64К слов), либо в младших разрядах скалярного регистра с адресом <АБП>.

Основными операциями являются:

- 40 – запись в регистр скалярного операнда из буферной памяти по прямому адресу;
- 41 – считывание из регистра скалярного операнда и запись его в буферную память по прямому адресу;
- 42 – запись в регистр скалярного операнда из буферной памяти по косвенному адресу;
- 43 – считывание из регистра скалярного операнда и запись его в буферную память по косвенному адресу.

4.7.6.3. Обмен между адресным регистром и буферной памятью

КОП	AAP	АБП
-----	-----	-----

8 8 16

КОП	ААР	//////////	<АБП>
8	8	8	8

В командах считывания или записи адресного операнда адрес адресного операнда указан в поле ААР, а адрес в БП указан либо в поле АБП (только при адресации младшей части памяти объемом 64К слов), либо в младших разрядах скалярного регистра с адресом <АБП>.

Основными операциями являются:

- 44 – запись в регистр адресного операнда из буферной памяти по прямому адресу;
- 45 – считывание из регистра адресного операнда и запись его в буферную память по прямому адресу;
- 46 – запись в регистр адресного операнда из буферной памяти по косвенному адресу;
- 47 – считывание из регистра адресного операнда и запись его в буферную память по косвенному адресу.

4.7.6.4. Групповой обмен для скалярных operandов

КОП	НАСР	<сч-к>	<НАБП>
8	8	8	8

КОП	<НАСР>	<сч-к>	<НАБП>
8	8	8	8

Данные команды задают операции считывания или записи группы слов, адреса скалярных регистров которых определяются начальным адресом в поле НАСР или младшими разрядами в скалярном регистре <НАСР>. Число передаваемых слов указано в младших разрядах скалярного регистра <сч-к>. Начальный адрес массива слов в БП задан содержимым младших разрядов скалярного регистра с адресом <НАБП>.

Основными операциями являются:

- 48 – запись группы слов в регистры скалярных operandов из буферной памяти по прямому адресу;
- 49 – считывание группы слов из регистров скалярных operandов и запись их в буферную память по прямому адресу;

- 4A – запись группы слов в регистры скалярных operandов из буферной памяти по косвенному адресу;
- 4B – считывание группы слов из регистров скалярных operandов и запись их в буферную память по косвенному адресу.

4.7.6.5. Групповой обмен для адресных operandов

КОП	НААР	<сч-к>	<НАБП>
8	8	8	8

КОП	<НААР>	<сч-к>	<НАБП>
8	8	8	8

Данные команды задают операции считывания или записи группы слов, адреса адресных регистров которых определяются начальным адресом в поле НААР или младшими разрядами в адресном регистре <НААР>. Число передаваемых слов указано в младших разрядах скалярного регистра <сч-к>. Начальный адрес массива слов в буферной памяти задан содержимым младших разрядов скалярного регистра с адресом <НАБП>.

Основными операциями являются:

- 4C – запись группы слов в регистры адресных operandов из буферной памяти по прямому адресу;
- 4D – считывание группы слов из регистров адресных operandов и запись их в буферную память по прямому адресу;
- 4E – запись группы слов в регистры адресных operandов из буферной памяти по косвенному адресу;
- 4F – считывание группы слов из регистров адресных operandов и запись их в буферную память по косвенному адресу.

4.7.7. Команды обработки

4.7.7.1. Скалярные операции

КОП	АР	A1	A2
8	8	8	8

Команды обработки задают выполнение скалярных операций с плавающей и фиксированной запятой.

Операнды, адреса которых задаются в полях A1 и A2, находятся в оперативных скалярных регистрах.

Результат операции, адрес которого задается полем AP, помещается в оперативный скалярный регистр.

Набор кодов операций (КОП) соответствует набору функциональных устройств и выполнению всех необходимых разновидностей операций.

Основными операциями являются:

- 80 – логическое умножение (И);
- 81 – логическое сложение (ИЛИ);
- 82 – поразрядное сложение по модулю 2 (исключающее ИЛИ);
- 83 – логическое равенство;
- 84 – сборка по маске;
- 85 – разборка по маске;
- 88 – сдвиг влево операнда по адресу A1 и запись результата по адресу A1;
- 89 – сдвиг вправо операнда по адресу A1 и запись результата по адресу A1;
- 8A – сдвиг влево операнда по адресу A1 и запись результата по адресу регистра AP;
- 8B – сдвиг вправо операнда по адресу A1 и запись результата по адресу регистра AP;
- 90 – целочисленное сложение скалярных operandов;
- 91 – целочисленное вычитание скалярных operandов;
- 94 –
- 95 –
- 96 –
- 97 –
- 98 – вычисление числа единиц в коде;
- 99 – вычисление числа нулей до первой единицы в коде;
- 9C – установка непосредственных данных из младших 16 разрядов команды в младшие 16 разрядов слова по адресу AP.

4.7.7.2. Адресные операции

КОП	AP	A1	A2
8	8	8	8

Команды обработки задают выполнение адресных операций целочисленного сложения и умножения.

Операнды, адреса которых задаются в полях A1 и A2, находятся в оперативных адресных регистрах.

Результат операции, адрес которого задается полем AP, помещается в оперативный адресный регистр.

Основными операциями являются:

- 9A – целочисленное сложение;
- 9B – целочисленное вычитание;
- 9E – целочисленное умножение.

4.8. Система команд модуля скалярно-адресного

4.8.1. Модуль скалярно-адресный

Модуль скалярно-адресный (МСА) принимает из МДУ команду запуска программы, которая должна быть до этого загружена в его буферной памяти.

Основные функции модуля состоят в выполнении той части программы, в которой имеются только команды обработки скалярных operandов. В модуле выполняются выборка из буферной памяти векторных мультикоманд и выдача их для выполнения цепочку модулей МВО. Число команд в мультикоманде равно числу модулей МВО, подключенных к данному модулю МСА. В модуле также выполняются выборка из памяти скалярных мультикоманд и выдача их для выполнения в цепочку модулей МСО. Число команд в скалярной мультикоманде равно или меньше числа функциональных устройств в цепочке модулей МСО.

В состав модуля МСА входят набор функциональных устройств, входной и выходной коммутаторы, 256 скалярных 64-разрядных регистров, 256 адресных 32-разрядных регистров, буферная память, буфер команд запуска программ, регистр команды и схемы управления, регистры выдачи данных в соседний (первый в цепочке) МВО, регистр выдачи мультикоманд.

В МСА выполняются следующие группы операций:

- арифметические и логические скалярные операции;

- обмен между адресуемыми регистрами и буферной памятью модуля;
- выдача мультикоманд;
- безусловные и условные переходы.

4.8.2. Мониторные команды

КОП			
8	8	8	8

Основными операциями являются:

- 008 – отсутствие операции;
- 009 – выход из программы по ошибке;
- 00A – нормальное завершение программы;
- 00B – обращение к монитору;
- 00C – установка маски прерывания.

4.8.3. Команда выдачи векторной мультикоманды

КОП	<длина>		<АБП>
8	8	8	8

Команда предназначена для считывания из буферной памяти модуля МСА векторной мультикоманды и выдачи ее в цепочку модулей МВО. В младших разрядах адресного регистра <АБП> задается начальный адрес размещения мультикоманды в буферной памяти. В поле <длина> указывается адрес адресного регистра, в младших разрядах которого задается число 32-разрядных команд, выдаваемых в модуль МВО. Это число должно быть равно числу модулей МВО, подключенных в цепочке к данному модулю МСА.

- 0C – выдача векторной мультикоманды в цепочку модулей МВО.

КОП	<длина>	НМ	<АБП>
8	8	8	8

Команда предназначена для считывания из буферной памяти модуля МСА векторной мультикоманды, являющейся фактически векторной программой, и выдачи ее в заданный модуль МВО. В младших разря-

дах адресного регистра <АБП> задается начальный адрес размещения мультикоманды в буферной памяти. В поле <длина> указывается адрес адресного регистра, в младших разрядах которого задается число 32-разрядных команд, выдаваемых в модуль МВО. В поле НМ указывается номер модуля, в который поступает векторная программа.

- 0D – выдача векторной программы в заданный модуль МВО.

4.8.4. Команда выдачи скалярной мультикоманды

КОП	<длина>	//////////	<АБП>
8	8	8	8

Команда предназначена для считывания из буферной памяти модуля МСА скалярной мультикоманды и выдачи ее в модуль МСО. В младших разрядах адресного регистра <АБП> задается начальный адрес размещения мультикоманды в буферной памяти. В поле <длина> указывается адрес адресного регистра, в младших разрядах которого задается число 8-разрядных байтов, выдаваемых в модуль МСО.

- 0E – выдача скалярной мультикоманды.

4.8.5. Команды передачи управления

КОП	//////////	АП
8	8	16

Переходы осуществляются в пределах программного пакета, размещаемого в буферной памяти программ модуля МСА. В поле АП указывается адрес перехода. Имеется полный набор операций условного и безусловного перехода.

Основными операциями являются:

- 30 – безусловная передача управления по прямому адресу;
- 31 – безусловная передача управления по косвенному адресу;
- 32 – безусловная передача управления по прямому адресу с возвратом;
- 33 – безусловная передача управления по косвенному адресу с возвратом;
- 38 –

- 39 –
- 3A –
- 3B –
- 3C – условная передача управления по нулевому коду в нулевом скалярном регистре;
- 3D – условная передача управления по ненулевому коду в нулевом скалярном регистре;
- 3E – условная передача управления по положительному коду в нулевом скалярном регистре;
- 3F – условная передача управления по отрицательному коду в нулевом скалярном регистре.

4.8.6. Команды обмена

4.8.6.1. Обмен между скалярными регистрами и буферной памятью

КОП	ACP	АБП
8	8	16

КОП	ACP	//////////	<АБП>
8	8	8	8

В команде считывания или записи скалярного операнда адрес скалярного операнда указан в поле АСР, а адрес в буферной памяти указан либо в поле АБП (только при адресации младшей части памяти объемом 64К слов), либо в младших разрядах скалярного регистра с адресом <АБП>.

Основными операциями являются:

- 50 – запись в регистр скалярного операнда из буферной памяти по прямому адресу;
- 51 – считывание из регистра скалярного операнда и запись его в буферную память по прямому адресу;
- 52 – запись в регистр скалярного операнда из буферной памяти по косвенному адресу;
- 53 – считывание из регистра скалярного операнда и запись его в буферную память по косвенному адресу.

4.8.6.2. Групповой обмен для скалярных operandов

КОП	НАСР	<сч-к>	<НАБП>
8	8	8	8

КОП	<НАСР>	<сч-к>	<НАБП>
8	8	8	8

Данные команды задают операции считывания или записи группы слов, адреса скалярных регистров которых определяются начальным адресом в поле НАСР или младшими разрядами в скалярном регистре <НАСР>. Число передаваемых слов указано в младших разрядах скалярного регистра <сч-к>. Начальный адрес массива слов в буферной памяти задан содержимым младших разрядов скалярного регистра с адресом <НАБП>.

Основными операциями являются:

- 58 – запись группы слов в регистры скалярных operandов из буферной памяти по прямому адресу;
- 59 – считывание группы слов из регистров скалярных operandов и запись их в буферную память по прямому адресу;
- 5A – запись группы слов в регистры скалярных operandов из буферной памяти по косвенному адресу;
- 5B – считывание группы слов из регистров скалярных operandов и запись их в буферную память по косвенному адресу.

4.8.7. Команды обработки

КОП	АР	А1	А2
8	8	8	8

Команды обработки задают выполнение скалярных операций с плавающей и фиксированной запятой.

Операнды, адреса которых задаются в полях А1 и А2, находятся в оперативных скалярных регистрах.

Результат операции, адрес которого задается полем АР, помещается в оперативный скалярный регистр.

Набор кодов операций (КОП) соответствует набору функциональных устройств и выполнению всех необходимых разновидностей операций.

Основными операциями являются:

- A0 – логическое умножение (И);
- A1 – логическое сложение (ИЛИ);
- A2 – поразрядное сложение по модулю 2 (исключающее ИЛИ);
- A3 – логическое равенство;
- A4 – сборка по маске;
- A5 – разборка по маске;
- A8 – сдвиг влево операнда по адресу A1 и запись результата по адресу A1;
- A9 – сдвиг вправо операнда по адресу A1 и запись результата по адресу A1;
- AA – сдвиг влево операнда по адресу A1 и запись результата по адресу регистра AP;
- AB – сдвиг вправо операнда по адресу A1 и запись результата по адресу регистра AP;
- B0 – целочисленное сложение скалярных operandов;
- B1 – целочисленное вычитание скалярных operandов;
- B4 – сложение скалярных operandов с плавающей запятой;
- B5 – вычитание скалярных operandов с плавающей запятой;
- B6 – умножение скалярных operandов с плавающей запятой;
- B7 – деление скалярных operandов с плавающей запятой;
- B8 – вычисление квадратного корня с плавающей запятой;
- B9 – вычисление числа единиц в коде;
- BA – вычисление числа нулей до первой единицы в коде;
- BB – установка непосредственных данных из младших 16 разрядов команды в младшие 16 разрядов слова по адресу AP.

4.9. Система команд модуля векторной обработки

4.9.1. Модуль векторной обработки

Модуль векторной обработки предназначен для выполнения векторных операций. Команды для выполнения в модуле МВО поступают

из модуля МСА в составе мультикоманды. Мультикоманда выдается из модуля МСА последовательно, по одной 32-разрядной команде за такт взаимодействия соседних модулей. В модуле МВО команда принимается в буфер мультикоманд, и, если данная команда предназначена для данного модуля МВО, она передается в регистр команды для исполнения. Остальная часть мультикоманды выдается в соседний модуль МВО.

Основным режимом работы модуля МВО является участие в выполнении цепочки векторных операций в рамках вычисления сложной векторной функции. Кроме этого, возможно выполнение небольшой программы внутри одного модуля. Команды для этой программы загружаются из выдаваемой из модуля МСА мультикоманды. Результаты вычислений при выполнении такой программы помещаются в регистрах или в буферной памяти, а также могут выдаваться в соседний модуль МВО. В таком режиме фактически выполняются независимые программы векторной обработки или макроконвейерный режим взаимодействия этих программ.

С состав модуля МВО входят набор функциональных устройств, входной и выходной коммутаторы, 256 скалярных 64-разрядных регистров, 256 векторных регистров, каждый из которых может включать до 256 64-разрядных слов, буферная память, буфер мультикоманд, регистр команды и схемы управления, входные и выходные регистры для приема данных из МСА или соседнего МВО и выдачи данных в соседний МВО, регистр выдачи мультикоманд.

В МВО выполняются следующие группы операций:

- арифметические и логические векторные операции;
- операции обмена между векторными и скалярными регистрами и буферной памятью.

4.9.2. Мониторные команды

КОП			
8	8	8	8

Основными операциями являются:

- 07 – установка регистра длины вектора;
- 0F – установка векторной маски.

4.9.3. Команды обмена

4.9.3.1. Обмен между векторными регистрами и буферной памятью

КОП	АВР	АБП
8	8	16

Команда задает операцию считывания или записи вектора в оперативный регистр по адресу, указанному в поле АВР из буферной памяти, причем начальный адрес первого элемента вектора задается в поле АБП – только при адресации младшей части памяти объемом 64К слов.

Формат варианта этой команды в случае косвенной адресации первого элемента вектора в буферной памяти.

КОП	ACP	//////////	<АБП>
8	8	8	8

В поле <АБП> задается адрес скалярного регистра, в 16 младших разрядах которого указан адрес первого элемента вектора в БП.

Основными операциями являются:

- 64 – запись в регистр векторного операнда из буферной памяти по прямому адресу;
- 65 – считывание из регистра векторного операнда и запись его в буферную память по прямому адресу;
- 66 – запись в регистр векторного операнда из буферной памяти по косвенному адресу;
- 67 – считывание из регистра векторного операнда и запись его в буферную память по косвенному адресу.

4.9.3.2. Обмен между скалярными регистрами и буферной памятью

КОП	ACP	АБП
8	8	16

КОП	ACP	//////////	<АБП>
8	8	8	8

В команде считывания или записи скалярного операнда адрес скалярного операнда указан в поле АСР, а адрес в буферной памяти указан

либо в поле АБП (только при адресации младшей части памяти объемом 64К слов), либо в младших разрядах скалярного регистра с адресом <АБП>.

Основными операциями являются:

- 60 – запись в регистр скалярного операнда из буферной памяти по прямому адресу;
- 61 – считывание из регистра скалярного операнда и запись его в буферную память по прямому адресу;
- 62 – запись в регистр скалярного операнда из буферной памяти по косвенному адресу;
- 63 – считывание из регистра скалярного операнда и запись его в буферную память по косвенному адресу.

4.9.3.3. Групповой обмен для скалярных operandов

КОП	НАСР	<сч-к>	<НАБП>
8	8	8	8
КОП	<НАСР>	<сч-к>	<НАБП>
8	8	8	8

Данные команды задают операции считывания или записи группы слов, адреса скалярных регистров которых определяются начальным адресом в поле НАСР или младшими разрядами в скалярном регистре <НАСР>. Число передаваемых слов указано в младших разрядах скалярного регистра, адрес которого указан в поле <сч-к>. Начальный адрес массива слов в буферной памяти задан содержимым младших разрядов скалярного регистра с адресом, указанным в поле <НАБП>.

Основными операциями являются:

- 68 – запись группы слов в регистры скалярных operandов из буферной памяти по прямому адресу;
- 69 – считывание группы слов из регистров скалярных operandов и запись их в буферную память по прямому адресу;
- 6A – запись группы слов в регистры скалярных operandов из буферной памяти по косвенному адресу;
- 6B – считывание группы слов из регистров скалярных operandов и запись их в буферную память по косвенному адресу.

4.9.4. Команды обработки

КОП	АР	A1	A2
8	8	8	8

Команды обработки задают выполнение векторной операции типа «вектор-скаляр» и «вектор-вектор».

Операнды, адреса которых задаются в полях A1 и A2, могут находиться в оперативных регистрах или в одном из входных регистров.

Результат операции, адрес которого задается полем АР, помещается либо в оперативный регистр, либо в выходной регистр.

Набор кодов операций (КОП) соответствует набору функциональных устройств и выполнению всех необходимых разновидностей операций.

4.9.4.1. Операции «вектор-скаляр»

Основными операциями являются:

- C0 – логическое умножение (И);
- C1 – логическое сложение (ИЛИ);
- C2 – поразрядное сложение по модулю 2 (исключающее ИЛИ);
- C3 – логическое равенство;
- C4 – сборка по маске;
- C5 – разборка по маске;
- C7 – маскирование;
- C8 – сдвиг влево операнда по адресу A1 и запись результата по адресу A1;
- C9 – сдвиг вправо операнда по адресу A1 и запись результата по адресу A1;
- DA – сдвиг влево операнда по адресу A1 и запись результата по адресу регистра АР;
- DB – сдвиг вправо операнда по адресу A1 и запись результата по адресу регистра АР;
- D0 – целочисленное сложение operandов с фиксированной запятой;
- D1 – целочисленное вычитание operandов с фиксированной запятой;
- D4 – сложение векторных operandов с плавающей запятой;
- D5 – вычитание векторных operandов с плавающей запятой;

- D6 – умножение векторных operandов с плавающей запятой;
- D7 – деление векторных operandов с плавающей запятой;
- D8 – вычисление квадратного корня с плавающей запятой;
- D9 – вычисление числа единиц в коде;
- DA – вычисление числа нулей до первой единицы в коде;
- DB – установка непосредственных данных из младших 16 разрядов команды в младшие 16 разрядов слова по адресу AP.

4.9.4.2. Операции «вектор-вектор»

Основными операциями являются:

- E0 – логическое умножение (И);
- E1 – логическое сложение (ИЛИ);
- E2 – поразрядное сложение по модулю 2 (исключающее ИЛИ);
- E3 – логическое равенство;
- E4 – сборка по маске;
- E5 – разборка по маске;
- E7 – маскирование;
- E8 – сдвиг влево операнда по адресу A1 и запись результата по адресу A1;
- E9 – сдвиг вправо операнда по адресу A1 и запись результата по адресу A1;
- EA – сдвиг влево операнда по адресу A1 и запись результата по адресу регистра AP;
- EB – сдвиг вправо операнда по адресу A1 и запись результата по адресу регистра AP;
- F0 – целочисленное сложение векторных operandов;
- F1 – целочисленное вычитание векторных operandов;
- F4 – сложение векторных operandов с плавающей запятой;
- F5 – вычитание векторных operandов с плавающей запятой;
- F6 – умножение векторных operandов с плавающей запятой;
- F7 – деление векторных operandов с плавающей запятой;
- F8 – вычисление квадратного корня с плавающей запятой;
- F9 – вычисление числа единиц в коде;
- FA – вычисление числа нулей до первой единицы в коде;
- FB – установка непосредственных данных из младших 16 разрядов команды в младшие 16 разрядов слова по адресу AP.

4.10. Система команд модуля скалярной обработки

4.10.1. Модуль скалярной обработки

В модуле скалярной обработки выполняются скалярные мультикоманды, предназначенные для арифметических операций и операций обмена между буферной памятью и оперативными регистрами. Эти мультикоманды принимаются из буфера мультикоманд, в который они загружаются из модуля МСА.

Основным режимом при выполнении арифметических операций является выполнение последовательных и связанных между собой команд. Поэтому целесообразно использовать одноадресную систему команд. При этом результат предыдущей операции является первым операндом, а второй операнд задается адресом в команде. Однако следует обеспечить возможность как записи промежуточного результата, так и считывания первого операнда из памяти. Для этого необходимы операции записи и чтения. Для каждого выполнения мультикоманды в каждом модуле МСО необходима подготовка операндов на оперативных регистрах, их число для одного модуля с учетом наличия в модуле двух или более функциональных устройств может составлять 16.

Такой набор регистров необходим для каждой мультикоманды, подготовленной к выполнению и помещенной в специальную буферную память. При выполнении отдельных команд из мультикоманды в модулях МСО необходима информация для их идентификации. Такой информацией является номер текущей мультикоманды. Этот номер помещается в каждой мультикоманде и передается в каждый модуль по мере выполнения отдельных команд. В соответствии с этим в каждом модуле необходима память для хранения содержимого регистров всех текущих мультикоманд, т.е. мультикоманд, размещенных в специальном буфере. Допустим, что максимально в буфере помещается до 256 мультикоманд. Это означает, что в каждом модуле необходимо иметь 256 групп по 16 регистров в группе для каждой команды.

С точки зрения сокращения оборудования желательно, чтобы в модуле было минимальное число функциональных устройств. Однако при такой специализации устройств их полная загруженность может достигаться только при условии совпадения последовательности операций в цепочке

модулей и в программе. Для улучшения степени использования функциональных устройств целесообразно обеспечить возможность выполнения в одном модуле по крайней мере двух последовательных операций, причем последовательность их работы должна определяться программно.

Формат скалярной мультикоманды обработки

КОП	НМК	K1	K2	K3	Ki	KN
8	8	8	8	8	8	8

КОП – код операции;

НМК – номер мультикоманды;

K1 – KN – команды 1 – N – команды для последовательных функциональных устройств в модулях МСО в цепочке. Число команд в скалярной мультикоманде зависит от конфигурации модулей МСО и конфигурации процессора. В модулях, предназначенных для выполнения двух или четырех операций, выполняются соответственно две или четыре соседние команды. Возможно построение модулей, предназначенных и для большего числа операций. В модуле МСО целесообразно иметь набор функциональных устройств для выполнения различных операций, что обеспечит более эффективное выполнение программ.

- 0F – скалярная мультикоманда обработки.

4.10.2. Обмен между оперативными регистрами и буферной памятью

КОП	НМК	АБП
8	8	16

Команда задает операцию считывания или записи в оперативный регистр группы из 16 слов для мультикоманды по номеру, указанному в поле НМК, из буферной памяти, причем начальный адрес первого слова в группе задается в поле АБП – только при адресации младшей части памяти объемом 64К слов.

Формат варианта этой команды в случае косвенной адресации первого слова из группы слов в буферной памяти.

КОП	НМК	<АБП>
-----	-----	-------

8 8 16

В поле <АБП> задается адрес скалярного регистра, в 16 младших разрядах которого указан адрес первого слова из группы слов в БП.

Основными операциями являются:

- 7C – запись в оперативные регистры группы слов для скалярной мультикоманды из буферной памяти по прямому адресу;
- 7D – считывание из оперативных регистров группы слов для скалярной мультикоманды в буферную память по прямому адресу;
- 7E – запись в оперативные регистры группы слов для скалярной мультикоманды из буферной памяти по косвенному адресу;
- 7F – считывание из оперативных регистров группы слов для скалярной мультикоманды в буферную память по косвенному адресу.

4.10.3. Команды скалярной обработки

КОП	АО
-----	----

4 4

Команды обработки задают операцию в поле КОП. В поле АО задается адрес операнда на оперативном регистре данной мультикоманды. В качестве второго операнда для арифметических операций используется результат предыдущей операции.

Основными операциями являются:

- 0 – отсутствие операции;
- 1 – считывание операнда;
- 2 – запись результата;
- 3 –
- 4 – сложение скалярных operandов с плавающей запятой;
- 5 – вычитание скалярных operandов с плавающей запятой;
- 6 – умножение скалярных operandов с плавающей запятой;
- 7 – деление скалярных operandов с плавающей запятой;
- 8 – целочисленное сложение operandов с фиксированной запятой;
- 9 – целочисленное вычитание operandов с фиксированной запятой.

4.11. Сетевая структура масштабируемой основной машины

4.11.1. Внутренние сети вычислительного узла

Сеть памяти вычислительного узла включает его общую память и устройства памяти основных, управляющей и сетевой машин, а также средства передачи данных между общей памятью вычислительного узла и оперативными памятьями указанных машин.

Сеть управления вычислительного узла включает буфер директив управляющей машины, буферы директив основных машин, буфер директив обменно-редактирующей машины и буфер директив сетевой машины.

4.11.2. Внутренние межсоединения основной машины

Для объединения компонентов основной машины используются следующие внутренние сети и средства межсоединений:

- двухразмерная сеть для подключения модулей МДУ, МСА, МВО, МСО к оперативной памяти;
- последовательная схема управления для выдачи группы команд из модуля МДУ в модули МСА;
- последовательная схема управления для выдачи векторной мультикоманды из модуля МСА в модули МВО;
- последовательная схема управления для выдачи скалярной мультикоманды из модуля МСА в модули МСО;
- схемы передачи данных из модуля МСА в первый в цепочке модуль МВО и из предыдущего в цепочке модуля МВО в последующий модуль МВО;
- схемы связи управления буфером директив с оперативной памятью и с модулем МДУ;
- схемы связи оперативной памяти с адаптером для подключения основной машины к обменно-редактирующей машине вычислительного узла.

Все функции, связанные со взаимодействием программ, выполняются в межузловой сети с помощью управляющей и сетевой машины.

Поскольку загрузка оперативной памяти и управление запуском программ реализуются управляющей машиной совместно с обменно-редактирующей машиной, самостоятельной сети для обмена между основными машинами в узле не требуется.

В случае реализации нескольких основных машин на одной БИС в таком подузле должны быть реализованы разветвители указанных внутренних сетей и средств межсоединений.

ГЛАВА 5.

АРХИТЕКТУРА ФУНКЦИОНАЛЬНО-СПЕЦИАЛИЗИРОВАННЫХ МАШИН

5.1. Принципы программирования

Новая парадигма программирования и трансляции прикладных программ для мультиархитектурной вычислительной системы состоит из следующих условий:

- разделение программы на модули обработки с разными формами параллелизма и подготовка соответствующих программ;
- разделение программы на модули обработки, выполняемые на основной машине, и модули управления данными;
- выполнение этих модулей на разных по архитектуре машинах;
- формулирование требований к архитектуре управляющих и обменно-редактирующих машин;
- разделение модулей управления данными на модули для управляющих машин и модули (задания) для обменно-редактирующих машин. Возможен вариант, при котором модули управления данными выполняются в обменно-редактирующих машинах, а в управляющих машинах выполняются только функции операционной системы.

Программное задание поступает в масштабируемую основную машину из управляющей машины вычислительного узла. Кроме собственно программы масштабируемой основной машины, точнее набора модулей с ограниченным набором данных с учетом их локализации, транслятор формирует набор заданий для управляющих машин всех уровней для управления данными, в том числе для преобразования виртуальных адресов в физические, для управления обменом между уровнями памяти и редактированием данных в процессе обмена. Эти задания отражают те части прикладной задачи, которые обеспечивают управление пересыпкой данных.

Возможна иерархия заданий в соответствии с иерархией памяти (уровней системы). На каждом уровне устанавливается предельный раз-

мер физической памяти и свой размер виртуальной памяти. На каждом уровне имеется механизм преобразования адресов, включающий таблицу преобразования.

Модули управления данными готовятся транслятором в виртуальных адресах (до 64 бит). При запуске программы должно выполняться преобразование адресов на всех уровнях, причем на уровне оперативной памяти масштабируемой основной машины для быстрого преобразования адресов должно использоваться специальное размещение в фиксированных по размерам областях физической памяти.

5.2. Операционная система

В большинстве систем на стандартных микропроцессорах операционная система имеется в каждом процессоре.

В мультиархитектурной вычислительной системе операционная система распределена в виде мониторно-моделирующей подсистемы, состоящей из сети управления, управляющих машин всех уровней с их программами. Часть функций операционной системы выполняет обменно-редактирующая машина, однако она выполняет и функции прикладных задач по подготовке данных.

Функции фрагментации программ и подготовки программ для масштабируемой основной машины выполняются операционной системой, однако желательно участие прикладного программиста, например, с помощью команд разрешения прерывания или управления программными модулями.

Структура прикладных программ должна учитывать иерархическую структуру системы. Они должны состоять из функциональных программных модулей: обработки – с указателями распараллеливания и типа распараллеливания, редактирования данных, ввода (подкачки) данных, вывода данных, взаимодействия с операционной системой.

Функциональные модули соответствуют разным уровням в системе, однако число уровней в прикладной программе инвариантно по отношению к размеру системы и поэтому может не соответствовать числу уровней в системе.

5.3. Принципы аппаратной реализации функционально-специализированных машин

Функционально-специализированные машины, входящие в состав системы, существенно отличаются от масштабируемых основных машин прежде всего из-за сокращения числа форматов данных за счет исключения операций с плавающей запятой, а также благодаря более простым схемам выполнения операций и меньшей глубине конвейерных схем.

Для сокращения потерь при прерываниях эти машины не должны содержать большого объема информации, сохраняемой при прерывании. Это может быть достигнуто, с одной стороны, за счет сокращения числа регистров, используемых программой, а с другой стороны, за счет такой организации программ, при которой прерывания разрешены только в тех местах программы, где не требуется сохранения содержимого регистров. Последнее свойство возможно только для программ операционной системы, которые в основном и работают на функционально-специализированных машинах.

При всем разнообразии этих машин при модульном принципе их построения возможно выделение ряда модулей, общих для всех специализированных машин.

К числу индивидуальных модулей относятся дополнительные функциональные устройства, специализированные контроллеры и другие устройства.

Модульный принцип построения позволит объединять на одном кристалле целый набор машин. Конкретная реализация такого набора зависит от целого ряда факторов, связанных как с технологическими аспектами, так и выбором архитектурных решений, зависящих от масштабов системы и места набора машин в системе. Принцип мультиархитектурности, лежащий в основе построения масштабируемых процессоров основных машин, в каком-то смысле может быть распространен и на процессоры функционально-специализированных машин.

5.4. Архитектура и система команд функционально-специализированных машин

Рассмотрим особенности функционального назначения, архитектуры и системы команд процессоров и машин на их основе, входящих в состав мониторно-моделирующей подсистемы.

Архитектура и системы команд управляющей машины, сетевой машины, обменно-редактирующей машины, периферийной и моделирующей машины ориентированы на выполнение функций управления и адресной обработки. Формат данных – 8-, 16-, 32- и 64-разрядные целые числа, адресные поля, байты (символы), поля сообщений. Набор операций обработки состоит из операций сложения, вычитания, умножения и логических операций. Для построения средств взаимодействия между машинами и построения системы прерывания используется механизм директив. Функциональная специализация указанных машин приводит к необходимости наличия соответствующих дополнительных средств в их архитектуре и системе команд.

В основе всех указанных машин может быть использован единый модуль процессора – базовый процессор. Функционально-специализированная машина состоит из базового процессора, оперативной памяти, модуля расширения для процессора, специализированных модулей и адаптера сети памяти.

Описание архитектуры и системы команд функционально-специализированных машин состоит из двух частей – из описания архитектуры и системы команд базового процессора и описания архитектуры и системы команд модуля расширения для процессора и специализированных модулей для каждого типа процессора. Приведенное ниже описание системы команд носит эскизный характер и должно дорабатываться на этапе ОКР.

5.4.1. Архитектура и система команд базового процессора

Базовый процессор является основой для построения функционально-специализированных машин. Индивидуальные функции, необходимые для той или иной машины, реализуются в виде модулей расширения.

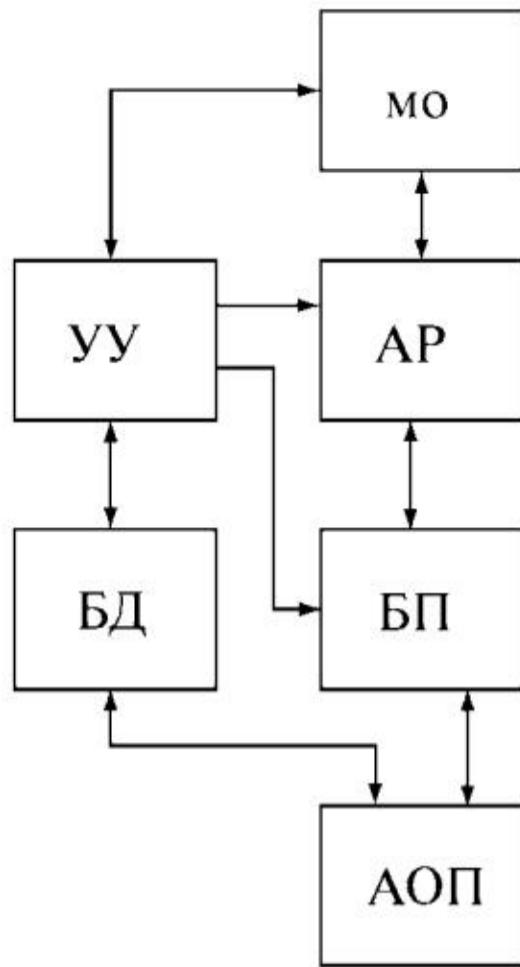


Рис. 5.1. Базовый процессор

Базовый процессор состоит из следующих модулей: модуль обработки данных (МО), устройство управления (УУ), управление буфером директив (БД), адресуемые регистры (АР), буферная память (БП) и адаптер оперативной памяти (АОП) (рис. 5.1).

Форматы данных – байты, двойные байты, полуслова (32 разряда) и слова (64 разряда).

Форматы команд – 32- и 64-разрядные.

Адресация operandов в операциях обработки – трехадресная. Operandы должны быть размещены в адресуемых регистрах. Пересылки между оперативной памятью, буферной памятью и оперативными регистрами осуществляются программно.

В состав модуля обработки данных входит набор функциональных устройств.

5.4.1.1. Буфер директив

Буфер директив хранится в определенном месте памяти и управляется регистром загрузки и регистром разгрузки. Директивы поступают в буфер в порядке обычной очереди. При прерывании процессора очередная директива поступает на выполнение. На каждом уровне системы буфер директив взаимодействует с буферами директив соседних машин

более высокого и низкого уровня. Кроме того, буфер директив используется для обеспечения работы системы прерывания, при этом директивы направляются в собственный буфер машины.

Исключением является буфер директив масштабируемой основной машины, который взаимодействует только с буфером директив управляющей машины вычислительного узла.

Буфер директив управляющей машины вычислительного узла взаимодействует с буфером директив масштабируемой основной машины и с буфером директив управляющей машины мультикомпьютера, а также с буферами директив сетевой, обменно-редактирующей и периферийной машины.

Буфер директив управляющей машины мультикомпьютера взаимодействует с буфером директив управляющей машины вычислительного узла и с буфером директив управляющей машины мультиархитектурного вычислительного комплекса, а также с буферами директив обменно-редактирующей машины и периферийной машины.

Буфер директив управляющей машины мультиархитектурного вычислительного комплекса взаимодействует с буфером директив управляющей машины мультикомпьютера и с буфером директив управляющей машины вычислительной подсистемы, а также с буферами директив обменно-редактирующей машины и периферийной машины.

Формат директив

КД	НМ	НП	АП
8	16	8	32

КД – код директивы;

НМ – номер основной машины в узле, номер узла в мультикомпьютере, номер мультикомпьютера в комплексе или номер комплекса в системе;

НП – номер программы (подканала);

АП – адрес перехода на программу.

Основные виды директив для процессора машины:

- возврат из прерывания;
- переход на программу монитора в управляющей машине вычислительного узла, мультикомпьютера, комплекса или системы;

- переход на прикладную программу в основной машине вычислительного узла;
- переход по окончании работы программы;
- переход по прерыванию программы.

5.4.1.2. Система команд базового процессора

5.4.1.2.1. Мониторные команды

КОП			
8	8	8	8

Основные мониторные операции:

- 00 – отсутствие операции;
- 01 – выход из программы по ошибке;
- 02 – нормальное завершение программы;
- 03 – обращение к монитору;
- 06 – установка маски прерывания;
- 07 – установка регистра реального времени;
- 08 – загрузка регистров преобразования адресов.

5.4.1.2.2. Команды управления взаимодействием

КОП	РД	РНВУ	РНМ
8	8	8	8

КОП – код операции;

РД – адрес регистра для хранения директивы;

РНВУ – номер регистра для хранения номера вычислительного узла;

РНМ – номер регистра для хранения номера машины в узле.

Команды управления взаимодействием предназначены для выдачи директив и управления прерываниями.

К числу операций управления взаимодействием относятся:

- 0A – пуск стандартной программы;
- 0B – пуск программы монитора (выдача директивы для управляющей машины);
- 0C – пуск программы устройства обмена (выдача директивы) для записи-считывания массива данных из памяти в буфер сообщения (для сетевой машины) или пуск программы каналлера (для периферийной машины);
- 0D – переход в режим ожидания (разрешение прерывания).

5.4.1.2.3. Команды передачи управления

КОП	<AM>	РБА	//////////	СМ
8	8	8	8	32

КОП – код операции;

<AM> – адрес регистра хранения модификатора адреса перехода;

РБА – адрес регистра хранения базового адреса;

СМ – смещение адреса перехода.

Адрес перехода формируется при суммировании базового адреса, модификатора адреса перехода, находящегося в регистре <AM>, и смещения.

К числу операций передачи управления относятся:

- 10 – отсутствие операции;
- 11 – безусловный переход;
- 12 – безусловный переход с запоминанием адреса возврата;
- 14 – конец цикла с уменьшением содержимого счетчика;
- 15 – конец цикла с увеличением содержимого счетчика;
- 18 – условный переход по заданному признаку.

5.4.1.2.4. Команды обращения к памяти

Формат команды записи-считывания слова

КОП	<AM>	РБА	РСЧ	СМ
8	8	8	8	32

КОП – код операции;

<AM> – адрес регистра хранения модификатора адреса;

РБА – адрес регистра хранения базового адреса;

РСЧ – регистр счетчика;

СМ – смещение адреса.

Адрес слова формируется при суммировании базового адреса, модификатора адреса перехода, находящегося в регистре <AM>, и смещения.

К числу операций обращения к памяти относятся:

- 30 – запись слова из оперативной в буферную память;
- 31 – считывание слова из буферной в оперативную память;
- 32 – запись группы слов из оперативной в буферную память;
- 33 – считывание группы слов из буферной в оперативную память;
- 34 – запись адреса возврата;

- 35 – переход по адресу возврата;
- 36 – выполнить;
- 37 – синхронизационное считывание;
- 38 – загрузка базовых регистров.

5.4.1.2.5. Команды пересылки данных

КОП	НААР	НАБП	РСч
8	8	8	8

КОП – код операции;

НААР – начальный адрес адресуемых регистров;

НАБП – начальный адрес буферной памяти;

РСч – регистр счетчика.

В поле НААР может указываться адрес адресуемого регистра, в котором хранится прямой адрес регистра или косвенный адрес, т.е. адрес адресуемого регистра, в котором хранится начальный адрес.

В поле НАБП указывается адрес адресуемого регистра, в котором хранится начальный адрес.

В поле РСч может указываться адрес адресуемого регистра, в котором хранится число пересылаемых слов или косвенный адрес, т.е. адрес адресуемого регистра, в котором хранится число пересылаемых слов.

К числу операций пересылки данных относятся:

- 40 – пересылка группы слов из буферной памяти в адресуемый регистр с прямой адресацией НААР и РСч;
- 41 – пересылка группы слов из буферной памяти в адресуемый регистр с косвенной адресацией НААР и РСч;
- 42 – пересылка группы слов из адресуемого регистра в буферную память с прямой адресацией НААР и РСч;
- 43 – пересылка группы слов из адресуемого регистра в буферную память с косвенной адресацией НААР и РСч.

5.4.1.2.6. Команды загрузки непосредственных данных

КОП	АР			НД
8	8	8	8	32

КОП – код операции;

АР – адрес регистра;

НД – непосредственные данные.

- 81 – загрузка 32-разрядного поля непосредственных данных в младшую половину слова по адресу, хранящемуся в регистре AP;
- 82 – загрузка 32-разрядного поля непосредственных данных в старшую половину слова по адресу, хранящемуся в регистре AP;

КОП	AP	НД
8	8	16

- 83 – загрузка 16-разрядного поля непосредственных данных по адресу байта, хранящемуся в регистре AP;

КОП	AP	//////////	НД
8	8	8	8

- 84 – загрузка 8-разрядного поля непосредственных данных по адресу байта, хранящемуся в регистре AP.

5.4.1.2.7. Команды обработки

КОП	AP	A1	A2
8	8	8	8

В командах обработки участвуют операнды, находящиеся на одном из 256 адресуемых оперативных регистров.

Операнды, адреса которых задаются в полях A1 и A2, находятся в адресуемых оперативных регистрах.

Результат операции, адрес которого задается полем AP, помещается в адресуемый оперативный регистр.

Обработка 64-разрядных operandов использует 256 64-разрядных регистров.

Обработка 32-разрядных operandов использует 128 младших 64-разрядных регистров.

Обработка 16-разрядных operandов использует 64 младших 64-разрядных регистра.

Обработка 8-разрядных operandов использует 32 младших 64-разрядных регистра.

К операциям обработки относятся:

операции обработки байтов:

- 80 – логическое сложение байтов;
- 81 – логическое умножение байтов;
- 82 – сложение по модулю 2 байтов;
- 83 – сдвиг байтов вправо;
- 84 – сдвиг байтов влево;
- 85 – сборка байтов;
- 86 – разборка байтов;
- 87 – сложение байтов;
- 88 – вычитание байтов;
- 89 – сравнение байтов;
- 8A – умножение байтов;
- 8B – деление байтов;
- 8C – пересылка байтов;
- 8D –
- 8E –
- 8F –

операции обработки двойных байтов:

- 90 – логическое сложение двойных байтов;
- 91 – логическое умножение двойных байтов;
- 92 – сложение по модулю 2 двойных байтов;
- 93 – сдвиг вправо двойных байтов;
- 94 – сдвиг влево двойных байтов;
- 95 – сборка двойных байтов;
- 96 – разборка двойных байтов;
- 97 – сложение двойных байтов;
- 98 – вычитание двойных байтов;
- 99 – сравнение двойных байтов;
- 9A – умножение двойных байтов;
- 9B – деление двойных байтов;
- 9C – пересылка двойных байтов;
- 9D –
- 9E –
- 9F –

операции обработки полуслов:

- A0 – логическое сложение полуслов;
- A1 – логическое умножение полуслов;

- A2 – сложение по модулю 2 полуслов;
- A3 – сдвиг вправо полуслов;
- A4 – сдвиг влево полуслов;
- A5 – сборка полуслов;
- A6 – разборка полуслов;
- A7 – сложение полуслов;
- A8 – вычитание полуслов;
- A9 – сравнение полуслов;
- AA – умножение полуслов;
- AB – деление полуслов;
- AC – пересылка полуслов;
- AD –
- AE –
- AF –

операции обработки слов:

- B0 – логическое сложение слов;
- B1 – логическое умножение слов;
- B2 – сложение по модулю 2 слов;
- B3 – сдвиг вправо слов;
- B4 – сдвиг влево слов;
- B5 – сборка слов;
- B6 – разборка слов;
- B7 – сложение слов;
- B8 – вычитание слов;
- B9 – сравнение слов;
- BA – умножение слов;
- BB – деление слов;
- BC – пересылка слов;
- BD –
- BE –
- BF –

5.4.2. Оперативная память функционально-специализированных машин

Оперативная память функционально-специализированных машин является модулем для построения управляющей, сетевой, обменно-редактирующей и моделирующей машины.

Физическая адресация обеспечивает максимальный объем памяти – 32 Глов.

Адреса в командах обмена виртуальные. Физическая память разделена на фиксированные сегменты. Программа и ее данные должны размещаться в одном сегменте. При запуске программы ей назначается физический сегмент. Загрузку этого сегмента и запуск программы выполняет программа операционной системы более высокого уровня, размещаемая, как правило, в управляющей машине более высокого уровня.

Блок сопряжения с адаптером сети памяти является модулем для соединения машины с адаптером сети памяти того или иного уровня.

5.4.3. Коммутаторы и адаптеры сети памяти

Система коммутации в сети памяти основана на коммутации пакетов и передаче сообщений. В каждом сообщении содержится адрес абонента сети, которому предназначено сообщение, пакет пересылаемых данных и адрес источника сообщения.

На каждом уровне иерархии сети памяти имеются входной и выходной коммутаторы.

Входной коммутатор с помощью адаптера сети памяти подключается к более высокому уровню иерархии сети памяти. Кроме того, возможно подключение дополнительных устройств для данного уровня. К числу таких устройств может относиться дисковая память. К входному коммутатору подключаются все машины и общая память данного уровня. Входной коммутатор обеспечивает обмен устройствами данного уровня как с устройствами более высокого уровня, так и между устройствами данного уровня.

Выходной коммутатор обеспечивает связь общей памяти с абонентами более низкого уровня иерархии. Этот обмен выполняется под управлением обменно-редактирующей машины, при этом данные могут поступать как из общей памяти, так и из оперативной памяти обменно-редактирующей машины. Другие машины осуществляют обмен с абонентами более низкого уровня через общую память или путем передачи директив для обменно-редактирующей машины.

Коммутаторы и подключенные к ним устройства на каждом входе/выходе имеют адаптер сети памяти, в котором формируется и хранится в буферной памяти сообщение для сети памяти.

5.4.4. Архитектура и система команд управляющей машины

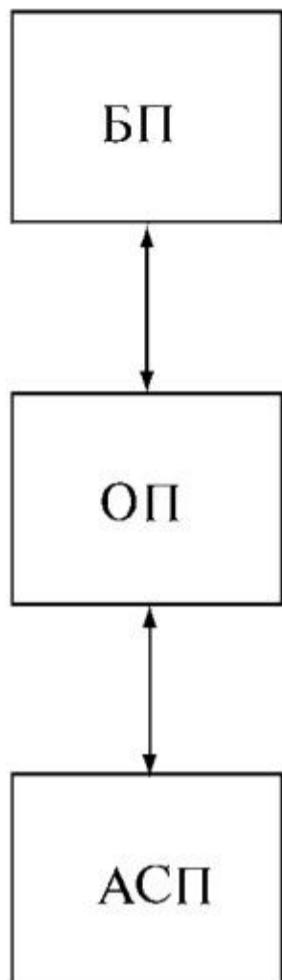


Рис. 5.2. Управляющая машина

Управляющие машины предназначены для построения сети управления и выполнения функций распределенной операционной системы. На каждом уровне системы имеется своя управляющая машина.

Управляющая машина состоит из базового процессора (БП), оперативной памяти (ОП) и адаптера сети памяти (АСП) (рис. 5.2). Расширения системы команд базового процессора нет.

5.4.5. Архитектура и система команд сетевой машины

Сетевые машины предназначены для построения межузловой сети, в которую входит коммутационная структура, объединяющая сетевые машины. Сетевые машины входят в состав каждого вычислительного узла.

Сетевая машина состоит из базового процессора (БП), оперативной памяти (ОП), модуля расширения набора команд для сетевой машины (МР), устройства обмена (УО), буфера директив устройства обмена (БДО), буферной памяти сообщений (БПС), узла коммутатора межузловой сети с адаптером межузловой сети (УК) и адаптера сети памяти (АСП) (рис. 5.3). Фактически сетевая машина состоит из двух процессоров, двух буферов директив и общей оперативной памяти. Одним из процессоров является процессор сетевой машины, основанный на расширении базового процессора. Другой процессор – это узкоспециализированное устройство для управления пересылкой массивов данных между общей буферной памятью сообщений и оперативной памятью и пересылкой сообщений по межузловой сети. Процессор сетевой машины получает задания в виде директив от управляющей машины, формирует задание для устройства обмена, запускает обмен путем выдачи директивы для устройства обмена, принимает директивы от устройства обмена о выполнении обмена и выдает сообщение в управляющую машину о выполнении задания.

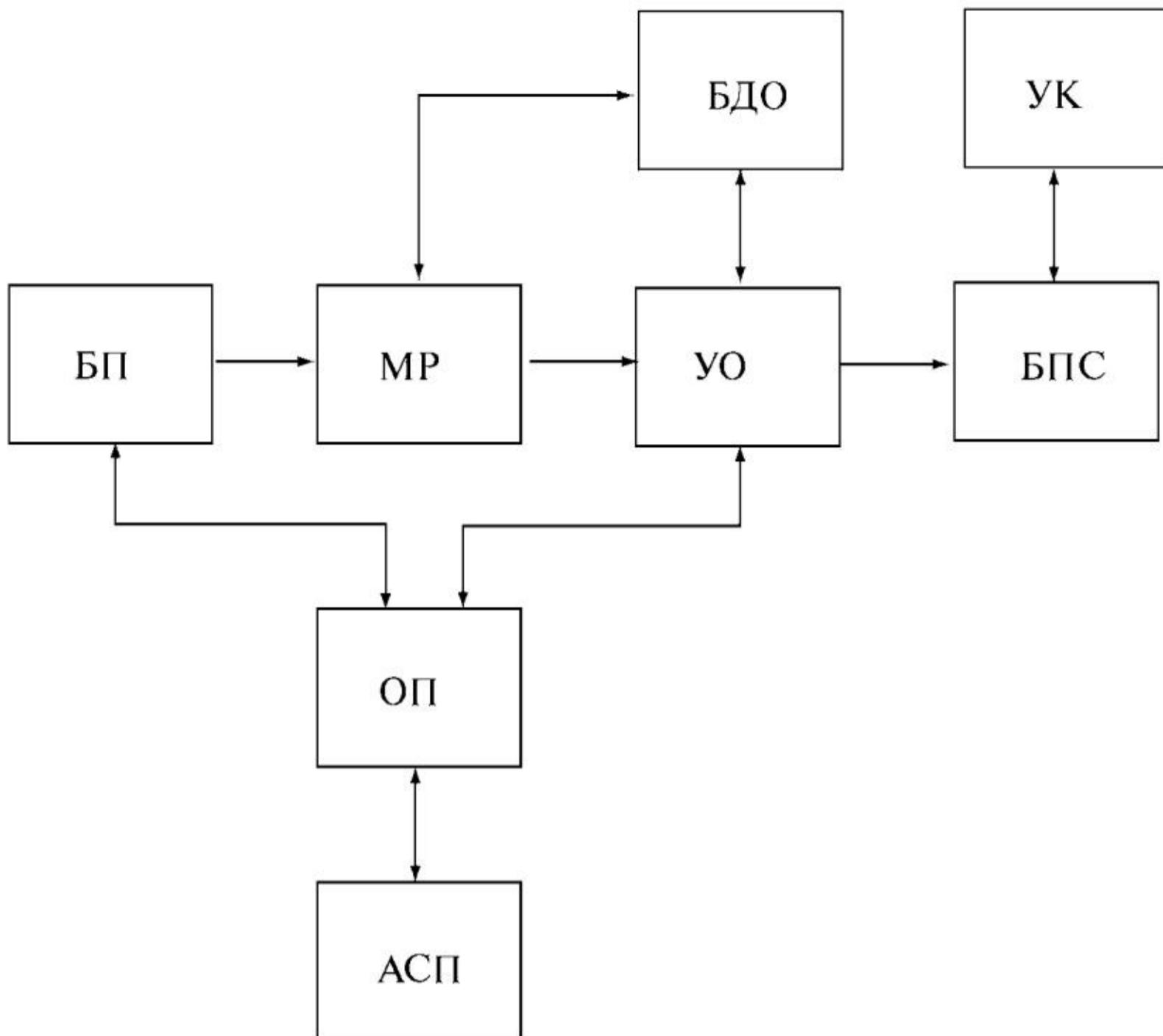


Рис. 5.3. Сетевая машина

Модуль сетевого расширения обеспечивает выполнение дополнительных команд.

Устройство обмена выполняет функции управления обменом данными между оперативной памятью и буферной памятью сообщений. Эти функции инициируются командами, выполняемыми процессором сетевой машины, а их выполнение осуществляется автономно и параллельно с продолжением программы процессора. Взаимодействие осуществляется благодаря обмену директивами между процессором и устройством обмена.

Буфер сообщений входит в состав сетевой машины и предназначен для хранения сообщений, как поступающих на вход сетевой машины, так и выдаваемых сетевой машиной. Формат сообщений и другие параметры буфера зависят от масштаба системы и алгоритмов межмашинного обмена.

Узел коммутации сети межузлового обмена, с одной стороны, является частью всей системы коммутации этой сети, а с другой – принадлежностью сетевой машины. Это приводит к его двойственной природе, поскольку управление всей сетью межузлового обмена выполняет управляющая машина, находящаяся на более высоком уровне, а управление узлом на локальном уровне осуществляет сетевая машина, управляемая, в свою очередь, управляющей машиной вычислительного узла. Для исключения конфликтов по управлению должны быть четко разделены функции управления всей сетью и функции передачи сообщений через соответствующий узел коммутации.

5.4.5.1. Команды обмена одним словом

КОП	РД	РАБС	РНБ
8	8	8	8

КОП – код операции;

РД – регистр данных;

РАБС – регистр адреса слова в буфере сообщений;

РНБ – регистр номера буфера сообщений.

Выполняется пересылка слова между регистровой памятью процессора и регистром заданного буфера сообщений.

- С1 – выдача слова в буфер сообщений;
- С2 – прием слова из буфера сообщений.

5.4.5.2. Команды обмена группой слов

Формат команды загрузки адреса

КОП	/////////	РАНБС	РНБ
8	8	8	8

КОП – код операции;

РАНБС – регистр адреса начала массива в буфере сообщений;

РНБ – регистр номера буфера сообщений,

- D0 – загрузка начального адреса.

Формат команды обмена

КОП	РАНМ	РСч	РНБ
8	8	8	8

КОП – код операции;

РНБ – регистр номера буфера сообщений;

РАНМ – регистр адреса начала массива в оперативной памяти;

РСч – регистр счетчика.

Выполняется пересылка группы слов между оперативной памятью процессора и регистрами заданного буфера сообщений. Начало массива в оперативной памяти задается в регистре РАНМ, начало массива в буфере сообщений задается в регистре РАНБС, загружаемом командой загрузки адреса.

- С3 – выдача группы слов в буфер сообщений;
- С4 – прием группы слов из буфера сообщений.

5.4.6. Архитектура и система команд обменно-редактирующей машины

Обменно-редактирующая машина предназначена для реализации программно-управляемого обмена между двумя смежными уровнями памяти, т.е. между центральной и абонентской памятью. Использование данной функции позволяет пересылать только данные, необходимые для выполнения программы. При этом достигаются две цели – экономятся время и память основной машины вычислительного узла и сокращается объем данных, пересылаемых по сети памяти. Программное управление адресацией позволяет задавать любой алгоритм обмена – от вычисления адреса по заданной формуле до использования списка адресов. Загрузка программы управления адресацией выполняется управляющей машиной соответствующего уровня по запросу программы основной машины. Возможно использование набора стандартных программ управления адресацией, перед их активизацией потребуется загрузка параметров.

Исходная информация о пересылке и редактировании данных поступает непосредственно из прикладной программы или из программ операционной системы. На основе этой исходной информации операцион-

ная система после загрузки программ и данных формирует программу обменно-редактирующей машины. Программы операционной системы выполняются управляющими машинами мониторно-моделирующей подсистемы. Управляющая машина соответствующего уровня выполняет загрузку программы основной машины или обменно-редактирующей машины своего уровня и в согласованный с выполнением прикладной программы момент запускает программу основной или обменно-редактирующей машины. Фактически загрузку выполняет обменно-редактирующая машина следующего уровня, при этом входной порт основной машины или обменно-редактирующей машины самостоятельно выполняет загрузку без прерывания работы машин своего уровня.

В системе используется глобально адресуемая память и, соответственно, на определенном этапе должно происходить преобразование виртуальных адресов в физические. Уровень, на котором выполняется преобразование адресов, зависит от масштабов системы. В максимальном масштабе это уровень мультиархитектурного вычислительного комплекса, в минимальном – уровень вычислительного узла.

При обращении к оперативным регистрам и буферной памяти модулей масштабируемой основной машины используются физические адреса. Оперативная память масштабируемой основной машины также использует физические адреса в пределах переключаемых разделов (секторов).

В управляющей машине и в обменно-редактирующей машине соответствующего уровня имеются аппаратно-программные средства преобразования адресов. После преобразования адресов обменно-редактирующая машина выполняет перемещение массивов данных конкретной задачи между уровнями. При этих перемещениях используются физические адреса общих памятей того или иного уровня и оперативной памяти масштабируемой основной машины. При этих перемещениях под управлением управляющей машины выполняются такие функции, как распределение данных между уровнями и между компонентами на одном уровне, а также програмно-управляемое редактирование, совмещенное с перемещением данных.

Обменно-редактирующая машина состоит из базового процессора (БП), оперативной памяти (ОП), модуля расширения набора команд для обменно-редактирующей машины (МР), устройства обмена (УО), буфера директив устройства обмена (БДО), блока управления центральной памятью (УЦП) и адаптера сети памяти (АСП) (рис. 5.4). Фактически обменно-ре-

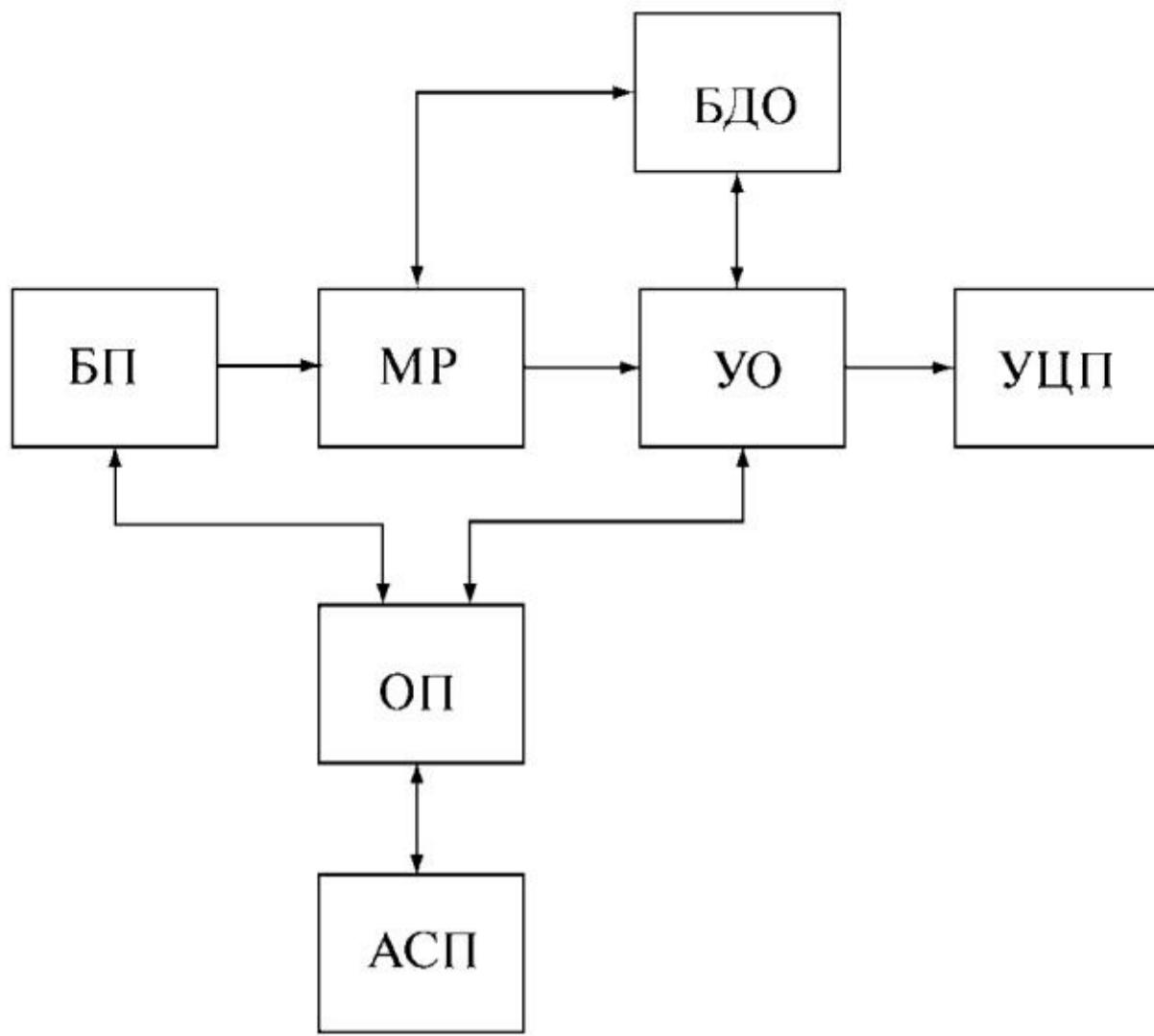


Рис. 5.4. Обменно-редактирующая машина

дактирующая машина состоит из двух процессоров, двух буферов директив и общей оперативной памяти. Одним из процессоров является процессор обменно-редактирующей машины, основанный на расширении базового процессора. Другой процессор – это узкоспециализированное устройство для управления пересылкой массивов данных между общей памятью и оперативной памятью. Процессор обменно-редактирующей машины получает задания в виде директив от управляющей машины, формирует пакет для устройства обмена, запускает обмен путем выдачи директивы для устройства обмена, принимает директивы от устройства обмена о выполнении обмена и выдает сообщение в управляющую машину о выполнении задания.

По заданию управляющей машины процессор обменно-редактирующей машины выполняет преобразование виртуальных адресов в физические, осуществляет подготовку управляющего пакета для устройства обмена. При выполнении команды пуска обмена управляющий пакет направляется в устройство обмена, которое затем осуществляет обмен автономно от процессора. По завершении обмена или при обнаружении ошибки формируется директива, направляемая в буфер директив обменно-редактирующей машины.

При программно-управляемом обмене возможны следующие разновидности формирования адресов.

1. Прямая последовательная адресация, что соответствует адресации с постоянным приращением адреса, равным 1.
2. Адресация с постоянным приращением (регулярная адресация).
3. Программно-вычисляемый адрес.
4. Косвенная адресация по списку адресов.

Прямая последовательная адресация фактически задается как адресация с постоянным приращением, равным 1. Программно-вычисляемый адрес формируется процессором. Для передачи одного слова выдается команда пуска обмена и выдается управляющий пакет для передачи одного слова. При выдаче массива слов по программно-вычисляемым адресам формируется список косвенных адресов и запускается соответствующий обмен.

5.4.6.1. Команды преобразования адреса

КОП	РВА	РФА	//////////
8	8	8	8

Команда предназначена для подготовки информации о физическом адресе начала массива пересылаемых данных. Физический адрес начала массива используется для определения начального адреса центральной памяти и формирования управляющего пакета. Аналогично формируется начальный адрес массива для абонентской памяти.

- D0 – поиск по виртуальному адресу, размещенному в регистре РВА; производится поиск в таблице преобразования адресов и запись физического адреса в регистр РФА.

5.4.6.2. Команды пересылки данных

КОП	//////////	//////////	РАП
8	8	8	8

Команды обмена с данным форматом выполняют пуск обмена и выдачу адреса управляющего пакета, который содержится в регистре, указанном в поле РАП.

- D2 – пуск обмена, чтение адреса управляющего пакета из регистра РАП и выдача его в устройство обмена.

КОП	////////	АДРЕС ПАКЕТА
8	8	16

Команды обмена с данным форматом выполняют пуск обмена и выдачу адреса управляющего пакета, который содержится в соответствующем поле команды. Пакет размещается в младшей части оперативной памяти.

- D3 – пуск обмена и выдача адреса управляющего пакета из поля команды в устройство обмена.

Формат управляющего пакета

КОП	ФЛАГИ	НП	СЧЕТЧИК		
8	8	16	32		
НАОП		СМ или НАСА			
32		32			
НАЦП					
64					
НААП					
64					

Управляющий пакет состоит из 4 слов, в которых имеются следующие поля:

КОП – код операции задает разновидность команды пересылки данных;

ФЛАГИ – признаки режимов работы устройства обмена;

НП – номер порта абонента;

СЧЕТЧИК – число передаваемых слов;

НАОП – начальный адрес оперативной памяти обменно-редактирующей машины;

СМ – смещение адреса при регулярной адресации;

НАСА – начальный адрес списка адресов при косвенной адресации;

НАЦП – начальный адрес центральной памяти;

НААП – начальный адрес абонентской памяти.

При последовательной адресации в поле СМ указывается 1, при регулярной – приращение адреса. При программно-управляемой и косвенной адресации подготавливается список адресов, начальный адрес которого находится в поле НАСА.

К числу операций устройства обмена относятся:

- 01 – пересылка данных из центральной памяти в оперативную с регулярной адресацией;
- 02 – пересылка данных из центральной памяти в оперативную с косвенной адресацией по списку адресов;
- 03 – пересылка данных из оперативной памяти в центральную с регулярной адресацией;
- 04 – пересылка данных из оперативной памяти в центральную с косвенной адресацией по списку адресов;
- 05 – пересылка данных из абонентской памяти в оперативную с регулярной адресацией;
- 06 – пересылка данных из абонентской памяти в оперативную с косвенной адресацией по списку адресов;
- 07 – пересылка данных из оперативной памяти в абонентскую с регулярной адресацией;
- 08 – пересылка данных из оперативной памяти в абонентскую с косвенной адресацией по списку адресов;
- 09 – пересылка данных из центральной памяти в абонентскую с регулярной адресацией;
- 0A – пересылка данных из центральной памяти в абонентскую с косвенной адресацией по списку адресов;
- 0B – пересылка данных из абонентской памяти в центральную с регулярной адресацией;
- 0C – пересылка данных из абонентской памяти в центральную с косвенной адресацией по списку адресов;
- 10 – операция перехода;
- 11 – выдача приказа (управляющей информации);
- 12 – выдача сообщения: параллельно с выдачей запроса в коммутатор о приеме сообщения выдается директива в процессор о завершении операции.

Учитывая необходимость заполнения буферной памяти сообщений для каждого выходного порта, а также по причине сравнительно более низкой скорости центральной памяти и ее расслоения целесообразно для заполнения буферной памяти использовать мультипрограммный режим в процессоре. Такая форма параллелизма даст возможность повысить пропускную способность подсистемы обмена. Каждая программа предназначена для вычисления адреса и выборки одного или группы слов для одного из абонентов. Возможен такой режим, при котором для каждой программы выделяется фиксированный интервал времени. Длительность этого интервала зависит от соотношения цикла модулей общей памяти, числа модулей и числа выходных портов. Такой режим процессора лучше назвать мультиканальным.

5.4.7. Архитектура и система команд периферийной машины

Периферийная машина предназначена для выполнения обмена между периферийными устройствами, накопителями на магнитных дисках и устройством общей памяти, входящим в состав системы на данном уровне.

Фактически периферийная машина состоит из двух процессоров – базового процессора и каналлера, взаимодействующих с помощью механизма директив и использующих единую оперативную память. Каналлер является специализированным процессором, выполняющим функции пересылки данных между устройствами ввода-вывода и памятью. Существует большое число терминов для подобных процессоров – устройство обмена, exchange unit, устройство управления вводом-выводом и т.п. При этом в большинстве случаев набор шин и кабелей с правилами передачи сигналов называется каналом (channel). Однако в системе IBM/360 в отличие от даже собственных разработок (система IBM Stretch) указанный набор шин и кабелей был назван input-output interface (интерфейс ввода-вывода), а устройство обмена – channel (канал). В системе АС-6 слово «канал» использовалось для обозначения специализированных каналов 1-го и 2-го уровня, используемых для, соответственно, памяти и внешних устройств. Поэтому для устройства, выполняющего передачу данных между двумя каналами, было использовано слово «каналлер».

Каналлер управляется программами базового процессора, получая от него директивы запуска программы каналлера. После завершения

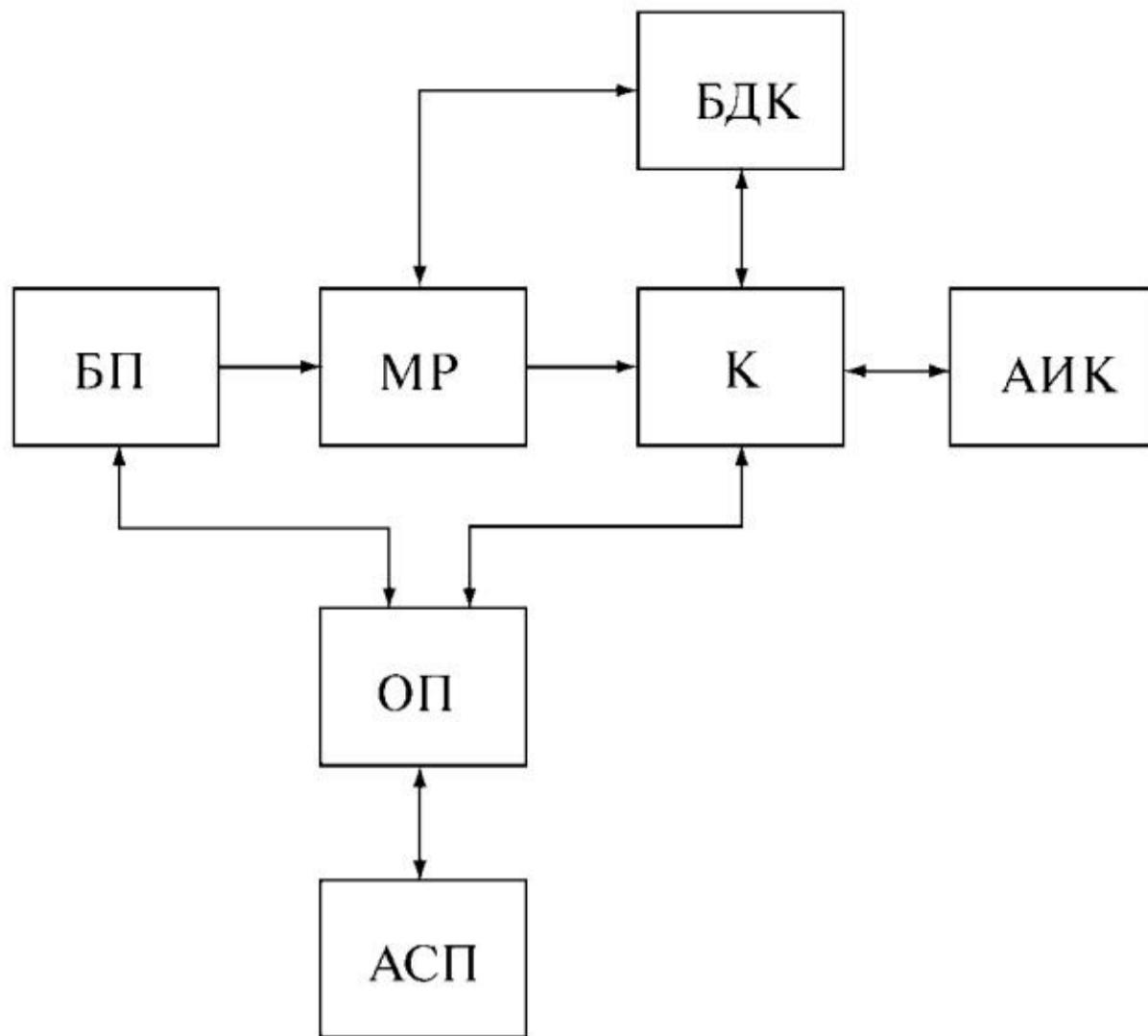


Рис. 5.5. Периферийная машина

этапа или всей своей программы каналлер направляет в буфер директив базового процессора директиву о завершении работы или обнаружении ошибки.

В зависимости от состава периферийных устройств периферийная машина может иметь специализацию, например использоваться в селекторном режиме для работы с накопителями на магнитных дисках или использоваться в мультиплексном режиме для работы с набором устройств ввода-вывода.

Периферийная машина состоит из базового процессора (БП), оперативной памяти (ОП), модуля расширения набора команд для периферийного процессора (МР), каналлера (устройства обмена) (К), буфера директив каналлера (БДК), адаптера интерфейса каналов ввода-вывода (АИК) и адаптера сети памяти (АСП) (рис. 5.5). Фактически периферийная машина состоит из двух процессоров, двух буферов директив и общей оперативной памяти. Одним из процессоров является периферийный процессор, основанный на расширении базового процессора. Другой процессор – это узко специализированное устройство для управления пересылкой массивов данных между периферийными устройствами и оперативной памятью. Периферийный процессор получает зада-

ния в виде директив от управляющей машины, формирует программу для каналлера, запускает ее путем выдачи директивы для каналлера, принимает директивы от каналлера о выполнении обмена и выдает директиву в управляющую машину о выполнении задания.

5.4.7.1. Директива пуска программ каналлера

Формат директивы

КТД	НПК	//////////	АП
8	8	16	32

КТД – код типа директивы;

НПК – номер подканала;

АП – адрес программы каналлера.

Периферийный процессор формирует и записывает в оперативную память машины программу каналлера и данные для ее выполнения. Директива формируется периферийным процессором и направляется в буфер директив каналлера.

Коды типа директивы

- 01 – начать ввод-вывод;
- 02 – проверить ввод-вывод;
- 03 – освободить ввод-вывод;
- 04 – остановить ввод-вывод;
- 05 – остановить устройство.

5.4.7.2. Система команд каналлера

5.4.7.2.1. Управляющие слова, или команды каналлера

Формат управляющего слова

КУСК	Ф	//////////	СЧЕТЧИК
8	8	16	32
АД			

64

КУСК – код управляющего слова каналлера;

АД – адрес данных;

Ф – флаги;

СЧЕТЧИК – счетчик слов.

Программа каналлера состоит из управляемых слов, каждое из которых описывает пересылаемый массив данных – начальный адрес в поле АД и размер в поле СЧЕТЧИК. В команде перехода в поле АД задается адрес очередного управляемого слова. Директива каналлера выдается в периферийный процессор после окончания программы каналлера при отсутствии флагов цепочки или после выполнения управляемого слова с флагом программно-управляемой выдачи директивы.

Код управляемого слова каналлера:

- 01 – выдача данных;
- 02 – выдача управляемой информации;
- 03 – прием данных;
- 04 – прием данных в обратном направлении;
- 05 – выдача приказа;
- 06 – уточнение состояния;
- 07 – переход в программе каналлера.

Флаги:

- флаг цепочки данных;
- флаг цепочки управляемых слов;
- флаг подавления индикации неправильной длины;
- флаг блокировки записи в память;
- флаг программно-управляемой выдачи директивы.

5.4.7.2.2. Директивы каналлера

Формат директивы

КТД	НПК	//////////	ИС
8	8	16	32

КТД – код типа директивы;

НПК – номер подканала;

ИС – информация о состоянии подканала.

Директивы каналлера выдаются в буфер директив периферийного процессора.

5.4.8. Центральная управляющая машина

Центральная управляющая машина является масштабируемым мультикомпьютером, объединяющим управляющие машины самого высокого уровня. Число объединяемых машин и конфигурация мультикомпьютера зависят от масштабов вычислительной системы.

5.4.9. Моделирующая машина

Моделирующая машина предназначена для интерактивного исследования программ в целях их корректировки путем выполнения фрагментов программ и сбора сведений об их реальном поведении. Работа моделирующей машины возможна как в составе системы, так и автономно.

Анализ программ на предмет выявления формы параллелизма выполняется в интерактивном режиме. В принципе эта функция может выполняться программным способом, например на управляющей машине, однако для этого потребуется значительное время. Программное обеспечение моделирующей машины использует ее аппаратные средства для ускорения моделирования.

Вместе с моделированием возможна предварительная фрагментация программ, определение времени выполнения фрагмента программы и корректировка фрагментации в целях выравнивания времени выполнения программ различных фрагментов. Затем выполняется распределение фрагментов программ между вычислительными узлами для их параллельного выполнения. Взаимодействующие последовательно фрагменты программ образуют конвейер программных модулей.

В состав моделирующей машины входят базовый процессор, оперативная память, модули основной машины – МДУ, МСА, МВО, МСО, а также специализированные модули для управления запуском тестовых фрагментов программ, сбора и анализа результатов тестирования. Тестирование программ может состоять только в запуске и анализе временной диаграммы программы без конкретного выполнения вычислений.

Результаты работы моделирующей машины передаются в центральную управляющую машину для дальнейшей загрузки программ и данных и запуска программ.

Разработка детальной архитектуры и системы команд моделирующей машины будет проведена совместно с разработкой принципов моделирования программ и программных средств для реализации моделей.

ГЛАВА 6.

СРАВНЕНИЕ С ЗАРУБЕЖНЫМИ ПРОЕКТАМИ. КОНЦЕПТУАЛЬНЫЙ ПРИОРИТЕТ

В проекте системы и в ряде публикаций было предложено большое число новых идей. Для сравнения с зарубежными проектами сосредоточимся только на следующих основных фундаментальных концепциях:

- неоднородная система, мониторно-моделирующая подсистема;
- взаимная адаптация архитектуры и программ;
- масштабируемый процессор основной машины;
- мультиархитектура основной вычислительной машины и функционально-специализированных машин;
- сеть памяти, обменно-редактирующая машина и иерархия памяти;
- межузловая сеть и сеть управления;
- архитектурная и конструктивная иерархия.

6.1. Неоднородная система. Мониторно-моделирующая подсистема

Основы построения неоднородных вычислительных систем были заложены при разработке систем АС-6 и «Электроника СС БИС-1», проектов систем «Электроника СС БИС-2» и «Электроника СС БИС-3». В 1995 г. был опубликован доклад, в котором были изложены новые подходы к созданию суперсистем [11]. В аннотации доклада сказано:

«Рассмотрены проблемы параллелизма, неоднородности и оптимизации программ в высокопроизводительных вычислительных системах. **Обоснована концепция создания неоднородной суперсистемы, основанной на тесном взаимодействии процессоров для скалярной, векторной и параллельной обработки.** В систему входят также **мониторно-моделирующая подсистема для анализа и подготовки задач**, дисковая подсистема и под-

система ввода-вывода. Рассмотрены подходы к повышению эффективности суперсистем.»

Приведем еще несколько цитат из данной статьи.

«Однако выводы о преимуществах той или иной системы следует делать не на основе их пиковой производительности, а на основе их эффективности при решении реальных задач, но даже и в этом случае необходимо иметь в виду преобразование или видоизменение алгоритма в зависимости от типа архитектуры. Эти преобразования не всегда могут обеспечить необходимую эффективность, если архитектура ориентирована на одну форму параллелизма. **Поэтому значительного повышения эффективности можно ожидать в неоднородных системах, содержащих подсистемы с различной и взаимодополняющей архитектурой.** Так, некоторые части задачи могут быть распараллелены для решения на системе с большим числом процессорных элементов. Другие части задачи могут быть векторизованы.»

«Работа системы, таким образом, основана на четком разделении функций. Задачи через подсистему ввода-вывода поступают в мониторно-моделирующую подсистему, где, **во-первых, анализируются на предмет формы параллелизма.** Для этого анализа и трансформации алгоритмов могут использоваться средства программного моделирования, позволяющие оценить эффективность распределения задачи внутри вычислительной подсистемы. При этом возможно интерактивное взаимодействие с программистом. Работа мониторно-моделирующей подсистемы на этом этапе завершается статической подготовкой пакетов заданий – программ и данных для вычислительной подсистемы. Динамическое распределение заданий осуществляется за счет работы системы очередей, где хранятся директивы – описатели заданий и директивы – описатели результатов выполнения заданий.»

В марте 2006 г. фирма Cray объявила о планах создания архитектуры, объединяющей несколько типов процессоров. Для сравнения приведем несколько цитат из статьи [40].

«20 марта 2006 г. Фирма Cray Inc. сегодня объявила о планах разработки суперкомпьютеров, в которых **концепция неоднородных вычислений** поднимается на совершенно новый уровень путем объединения ряда различных процессоров на единой платформе. Такие системы **«адаптивного суперкомпьютинга»** (адаптивных супервычислений) будут быстрее решать научные и технические задачи, обеспечат более высокую производитель-

ность работы программистов и конечных пользователей путем адаптации обработки данных к требованиям каждого приложения.»

«В течение нескольких следующих лет в суперкомпьютерах фирмы Cray стандартные микропроцессоры (скалярная обработка), векторные процессоры, микромультипрограммные (multithreading – мультитредовые) процессоры и аппаратные акселераторы будут объединяться на одной высокопроизводительной вычислительной платформе, на которой будет использоваться стандартная для промышленности операционная система Linux.»

«Различные приложения наилучшим образом выполняются на процессорах разных типов, однако высокопроизводительные компьютеры предлагают только один тип процессора, – сказал главный инженер Стив Скотт. Даже современные неоднородные вычислительные установки в действительности являются набором слабо связанных компьютеров с различной архитектурой, при этом не обеспечивают настоящую неоднородность и адаптивность процессоров. Фирма Cray создаст **суперкомпьютеры, которые смогут адаптироваться к приложениям** вместо того, чтобы принуждать адаптировать приложения к суперкомпьютерам. Со временем эти системы будут включать **интеллектуальные средства, которые будут анализировать приложения, определять, какой тип процессора является для него наилучшим**, и затем выполнять приложение без вмешательства пользователя.»

Из сравнения указанных статей видно, что концепция неоднородной системы, объединяющей в одной системе процессоры с различной архитектурой, и идея создания программной подсистемы для предварительного анализа задач были сформулированы в весьма близких терминах на 10 лет раньше.

6.2. Масштабируемый процессор

В 1997 и 1998 годах были опубликованы статьи по архитектуре мультиконвейерного модульного масштабируемого унипроцессора, предназначенного для неоднородной суперсистемы. В аннотации доклада [14] сказано:

«Описана архитектура унипроцессора, являющегося частью неоднородной вычислительной суперсистемы. Унипроцессор выполняет только те части больших задач, которые могут эффективно использовать зацепле-

ние векторных операций и параллельное выполнение многих векторных и скалярных операций. Унипроцессор может масштабироваться и включать различные наборы конвейерных модулей для обработки и коммутации данных. Цепочки и параллельные ветви, состоящие из модулей, обеспечивают высокую производительность благодаря их топологии с наикратчайшими связями между функциональными устройствами. В максимальной конфигурации унипроцессор может включать до **1024 модулей** и обеспечивать выполнение до 4 тысяч операций с плавающей запятой в такт или 4 TFLOPS.»

В начале 2000-х годов фирма IBM совместно с фирмами Sony и Toshiba начала разработку кристалла Cell, сообщения о котором появились в 2005 г., в частности говорилось следующее [41]:

«Cell – это неоднородный мультипроцессорный кристалл, состоящий из ядра IBM 64-bit Power Architecture, дополненного восемью специализированными сопроцессорами, построенным на основе новой архитектуры типа «одна инструкция – множественные данные» (SIMD), получившими название синергетических процессоров (SPU), предназначенных для интенсивной обработки данных, которая встречается в задачах криптографии, обработки медиаинформации и научных вычислениях. Для интеграции системы в кристалле имеются когерентные межсоединения в виде шины.»

Этот кристалл использовался в суперкомпьютере Roadrunner фирмы IBM, который в 2008 году впервые достиг производительности в 1 Pflops.

В указанных выше работах [13, 14] термин «унипроцессор» подчеркивал тот факт, что на нем может выполняться единая программа. Однако фактически его структура основана на объединении модулей, каждый из которых является векторным или скалярным процессором. Таким образом, можно утверждать, что имеет место концептуальный приоритет.

Кристалл Xeon Phi фирмы Intel, о создании которого было объявлено в 2012 г., включает 61 векторный процессор, каждый из которых может обрабатывать в векторном режиме операнды шириной 512 бит, что эквивалентно 16 операциям с плавающей запятой с 32-разрядными операндами или 8 операциям с 64-разрядными операндами [36]. Таким образом, суммарно на 60 процессорах выполняется соответственно **1920 или 960 операций**, что при тактовой частоте 1,1 ГГц обеспечивает производительность более 2 Tflops и 1 Tflops. Фирма объявила о расширении линейки

кристаллов с учетом большого спроса не только для высокопроизводительных систем, но и для серверов и систем обработки больших данных.

Масштабируемый процессор основной машины с самого начала рассматривался как самостоятельный компонент, являющийся основой для реализации различных по архитектуре структур, состоящих из равноправных машин в том смысле, что ни одна из структур не является со-процессором. В отличие от масштабируемого процессора графические процессоры и кристалл Xeon Phi являются сопроцессорами и функционируют совместно со стандартными микропроцессорами. В презентации 2014 г. [42] указано, что следующее поколение Xeon Phi будет самостоятельным процессором, а не сопроцессором или акселератором. Планируется увеличение числа ядер и повышение производительности выше 3 Tflops.

6.3. Мультиархитектура

На следующем этапе исследований в 2003 г. в докладе [15] впервые предложен термин «мультиархитектура» применительно к неоднородным вычислительным системам и компьютерам. Приведем цитату из этого доклада.

«Для создания суперсистемы, обеспечивающей высокую эффективность при решении больших задач с фрагментами с различными формами параллелизма, целесообразно тесное объединение двух подсистем – мультиконвейерного векторного процессора, ориентированного в основном на параллелизм на уровне данных, и мультипроцессора, ориентированного в основном на параллелизм на уровне задач. **Такая мультиархитектурная суперсистема должна обладать свойствами масштабируемости на всех уровнях**, а также быть открытой для подключения подсистем с другой архитектурой.»

В указанной выше статье [40], опубликованной в 2006 г. и посвященной планам фирмы Cray, было сказано:

«Фирма Cray будет внедрять адаптивные супервычисления поэтапно. На первом этапе, которому присвоено название «Rainier», будет обеспечена интеграция всех платформ фирмы Cray в интересах пользователей. На втором этапе будет создана полностью **интегрированная мультиархитектурная система**, и на заключительном этапе дальнейшее развитие систем фирмы Cray будет заключаться в создании программного обеспе-

чения для динамического распределения ресурсов, что приведет к автоматизации адаптивных супервычислений.»

Идея создания подсистемы для распределения ресурсов и анализа задач на предмет формы параллелизма была предложена в 1995 г. в упомянутой выше работе [11], где сказано:

«Задачи через подсистему ввода-вывода поступают в мониторно-моделирующую подсистему, где, во-первых, анализируются на предмет формы параллелизма. Для этого анализа и трансформации алгоритмов могут использоваться средства программного моделирования, позволяющие оценить эффективность распределения задачи внутри вычислительной подсистемы. При этом возможно интерактивное взаимодействие с программистом.

Работа мониторно-моделирующей подсистемы на этом этапе завершается статической подготовкой пакетов заданий – программ и данных для вычислительной подсистемы и помещением этой информации в системную или дисковую память.

Динамическое распределение заданий осуществляется за счет работы системы очередей, где хранятся директивы – описатели заданий и директивы – описатели результатов выполнения заданий. Последние поступают в очередь на входе мониторно-моделирующей подсистемы».

В аннотации статьи, названной «Мультиархитектура – новая парадигма для суперкомпьютеров» [16] и опубликованной в 2005 г., сказано:

«Мультиархитектурные вычислительные суперсистемы, по мнению автора предлагаемой статьи, – это следующее поколение суперкомпьютеров. **Они не только обеспечивают наилучшее согласование алгоритмов задач с возможностями аппаратных средств, но и позволяют наиболее эффективно использовать перспективные СБИС с ультравысокой степенью интеграции.**

Изложенный в статье подход начал формулироваться в начале 90-х годов. Сегодня уже разработаны концепция, принципы построения и собственно проект мультиархитектурной вычислительной суперсистемы».

В статье [43], названной «Адаптивные супервычисления – изменение парадигмы», приводится высказывание Стива Скотта:

«**Адаптивные супервычисления приведут к изменению парадигмы выбора и применения высокопроизводительных вычислительных систем. Адаптивные супервычисления необходимы для обеспечения будущих потребностей пользователей высокопроизводительных вычислительных**

систем по мере того, как необходимость повышения производительности и применения более сложных прикладных задач будет опережать эффект закона Мура. Кредо фирмы Cray: адаптировать систему к приложению, а не приложение к системе.»

В ноябре 2007 года фирма Cray объявила о выпуске семейства суперкомпьютеров Cray XT5 и XT5h (hybrid – неоднородный), в состав которых входят кластерные, векторные и реконфигурируемые узлы, объединенные новой коммуникационной сетью с топологией 3D-тора [44]. Основная вычислительная мощность в Cray XT5h обеспечивается векторными узлами Cray X2. Обе системы при соответствующем масштабировании могут обеспечить реальную производительность более 1 Pflops. В дальнейшем сообщений о выпуске этих систем не было. Фирма продолжала разработку и производство систем на микропроцессорах.

6.4. Сеть памяти, обменно-редактирующая машина

В рамках построения внешней полупроводниковой памяти для системы «Электроника СС БИС-1» был разработан специализированный процессор управления обменом [7, 8], обеспечивающий обмен между внешней полупроводниковой памятью и оперативной памятью векторно-конвейерной машины. Благодаря программному управлению адресами данных во внешней полупроводниковой памяти возможен широкий набор алгоритмов обмена, в том числе заданная индексация адреса, косвенное задание адреса, адресация для работы с подматрицей. В результате в оперативную память поступают только данные, участвующие в обработке, а центральный процессор освобождается от рутинной работы, при этом сокращаются объем используемой оперативной памяти и объем пересылаемых данных.

Концепция специализации сетей имеет большую историю. В рамках проекта мультиархитектурной системы концепция сетевой структуры со специализированными сетями была сформулирована в 2009 г. [19]. Первые шаги были сделаны в 1985 г. при разработке внешней полупроводниковой памяти в системе «Электроника СС БИС-1» [7, 8].

В [19] была обоснована эффективность выделения иерархической сети памяти, в которой передача данных с одного уровня иерархии на

другой осуществляется с помощью обменно-редактирующей машины. Эта машина выполняет широкий спектр задач по управлению массивами данных, а также редактирование данных при обмене. На каждом уровне иерархии имеется общая память, которая на разных уровнях имеет различный объем и физическую реализацию.

Аналогичный подход построения иерархической структуры использован фирмой Cray в многуровневой иерархической адаптивной системе памяти (Tiered Adaptive Storage) [45], а также в специальной системе DataWarp I/O acceleration, предложенной для системы Cray XC40 [46]. В последней системе используется идея использования специального пикового буфера burst buffer, реализованного на флеш-памяти и обеспечивающего высокую пропускную способность при обмене с дисками.

В 2014 г. фирма Fujitsu объявила о создании новой системы Next-Generation PRIMEHPC, в которой использовала специальные вычислительные средства для редактирования данных, в том числе загрузку при заданном приращении адреса, загрузку или запись по косвенным адресам в списке, разборку и сборку данных [47].

Фирма IBM в своих планах намерена в целях повышения производительности и эффективности увеличить число сопроцессоров и акселераторов. При этом ставится задача повысить скорость на уровне памяти ввода-вывода. Для этого параллельные программы разбиваются на небольшие модули. Существующие программы затрачивают большое время на перемещение данных между процессорами, оперативной и внешней памятью. Задача состоит в сокращении перемещения данных. Концепция состоит в приближении вычислений к данным во внешней памяти [48].

Подходы этих фирм носят характер локальных усовершенствований, в то время как концепция сети памяти со специализированными обменно-редактирующими машинами распространяется на все уровни системы и обеспечивает не только повышение пропускной способности и эффективности использования всех уровней памяти, но и повышение производительности за счет совмещения редактирования данных в обменно-редактирующих машинах и их обработки в основных машинах.

6.5. Межузловая сеть

Межузловая сеть имеет узкую специализацию обеспечения взаимодействия между вычислительными узлами. При этом все пересылки массивов данных выполняются в рамках сети памяти, а управляющая информация направляется по сети управления. Межузловая сеть является горизонтальной сетью, т.е. в ней отсутствует иерархическая структура, а взаимодействие между программами не зависит от централизованного управления программами в рамках мониторно-моделирующей подсистемы. В результате требования к сети по пропускной способности достаточно скромные в силу малых объемов пересылаемых данных. Простота сети позволяет организовать эффективную систему коммутации. Примеров аналогичной структуры межузловой сети пока не найдено.

6.6. Архитектурная и конструктивная иерархия

Согласование уровней иерархии в рамках архитектуры и конструкции обеспечивает наилучшие условия при масштабировании системы. Прежде всего это касается масштабирования процессора основной машины. Данная особенность системы позволяет увеличить число процессоров с различной архитектурой и в итоге повысить степень адаптации архитектуры к алгоритмам задач. В перспективе за счет гибкости сетевой структуры возможно подключение машин и подсистем, использующих новые принципы, например квантовые или молекулярные машины.

ГЛАВА 7.

ЗАКЛЮЧЕНИЕ

7.1. Основные результаты

Основным результатом исследований является разработка рациональной архитектуры вычислительной суперсистемы за счет сбалансированного использования принципов специализации, взаимной адаптации, масштабирования при сокращении потребления энергии. В основе проекта лежат следующие концепции:

- неоднородность и мультиархитектура вычислительных средств;
- масштабируемая и конфигурируемая основная машина;
- функционально-специализированные вспомогательные машины;
- специализация сетей;
- обеспечение локальности данных;
- распределенная операционная система;
- взаимная адаптация прикладных программ, архитектуры и операционной системы.

7.2. Этапы реализации проекта

Для реализации проекта мультиархитектурной системы необходимо выполнение комплекса НИР и ОКР, а также освоение новых технологий и организация новых производств.

На первом этапе целесообразно программное моделирование основных особенностей системы команд, сетевых средств и методов анализа задач.

Затем после корректировки архитектурных особенностей необходимы разработка макетных образцов основных модулей и создание инструментария для построения аппаратно-программных моделей как среды для параллельной разработка аппаратных решений и программного обеспечения, в том числе САПР.

Параллельно с указанными работами необходимо освоение технологии изготовления всего комплекса БИС, в том числе масштабируемого, т.е. состоящего из заданного для данного класса задач, набора модулей. Необходимо освоение технологии изготовления блоков и средств межсоединений, а также систем питания и охлаждения (отвода тепла).

Определение приоритетных направлений обеспечит рациональное планирование работ в соответствии с потребностями. Очевидно, что к числу первоочередных должны быть отнесены работы по созданию отечественных вычислительных суперсистем на отечественной элементной базе.

На завершающем этапе должны быть проведены ОКР по всему фронту разработки, изготовления и наладки системы.

7.3. Перспективы развития

Основной задачей при создании суперкомпьютеров является достижение сверхэффективности за счет простоты, сокращения паразитных потерь и специализации (взаимной адаптации). При выполнении прикладной программы в универсальном процессоре нельзя достичь максимальной эффективности из-за лишних функций, а при выполнении программ операционной системы используется небольшая часть ресурсов процессора. По-существу массивно-параллельные системы на общедоступных микропроцессорах нельзя считать настоящими суперкомпьютерами, поскольку слишком велики расходы на распараллеливание, локализацию данных и управление программами.

Настоящие суперкомпьютеры – это CDC 6600, CDC 7600, БЭСМ-6, все модели векторных машин фирмы Cray и фирмы NEC. Последняя сохранила векторную архитектуру и разработала собственные кристаллы.

Микропроцессоры-«киллеры» действительно ликвидировали настоящие суперкомпьютеры и повернули развитие в ложном направлении. Сохранение совместимости обирается потерями эффективности из-за паразитных потерь, избыточной энергии и дополнительными искусственными средствами адаптации в операционной системе Microsoft в каждом процессоре Intel и системе Cuda в графических процессорах NVIDIA.

Возврат к нормальному развитию неизбежен. Сейчас он начинается с акселераторов и сопроцессоров, но эти половинчатые решения должны

привести к замене микропроцессоров на специализированные мультиархитектурные конструкции, в которых объединяются наборы процессоров для прикладных программ с различными формами параллелизма и наборы функционально-специализированных процессоров. Таким образом, в мультиархитектурной системе будут объединяться **различные, но равноправные** процессоры, а не акселераторы или сопроцессоры, играющие вспомогательные роли. При этом все типы процессоров должны иметь максимально простую архитектуру для обеспечения возможности разработки эффективных и надежных программ. Надо вернуть программистам прозрачность архитектуры вместо автоматизации всего и вся. Это возможно только при простой архитектуре и аппаратуре, не требующей для управления дополнительных ресурсов, кроме собственно вычислительных. Однако программисты не должны забывать о локализации данных и управлении перемещением данных. Собственно прикладные программы должны выполняться не только в проблемно-ориентированных процессорах, но и во вспомогательных процессорах, таких как обменно-редактирующие. Все управление следует вынести на функционально-специализированные процессоры. Удобство программирования и прозрачные средства распараллеливания необходимы для того, чтобы программисты начали реально создавать параллельные программы. Еще раз следует подчеркнуть важность эффективных подсистем ввода-вывода и управления данными.

Современное состояние высокопроизводительных систем в значительной степени определяется коммерческими соображениями, при этом, поскольку основными заказчиками на самые высокопроизводительные системы являются государственные структуры, конечная цена определяется ведущими монополиями. Кроме того, создание «монастыров», потребляющих более 30 МВт, показывает, что это тупиковый вариант развития. Из-за неудачных попыток внедрения ряда уникальных разработок, таких как системы Cray X1, Cray XT5h, микропроцессоры IBM Cell, DEC Alpha, Intel i860 не выдержали конкуренции не только по техническим причинам. Фирма Intel продолжает проведение жесткой монопольной политики, не желает отказываться от совместимости даже при разработке векторного процессора Phi и сохраняет прикладное программное обеспечение и операционную систему в каждом ядре. Выпуск фирмой этого процессора на внешние рынки с опережением его освоения

ния на рынке США также свидетельствует о стремлении к сдерживанию самостоятельного развития в других странах.

В настоящее время в США, Японии, Китае, Индии, России и Европейском союзе предложены планы создания систем с экзафлопсной производительностью. Основные проблемы связаны с преодолением проблем резкого роста потребляемой энергии, обеспечения локальности данных, обеспечения высокой степени параллелизма и надежности. Переход к петафлопсной производительности сопровождался сохранением совместимости и отсутствием радикальных изменений в архитектуре и технологии. По мнению большинства экспертов, для следующего этапа потребуются исследования и разработки неоднородных систем, новой архитектуры, новой элементной базы, а также нового подхода к программированию [49, 50].

Администрация США выделила 24,4 млн долларов на новую инициативу DesignForward и контракты с фирмами AMD, Cray, IBM, Intel Federal и NVIDIA. Три ведущие национальные лаборатории сформировали проект CORAL (Collaboration of Oak Ridge, Argonne and Livermore) для разработки двух проектов [51]. Первый проект – вычислительная система Aurora, выпуск которой запланирован на 2018 г. Основным исполнителем финансируемого правительством США в объеме 200 млн долларов проекта является фирма Intel, соисполнителем – фирма Cray [52]. Планируется использование третьего поколения кристалла Xeon Phi – Knights Hill, реализуемого по технологии 10 нм. Производительность кристалла – 4–5 TFlops для операций с плавающей запятой с двойной точностью. Производительность системы – 180 Pflops с возможностью увеличения до 450 Pflops. Второй проект выполняют совместно фирмы IBM и NVIDIA. Система Summit, реализуемая на микропроцессорах IBM Power9 и графических процессорах NVIDIA Volta, будет иметь производительность 150–300 Pflops [53].

В Японии объявлена разработка нового варианта системы K Computer с экзафлопсной производительностью [54]. Ожидаемый срок установки системы – 2019 г., запуск производства – 2020 г. К работе подключены три ведущие фирмы – Fujitsu, Hitachi и NEC, четыре университета и исследовательский центр RIKEN. Центральными аспектами проекта являются разработка оптимальных акселераторов для массивной неоднородной системы, называемой экстремальной SIMD-архитектурой, и совершенствование системы памяти, в том числе включение памяти с высокой про-

пускной способностью. Заманчивые перспективы имеет также применение векторного процессора фирмы NEC для системы SX-ACE [55].

В Китае установлена самая высокопроизводительная система Tianhe-2 с производительностью 34 Pflops. Ожидается, что в 2015 г. будет выпущено две системы с производительностью 100 Pflops, при этом одна полностью будет реализована на разработанных и изготовленных в Китае кристаллах процессоров и схем межсоединений [56]. Отмечается, что основные усилия будут направлены на разработку базовых алгоритмов, средств распараллеливания для систем на кристаллах с большим числом простых модулей (*many-core*) и сетевых ресурсов [57].

В нашей стране собственные исследования и разработки проводились в условиях отсутствия производства микропроцессоров и других компонентов, что привело к необходимости ориентации на использование импортных изделий. По существу должна ставиться задача развертывания исследований и разработок всего спектра технологий, включая электронное машиностроение. При этом программа работ не должна повторять ошибочный с технической точки зрения путь (выгодный с точки зрения быстрого получения прибыли) – от персональных компьютеров к суперсистемам, а должна строиться наоборот – от суперсистем ко всем классам машин, включая персональные.

Создание собственной архитектуры и собственного программного обеспечения является абсолютно необходимым условием обеспечения стратегической независимости и гарантии непрерывного развития. Только при этих условиях возможно освобождение от влияния глобальных монополий, которые в области информационных технологий влияют не только на экономические аспекты, но и на безопасность, социальные и культурные стороны жизни. Кроме того, следует учесть, что заимствование и повторение известных разработок, в частности т.н. импортозамещение, неизбежно приводят к отставанию. Однако, естественно, нельзя впадать в другую крайность и не изучать и не использовать зарубежный опыт [58–61].

Основные преимущества концептуального проекта мультмархитектурной вычислительной суперсистемы основаны на взаимной адаптации аппаратной реализации, архитектуры и программного обеспечения. Благодаря этому может быть достигнута весьма высокая эффективность аппаратуры за счет параллелизма на всех уровнях – на уровне команд, на уровне данных, на уровне малых программных модулей и на уровне

программ. Это относится ко всем рассмотренным особенностям системы. Выделим лишь наиболее существенные. Выбор архитектуры масштабируемого процессора, структуры вычислительного узла обеспечивают эффективность вычислительного процесса. Гибкость сети памяти позволяет оптимизировать подготовку данных, что также способствует повышению эффективности вычислительного процесса. Простота и узкая специализация межузловой сети сокращают накладные расходы на обеспечение параллелизма программ, выполняемых в различных вычислительных узлах. Мониторно-моделирующая подсистема, иерархия которой согласована с иерархией системы и сети памяти, обеспечивает оптимальный выбор аппаратных средств для выполнения той или иной части задачи, эффективную подготовку программ и данных, оптимизацию вычислительного процесса. Дальнейшее успешное развитие системы должно выполняться при параллельной разработке как аппаратных, так и программных средств.

Изложенные в проекте концепции и архитектурные решения могут быть применены и при создании новых систем, включающих нетрадиционные подходы. Так, наиболее вероятной структурой квантовых компьютеров будет та или иная форма их объединения и взаимодействия с классическими компьютерами. Для управления квантовым компьютером необходима подсистема управления, построенная на основе классической вычислительной системы. Предложенная в проекте мониторно-моделирующая подсистема содержит не только подсистему управления, но и средства анализа задач на предмет их соответствия той или иной архитектуре. Развитие этих средств в целях анализа задач для квантового компьютера даст возможность использовать еще одну форму взаимодействия – объединение классического и квантового компьютеров в единую мультиархитектурную систему.

Таким образом, развитие концепции мультиархитектуры в перспективе может быть использовано для совершенствования классических вычислений на основе новых технологий и повышения эффективности аппаратных и программных средств, а также для создания уникальных новых систем.

Литература

1. Митропольский Ю.И. БЭСМ-6, АС-6 и их влияние на развитие отечественной вычислительной техники // Информационные технологии и вычислительные системы. – М.: ОИТВС РАН, ИМВС РАН, 2002. – № 3. – С. 49–58.
2. Митропольский Ю.И. Вопросы соотношения универсальных и специализированных средств в вычислительных системах // Кибернетика и вычислительная техника / под ред. В.А. Мельникова, выпуск 1. – М., «Наука», 1985. – С. 35–48.
3. Мельников В.А., Митропольский Ю.И. Пути построения ЭВМ сверхвысокой производительности // Электронная вычислительная техника / под ред. В.В. Пржиялковского, вып. 1. – М., «Радио и связь», 1987. – С. 12–17.
4. Митропольский Ю.И. Концепции иерархического построения высокопроизводительных вычислительных систем // Кибернетика и вычислительная техника / под ред. В.А. Мельникова, вып. 5. – М., «Наука», 1991. – С. 7–14.
5. Мельников В.А., Митропольский Ю.И., Малинин А.И., Романков В.М. Требования к конструкции высокопроизводительных ЭВМ и проблемы ее реализации // Вопросы кибернетики. Комплексное проектирование элементно-конструкторской базы суперЭВМ / под ред. В.А. Мельникова и Ю.И. Митропольского, НСКАН СССР. – М., 1988.
6. Мельников В.А., Митропольский Ю.И., Шнитман В.З. Архитектура высокопроизводительной вычислительной системы «Электроника СС БИС-1». – Программные продукты и системы, № 1, 1992.
7. Митропольский Ю.И., Захаров Ю.В., Усан А.А., Шнитман В.З. Организация управления полупроводниковой внешней памятью высокопроизводительной вычислительной системы // Тезисы докладов на I Всесоюзной конференции «Проблемы создания суперЭВМ, суперсистем и эффективность их применения», Минск, 15–17 сент. 1987 г. – Институт математики АН БССР, Минск, 1987 г.
8. Митропольский Ю.И., Захаров Ю.В., Усан А.А., Шнитман В.З. Процессор управления полупроводниковой внешней памятью высокопроизводительной вычислительной системы // Авторское свидетельство № 1539789, приор. 14.01.87, зарегистрировано 01.10.89.

9. Мельников В.А., Митропольский Ю.И., Шнитман В.З. Научные, технологические и методические аспекты создания вычислительной системы «Электроника СС БИС-1» // Юбилейный сборник трудов отделения информатики, вычислительной техники и автоматизации Российской академии наук. – М.: ОИВТА РАН, 1993. – С. 28–41.
10. V. A. Melnikov, Yu. I. Mitropolski, G.V. Reznikov. Designing the Electronica SS BIS Supercomputer // IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A, June 1996. Vol. 19. No. 2, pp. 151–156.
11. Митропольский Ю.И. Концепции построения неоднородных вычислительных суперсистем // Распределенная обработка информации. – Труды Пятого Международного семинара. – Новосибирск: Институт физики полупроводников СО РАН. – 1995. – С. 42–46.
12. Анохин А.В., Ленгник Л.М., Митропольский Ю.И., Пучков И.И. Архитектура неоднородной вычислительной суперсистемы // Распределенная обработка информации. – Труды Пятого Международного семинара. – Новосибирск: Институт физики полупроводников СО РАН. – 1995. – С. 22–27.
13. Митропольский Ю. И. Мультиконвейерный унипроцессор // Вычислительные машины с нетрадиционной архитектурой. СуперВМ. Выпуск 5. Неоднородные вычислительные суперсистемы. – М.: ИВВС РАН. – 1997. – С. 50–64.
14. Митропольский Ю.И. Архитектура мультиконвейерного модульного масштабируемого унипроцессора // Труды Шестого Международного семинара «Распределенная обработка информации». – Новосибирск: Институт физики полупроводников СО РАН. – 1998. – С. 30–34.
15. Митропольский Ю.И. Мультиархитектурная вычислительная суперсистема // Труды Первой Всероссийской научной конференции «Методы и средства обработки информации». – М.: МГУ. – 2003. – С. 131–136.
16. Митропольский Ю.И. Мультиархитектура – новая парадигма для суперкомпьютеров // Электроника: наука, технология, бизнес. – 2005. – № 3. – С. 42–47.
17. Митропольский Ю.И. Масштабируемый векторный процессор в составе мультиархитектурной суперсистемы // Труды Второй Всероссийской научной конференции «Методы и средства обработки информации». – М.: МГУ. – 2005. – С. 47–52.

18. Митропольский Ю.И. Проект многоуровневой масштабируемой мультиархитектурной вычислительной системы // Труды Четвертой международной конференции «Параллельные вычисления и задачи управления». – Москва, 27–29 октября 2008 г. Институт проблем управления им. В.А. Трапезникова. – 2008. – С. 533–558.
19. Митропольский Ю.И. Принципы построения сетевой структуры мультиархитектурной вычислительной системы // Суперкомпьютерные технологии: разработка, программирование, применение (СКТ –2010) // Материалы Международной научно-технической конференции. Т. 1. – Таганрог: Изд-во ТТИ ЮФУ. – 2010. – С. 136–140.
20. Thornton, J. Design of a Computer – The Control Data 6600 // Glenview, Il: Scott, Foresman and Co, 1970.
21. Russell Richard M. The CRAY-1 Computer System//Communications of the ACM, Jan. 1978. V. 21, № 1, pp. 63–72.
22. Cray X-MP // http://en.wikipedia.org/wiki/Cray_X-MP.
23. Cray-2 // <http://en.wikipedia.org/wiki/Cray-2>.
24. Japanese 'Computenik' Earth Simulator shatters US supercomputer hegemony. – <http://www.hoise.com/primeur/02/articles/weekly/AE-PR-05-02-59.html>.
25. Dunigan T.H., Jr., Fahey M.R., White J.B. III, Worley P.H. Early Evaluation of the Cray X1 // Proceedings of the IEEE/ACM SC2003 Conference, Nov. 15–21, 2003.
26. NEC Launches World's Fastest Vector Supercomputer, SX-9 // <http://www.nec.co.jp/press/en/0710/2501.html>.
27. Gara, A, Blumrich, M A, Chen, D, Chiu, G L-T Et al Overview of the Blue Gene/L system architecture // IBM Journal of Research and Development / Mar-May 2005.
28. The Advantages of First-Generation Heterogeneous Computing on the Cray XT5h // SciDAC Review, Summer 2008, pp. 42-49, www.scireview.org.
29. Barker, Kevin J.; Davis, Kei; Hoisie, Adolfy; Kerbyson, Darren J.; Lang, Mike; Pakin, Scott; Sancho, Jose C. Entering the petaflop era: The architecture and performance of Roadrunner // International Conference for High Performance Computing, Networking, Storage and Analysis, 2008, pp. 1–11.

30. Bland A. S., Kendall R. A., Kothe D. B., Rogers J. H., Shipman G. M. Jaguar: The World's Most Powerful Computer // paper presented at CUG 2009, ORNL.
31. Tianhe-1A // <http://en.wikipedia.org/wiki/Tianhe-1A>.
32. Feldman, Michael Japanese Supercomputer is New TOP500 Champ // June 20, 2011 - http://www.hpcwire.com/hpcwire/2011-06-20/japanese_supercomputer_is_new_top500_champ.html.
33. DOE Labs Set Records with IBM Blue Gene/Q, November 28, 2012 // http://www.hpcwire.com/hpcwire/2012-11-28/doe_labs_set_records_with_ibm_blue_gene_q.html.
34. Oak Ridge Claims No. 1 Position on Latest TOP500 List with Titan // Nov. 12, 2012, - <http://www.top500.org/blog/lists/2012/11/press-release/>.
35. Burt, Jeffrey China's Tianhe-2 Still the World's Fastest Supercomputer // <http://www.eweek.com/servers/chinas-tianhe-2-still-the-worlds-fastest-supercomputer.html>.
36. Intel Reveals Architecture Details of Intel Xeon Phi Co-Processor, August 30, 2012 // <http://www.cdrinfo.com/Sections/News/Details.aspx?NewsId=34114>.
37. The Dally-nVIDIA-Stanford Prescription for Exascale Computing, 11/27/2011 // <http://www.monolithic3d.com/blog/the-dally-nvidia-stanford-prescription-for-exascale-computing>.
38. IBM System/360 // http://en.wikipedia.org/wiki/IBM_System/360.
39. IBM zEnterprise System // http://en.wikipedia.org/wiki/IBM_zEnterprise_System.
40. Cray Will Leverage an "Adaptive Supercomputing" Strategy to Deliver the Next Major Productivity Breakthrough // Seattle, Wa, March 20, 2006. – <http://investors.cray.com/phoenix.zhtml?c=98390&p=irol-newsArticle&ID=833494&highlight=>.
41. The Cell project at IBM Research - The Cell Architecture // August 2005. – http://researcher.ibm.com/view_project.php?id=2649.
42. Antypas, Katie Preparing your Application for Advanced Manycore Architectures // CSGF HPC Workshop July 17, 2014, <http://www.krellinst.org/csgf/conf/2014/video/kantypas>; <http://www.krellinst.org/doecsgf/conf/2014/pres/kantypas.pdf>.
43. Lazou, Christopher Cray's Adaptive Supercomputing - A Paradigm Shift // March 24, 2006 - http://www.hpcwire.com/hpcwire/2006-03-24/crays_adaptive_supercomputing_-_a_paradigm_shift-1.html.

44. Cray Introduces Next-Generation Supercomputers // Seattle, Wa, Nov 06, 2007. – <http://investors.cray.com>.
45. Feldman, Michael. The Big Data Challenge: Intelligent Tiered Storage at Scale // Intersect360 Research, White paper, November 2013.
46. Hemsoth, Nicole Cray Strikes Balance with Next-Generation XC40 Supercomputer // <http://www.hpcwire.com/2014/09/30/cray-strikes-balance-next-generation-xc40-supercomputer/>, September 30, 2014.
47. Trader, Tiffany Fujitsu Targets 100 Petaflops Supercomputing Next-Generation PRIMEHPC // Fujitsu, 2014, August 12, <http://www.hpcwire.com/2014/08/12/fujitsu-targets-100-petaflops-supercomputing/>.
48. Shah, Agam IBM shares plans for supercomputing future // Nov 13, 2014 <http://www.networkworld.com/article/2847876/data-center/ibm-shares-plans-for-supercomputing-future.html>.
49. Toward Exascale Computing with Heterogeneous Architectures. – http://sc10.supercomputing.org/schedule/event_detail.php?evid=pan129.
50. Wolfe, Michael. The Heterogeneous Programming Jungle // March 19, 2012. – http://www.hpcwire.com/hpcwire/2012-03-19/the_heterogeneous_programming_jungle.html?featured=top.
51. Russel, John, Trader, Tiffany. White House Launches National HPC Strategy // July 30, 2015 <http://www.hpcwire.com/2015/07/30/white-house-launches-national-hpc-strategy/>.
52. Hemsoth, Nicole. Future Intel Chips Shine in 180 Petaflops Argonne Supercomputer // April 9, 2015 <http://www.theplatform.net/2015/04/09/future-intel-chips-shine-in-180-petaflops-argonne-supercomputer/>.
53. Smith, Ryan NVIDIA Volta, IBM POWER9 Land Contracts For New US Government Supercomputers // November 17, 2014 <http://www.anandtech.com/show/8727/nvidia-ibm-supercomputers>.
54. Trader, Tiffany. Japan Moves Forward with 2020 Exascale Plans // January 8, 2014 <http://www.hpcwire.com/2014/01/08/japan-moves-forward-2020-exascale-deadline/>.
55. NEC begins developing next-generation vector supercomputer // November 17, 2014 http://www.nec.com/en/press/201411/global_20141117_02.html.
56. Morgan, Timothy Prickett. China fires up homegrown petaflops super. – http://www.theregister.co.uk/2011/10/31/china_shenwei_bluelight_supercomputer/.

57. Gelber, Robert. China Looks to a National Processor Architecture // April 24, 2012. - http://www.hpcwire.com/hpcwire/2012-04-24/china_looks_to_a_national_processor_architecture.html.
58. Митропольский Ю.И. Проблемы разработки новой архитектуры процессоров и вычислительных систем // Труды Физико-технологического института РАН / Гл. ред. А.А. Орликовский, ФТИАН, Москва, Наука. – Т. 23, 2013. – С. 109–140.
59. Митропольский Ю.И. Элементная база и архитектура будущих суперкомпьютеров // Микроэлектроника, № 3, 2015 // Микроэлектроника, том 44, 2015, № 3. – С.163–179.
60. Mitropol'skii Yu. I. Electronic Components and Architecture of Future Supercomputers // ISSN 1063-7397, Russian Microelectronics, 2015. Vol. 44. No. 3, pp. 139–153. © Pleiades Publishing, Ltd., 2015.

61. Митропольский Ю.И. Новые концепции построения вычислительных суперсистем // Труды Физико-технологического института РАН / Гл. ред. А.А. Орликовский, ФТИАН, Москва, Наука. – Т. 25, 2015.

