Пензенский государственный университет

А.П. Ремонтов, А.П. Писарев

Вычислительные машины и системы

Учебное пособие

Пенза 2006 г.

Аннотация

В настоящем пособии изложены принципы построения современных ЭВМ и систем, их классификация и области применения. Рассмотрены и проиллюстрированы примерами вопросы организации систем памяти, процессоров и систем ввода-вывода в ЭВМ, даны арифметические и логические основы их построения и приведена классификация и понятия о параллельных вычислительных системах. Пособие предназначено для студентов института информатики и вычислительной техники специальности 230102 и 080801.

Содержание

Введение

В настоящее время основой развития всех сфер жизни общества – материального производства, образования, науки и культуры – стало получение информации, её обработка и эффективное использование.

Доминирование компьютерных технологий стало возможным благодаря совершенствованию технологий создания и эксплуатации электронных вычислительных средств (ЭВС), построенных на основе высокопроизводительных ЭВМ. Применение компьютеров в самых различных сферах человеческой деятельности обуславливает необходимость изучения основ построения средств вычислительной техники (ВТ) специалистами, работающими в промышленности, научных и коммерческих организациях и структурах.

В настоящем пособии изложены принципы построения современных ЭВМ и систем, их классификация и области применения. Рассмотрены и проиллюстрированы примерами вопросы организации систем памяти, процессоров и систем ввода-вывода в ЭВМ, даны арифметические и логические основы их построения и приведена классификация и понятия о параллельных вычислительных системах. Пособие предназначено для студентов Института информатики и вычислительной техники Пензенского государственного университета и соответствует государственному образовательному стандарту, а также программам обучения по дисциплинам «Организация ЭВМ и систем» и «Вычислительные машины и системы».

1 ПРИНЦИПЫ ОРГАНИЗАЦИИ ЭВМ.

1.1 Факторы, определяющие принципы организации ЭВМ

Определяющее влияние на внутреннюю организацию ЭВМ оказывают два фактора - назначение ЭВМ и элементная база.

Влияние элементной базы. В ЭВМ используется исключительно двоично-кодированная форма представления информации. Пример: двоичная и десятичная арифметика – отличаются как небо и земля в смысле сложности алгоритмов операций и, как следствие, сложности устройств, реализующих эти операции. Более простой пример: триггер как элемент хранения двоичной цифры и элемент с десятью устойчивыми состояниями как элемент хранения десятичной цифры (существенно сложнее и дороже триггера). При использовании двоично-кодированной формы существенно возрастает надёжность элементов и ЭВМ в целом.

Второй фактор **назначение ЭВМ.** Из определения ЭВМ (автоматизация обработки информации на основе алгоритмов) следует, что принципы организации ЭВМ неизбежно должны зависеть от свойств алгоритмов. Наиболее существенное влияние на организацию

ЭВМ оказывают следующие три свойства алгоритмов.

1. Детерминированность (однозначность) вычислительных процессов, порождаемых алгоритмами.
2. При описании алгоритмов используется конечный набор элементарных операций. Примеры из начальной школы: правила умножения, деления и т.д.
3. Дискретное представление информации, с которой оперируют алгоритмы

Детерминированность процессов – это основное свойство алгоритмов, которое позволило Джону фон Нейману использовать алгоритм как основу, источник управления процессом вычислений, процессом обработки информации в ЭВМ. А именно: алгоритм представляется в форме программы, вводится в память машины и используется для управления вычислительным процессом (отсюда, кстати, потребность в “армии” программистов).

Конечный набор элементарных операций – отсюда вытекает, что и аппаратура ЭВМ (т.е. ВК) должна выполнять конечный набор сравнительно простых операций: сложение, вычитание, умножение, деление и др. Следовательно, F={+,–,×,/,...} - список машинных операций конечен и сравнительно прост.

Дискретное представление информации, с которой оперируют алгоритмы. Из этого свойства следует, что информация в ЭВМ представляется исключительно в дискретной форме — числовой, символьной, в форме логических значений. Причём, с учётом фактора элементной базы – не просто числовой, символьной и т.д., а ещё и в двоично-кодированной форме.

Анализируя сказанное, можно сформулировать принципы построения и функционирования современных ЭВМ в виде нескольких основных тезисов. Впервые их сформулировал Джон фон Нейман в 1945 году под названием “Принципы программного управления ЭВМ”. В популярном изложении их можно сформулировать следующим образом:

1. Информация, подлежащая обработке с помощью ЭВМ, кодируется в двоичной форме и разделяется на **единицы информации - слова**. Слово - это совокупность двоичных элементов a_1, a_2, ..., a_k, где $a_i \in \{0,1\}$, k=8,16,32,64, k=const.
2. Перед обработкой слова информации (исходные данные) размещаются в ячейках памяти ЭВМ. **Ячейка памяти** - это место хранения одного слова информации. Ячейки памяти нумеруются. Номер ячейки памяти называют **адресом**.
3. Алгоритм обработки информации представляется в виде последовательности **управляющих слов** - т.н. **команд**. Каждая команда задаёт, предписывает аппаратуре ЭВМ тип выполняемой операции (указывает одну операцию из списка F), т.е. указывает аппаратуре что делать. Кроме того, команда, в случае необходимости,

указывает и местоположение операндов в памяти машины путём указания номера ячейки, т.е. указывает аппаратуре, где взять данные для обработки. Алгоритм, представленный в терминах команд, называют **программой**.

4. Команды, как и данные, кодируются в двоичной форме и располагаются в ячейках памяти ЭВМ.
5. Выполнение операций, предписанных программой, сводится к поочерёдному выбору команд из памяти и их выполнению (интерпретации) аппаратурой ЭВМ. Порядок, в котором команды извлекаются из памяти, задаётся алгоритмом решения задачи и зависит от исходных данных.

Основная особенность ЭВМ, работающих по принципу программного управления, - универсальность. **Универсальность ЭВМ** вытекает из анализа сформулированных фон Нейманом принципов программного управления: функция ЭВМ задаётся программой, введённой в память ЭВМ, а не аппаратурой ЭВМ. Аппаратура ЭВМ может выполнять только операции из списка машинных операций F. Именно программа задаёт тот порядок, в котором операции должны выполняться для решения задачи (именно программа обеспечивает аранжировку операций). Таким образом, замена программы в памяти легко приводит к изменению функций ЭВМ, реализуемых аппаратурой ЭВМ. Универсальность – это, несомненно, основное достоинство ЭВМ.

Недостатки фон Неймановских машин. Свойство универсальности является и основным недостатком! Почему? Дело в том, что для решения задачи алгоритм разрабатывается человеком и в форме программы загружается в память ЭВМ. Именно программа и несёт в себе **всю** необходимую для решения задачи информацию. Если человек допустил ошибку, то решение будет неверным. Аппаратура ЭВМ лишь быстро и надёжно (т.е. без собственных ошибок) реализует ее. Следовательно, аппаратура ЭВМ не обладает интеллектом и не может быть помощником человеку при решении интеллектуальных задач, например – при разработке новых алгоритмов. Следовательно, ЭВМ - это просто автоматизированный калькулятор.

В связи с этим недостатком уже много лет актуальной является задача пересмотра классических принципов построения ЭВМ и поиск более рациональных. Переход к новым принципам организации ЭВМ специалисты связывают с появлением машин пятого поколения.

К настоящему времени сменилось четыре поколения машин. Все они фон Неймановские по принципу построения: первое поколение – ламповые ЭВМ, второе – ЭВМ на основе полупроводниковых дискретных элементов - транзисторов и интегральных схем (ИС) малой и средней степени интеграции, третье поколение – ЭВМ на основе ИС, чет-

вертое поколение – ЭВМ на основе микропроцессорных больших ИС (БИС).

1.2 Области применения ВТ и основные типы ЭВМ

Развитие вычислительной техники и сферы и методов ее использования - процессы взаимосвязанные и взаимообусловленные. С одной стороны, потребности народного хозяйства, науки и культуры стимулируют поиски учеными и конструкторами новых путей построения ЭВМ, а с другой стороны, появление электронных вычислительных машин, систем и устройств с большими функциональными возможностями, с существенно улучшенными показателями по производительности, стоимости, габаритным размерам, надежности и т.п. создает предпосылки для непрерывного расширения областей развития форм применения ЭВМ. Первоначально сравнительно узкая сфера применения ЭВМ, главным образом для научных и технических расчетов, в короткий срок существенно расширилась и охватила почти все области науки, техники, планирования и управления технологическими процессами, все области человеческой деятельности, связанные с обработкой больших объемов информации. Разнообразие областей и форм использования ЭВМ породило широкий спектр требований к характеристикам и особенностям организации машин и систем. В результате к настоящему времени в соответствии с областями применения определились основные типы ЭВМ, которые при сохранении указанных в п.1.2 основных фундаментальных принципов существенно разнятся не только по количественным характеристикам, но и по архитектуре, электронно-технологической базе и используемым периферийным устройствам. Основные средства современной вычислительной техники можно классифицировать следующим образом: сверхпроизводительные ЭВМ,ЭВМ общего назначения, малые ЭВМ, микро-ЭВМ, микропроцессоры.

К сверхпроизводительным машинам (системам) в настоящее время относят машины (системы), выполняющие свыше 100 млн. операций/с. Подобные машины используются для решения особенно сложных научно-технических задач, задач обработки больших объемов данных в реальном времени, поиска оптимальных решений в задачах экономического планирования и при автоматизированном проектировании.

1.2.1 ЭВМ общего назначения

Первые ЭВМ были созданы для выполнения научных и технических расчетов, для которых типичными являются возможность работы со словами фиксированной длины, относительно небольшие объемы входной (исходных данных)и выходной (результатов расчета)информации очень большое количество вычислений, которые необходимо проделать в процессе решения задачи. Совсем иной характер носят задачи учета, статистики и т.п. Эти

задачи связаны с вводом в машину и запоминанием очень большого количества исходных данных. Сама обработка данных требует выполнения сравнительно небольшого числа логических и арифметических операций. По окончании обработки выводится и печатается большое количество информации, причем результаты обработки должны печататься в отредактированной форме в виде таблиц, ведомостей и т.д. Задачи подобного типа получили название задач обработки данных, а ЭВМ, предназначенные для их решения, часто называют системами автоматической обработки данных. Системы автоматической обработки данных составляют основу автоматизированных систем управления (АСУ). Для систем обработки данных характерно наличие большого количества внешних, или периферийных устройств, способных хранить очень большой объем информации и устройств, осуществляющих ввод и вывод данных, их регистрацию, и отображение. Современные ЭВМ общего назначения универсальны, они могут использоваться как для решения научно-технических задач численными методами, так и в режиме автоматической обработки данных в АСУ. Такие машины имеют высокое быстродействие, память большого объема, гибкую систему команд, широкий набор периферийных устройств и способ кодирования информации, учитывающий требования обработки данных.

1.2.2 Малые ЭВМ

Имеется большое число, условно говоря, "малых" применений вычислительных машин, таких, как автоматизация производственного контроля изделий, обработка данных при экспериментах, прием и обработка данных с линий связи, управление технологическими процессами, управление станками и разнообразными цифровыми терминалами (расчерчивающими устройствами и др.), и т.д. Для этих применений ЭВМ общего назначения слишком велики и дороги. Возникла необходимость в небольших, простых, надежных и, главное, дешевых ЭВМ, в которых обеспечиваются простота программирования и наглядность системы программного обеспечения, а также сравнительная простота эксплуатационного обслуживания. Развитие технологии интегральных электронных схем позволило создать машины, удовлетворяющие указанным выше требованиям. Уменьшение объема аппаратуры и стоимости машин достигнуто, в первую очередь, за счет короткого машинного слова (12 - 16 разрядов вместо 32 - 64 в машинах общего назначения), уменьшения по сравнению с ЭВМ общего назначения количества типов обрабатываемых данных (в некоторых моделях только целые числа без знака), ограниченного набора команд, сравнительно небольшого объема оперативной памяти и небольшого набора периферийных устройств. Подобные машины за свои небольшие размеры получили название малых или мини-ЭВМ. Для преодоления трудностей, возникающих при проектировании малых ЭВМ из-за короткого машинного слова, предложен ряд решений по представлению данных, адресации, составу и структуре команд, логической структуре процессора, организа-

ции обмена информацией между устройствами ЭВМ. Первые модели малых ЭВМ имели длину слова 12 разрядов. Впоследствии достижения интегральной микроэлектроники позволили перейти в малых машинах к слову длиной 16 разрядов, что не только повысило точность вычислений и создало возможности для построения более гибкой системы команд, но и обеспечило согласованность форматов данных с ЭВМ общего назначения. Высокое быстродействие позволило малым ЭВМ обслуживать технологические процессы в реальном масштабе времени, а также компенсировать замедление обработки данных, связанное с тем, что многие процедуры обработки при ограниченном объеме аппаратуры, скромном наборе команд и отсутствии специализации машины приходится реализовать не аппаратурными средствами, а соответствующими подпрограммами (например, операции арифметики с плавающей запятой).

1.2.3. Микропроцессоры и МИКРО-ЭВМ

Положительный опыт разработки и применения малых ЭВМ оказал влияние на направления развития интегральной электроники. При переходе от схем малой и средней степени интеграции с большим и сверхбольшим интегральным микросхемам (БИС и СБИС) возникает проблема их применимости. Большую интегральную микросхему, содержащую тысячи логических элементов, не говоря о СБИС с ее десятками тысяч и более элементов, если это не схема памяти, трудно сделать пригодной для широкого круга потребителей. Первоначально считалось, что на основе автоматизированного проектирования будут выпускаться заказные БИС и СБИС, изготавливаемые по индивидуальным требованиям заказчиков. Однако в дальнейшем оказался возможным другой путь - создание на одной или нескольких БИС или СБИС функционально законченного малоразрядного (первоначально на 2-4, а впоследствии на 16-32 разрядов и более) устройства обработки информации, управляемого хранимой в памяти программой. Это устройство (микросхему или несколько образующих его микросхем) называют микропроцессором, так как оно по своим логическим функциям и структуре напоминает упрощенный вариант процессора обычных ЭВМ. Микропроцессоры по быстродействию и возможностям системы команд приближаются к процессорам малых ЭВМ.

Устройство обработки данных, имеющее в своем составе один или несколько микропроцессоров, БИС постоянной и оперативной памяти, БИС управления вводом и выводом и др., называется микро-ЭВМ. Микро-ЭВМ оснащают необходимыми периферийными устройствами (устройствами ввода-вывода, ЗУ на гибких дисках и др.). Электронная аппаратура микро-ЭВМ содержит несколько десятков корпусов БИС и СИС, размещаемых на одной или нескольких съемных платах. В микро-ЭВМ сочетаются высокая скорость выполнения операций в микропроцессоре, повышенная надежность и небольшая стои-

мость. Микропроцессоры и микро-ЭВМ открывают принципиально новые возможности для высокоэффективной автоматизации производственных процессов, научно-исследовательских и проектно-конструкторских работ, обработки информации при планировании и управлении производством на предприятиях во всех отраслях народного хозяйства. Поэтому с полным основанием создание и применение микропроцессоров и микро-ЭВМ оцениваются как одно из важнейших направлений научно-технического прогресса.

1.3. Понятие об архитектуре и структуре ЭВМ

Аппаратные средства вычислительной техники (ВТ) строятся по иерархической схеме (от низших уровней к высшим):

1. Элементарные логические схемы (базовые вентили), в свою очередь построенные на нескольких интегральных транзисторах.
2. Типовые схемотехнические узлы - комбинационные схемы (триггеры, регистры, дешифраторы, одноразрядные и параллельные сумматоры).
3. Функциональные узлы ЭВМ, состоящие из нескольких типовых схем (блок регистров, запоминающее устройство, матричный умножитель, АЛУ и т.д.)
4. Подсистемы ЭВМ (процессор, числовой сопроцессор, контроллер внешнего устройства, система памяти, подсистема ввода-вывода и т.д.)
5. Автономные вычислительные машины.
6. Вычислительные системы, комплексы и сети.

Для описания сложной системы, которую представляет собой вычислительная машина, часто используют 2 понятия: **архитектура** и **структура.**

Под архитектурой понимают абстрактное представление о ВМ с точки зрения программиста. Сюда входит описание программных средств, аппаратных средств и принципов организации вычислительного процесса на аппаратных средствах с помощью программных средств. В отличие от архитектуры структуру ВМ можно определить как совокупность аппаратных средств с указанием основных связей между ними.

Архитектура в более широком смысле определяется как концепция взаимосвязи элементов сложной структуры и включает в себя компоненты логической, физической и программной организации ВМ. В более узком смысле архитектура ВМ - это описание ее системы команд для программиста (архитектура системы команд).

Развернутое описание архитектуры должно включать:

- форму представления программ в ВМ и правила их интерпретации;
- основные форматы представления данных;

- способы адресации данных в программе;
- состав аппаратных средств ВМ и их характеристики;
- соотношение и взаимодействие аппаратных и программных средств.

К классическим архитектурам можно отнести, прежде всего, архитектуру фон-Неймана. Вот некоторые из характерных особенностей этой архитектуры:

1. Программное управление вычислительным процессом путем автоматического извлечения команд программы из памяти и последовательного их выполнения.
2. Общая память для программ и данных.
3. Одинаковое кодирование программ и данных.
4. Использование двоичной системы счисления.
5. Арифметическое устройство на базе двоичного сумматора.
6. Иерархическое построение памяти и др.

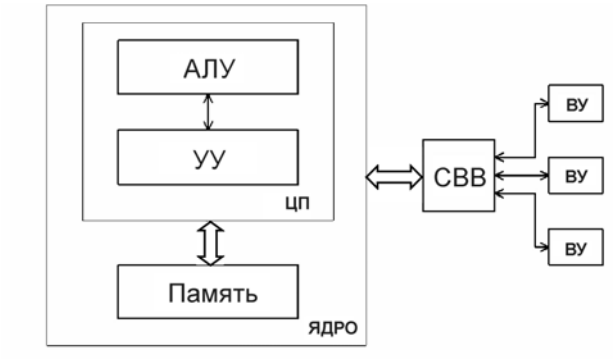


Рис. 1.1

Вычислительная машина фон-Неймана на самом общем уровне имеет структуру, представленную на рис. 1.1 Она состоит из центрального процессора (ЦП), запоминающего устройства, образующих процессорное ядро ВМ, и внешних устройств (ВУ), взаимодействующих с ядром через систему ввода-вывода (СВВ). Сам процессор включает устройство управления (УУ), последовательно извлекающее из общей памяти программ/данных команды и управляющее их исполнением, арифметико-логическое устройство (АЛУ) на базе двоичного сумматора, непосредственно исполняющее последовательности простых арифметических и логических операций под управлением УУ.

Классической архитектурой примерно с середины 70-х годов можно считать и гарвардскую архитектуру. Основным ее отличием от архитектуры фон-Неймана является раздельная память программ и данных. Такая архитектура характерна для управляющих, встраиваемых, специализированных машин, программное обеспечение которых зачастую не меняется с момента изготовления («зашитые» программы), в то время как программа в машине фон-Неймана может даже менять саму себя в процессе работы.

Все остальное многообразие архитектур ВМ можно также отнести к не-фон-неймановским. Многие из них также уже являются почти классическими (как, например, векторные машины). Как правило, основным отличием не-фон-неймановских архитектур

является распараллеливание работы, допускаемое ими. Современные ВМ довольно редко полностью соответствуют требованиям классической архитектуры фон-Неймана, тем не менее, одни из них в целом близки к ней, другие - сильно отличаются от «классической» вычислительной машины.

1.4. Принцип действия ЭВМ

Чтобы лучше понять принцип действия ЭВМ, вспомним, как проводятся вычисления с помощью простейшей настольной счетной машины (калькулятора).

Предварительно на листе бумаги выписываются исходные данные, формулы расчета и вычерчивается таблица для занесения промежуточных и конечных результатов. В процессе вычислений с листа бумаги переносятся на регистр счетной машины числа, участвующие в очередной операции, в соответствии с расчетной формулой включается нужная операция, и полученный результат переписывается с регистра счетной машины в таблицу на листе бумаги.

Таким образом, в этом процессе счетная машина выполняет арифметические операции над числами, которые человек в нее вводит. Лист бумаги служит запоминающим устройством, хранящим программу вычислений (расчетные формулы), исходные данные, промежуточные и конечные результаты. Человек управляет процессом вычислений, в том числе переносит информацию с листа бумаги в машину и обратно, заставляет машину выполнять необходимую операцию, а также выбирает нужный вариант продолжения процесса вычислений в соответствии с результатом, полученным на данном этапе счета.

В рассматриваемом процессе практически ничего не меняет переход от механической счетной машины к использованию электронного арифметического устройства, способного с огромной скоростью производить арифметические операции. Человек будет по-прежнему слишком медленно вводить числа в арифметическое устройство, включать нужную операцию, считывать и переносить на бумагу результат операции.

Эффект, и притом принципиальный, будет достигнут, если к электронному быстродействующему арифметическому устройству добавить быстродействующую память, которая, как лист бумаги при расчете, хранит программу вычислений, исходные данные, промежуточные и конечные результаты, а также быстродействующее управляющее устройство, производящее необходимый для реализации программы вычислений обмен числами между памятью и арифметическим устройством и включающее последнее на выполнение нужной операции.

Если еще позаботиться том, чтобы комплекс нашей аппаратуры имел средства для связи с внешним миром, т.е. устройство для ввода в память данных и программы вычислений, а также устройство вывода результатов вычислений, то придем к классической структурной схеме ЭВМ, изображенной на рис.1.1.

ЭВМ содержит следующие основные устройства:

- арифметическо-логическое устройство,
- память,
- устройство управления,
- устройства ввода данных в машину и вывода из нее результатов расчета,
- пульт ручного управления.

Арифметическо-логическое устройство (АЛУ) производит арифметические и логические преобразования над поступающими в него машинными словами, т.е. кодами определенной длины, представляющими собой числа или другой вид информации.

Память хранит информацию, передаваемую из других устройств, в том числе поступающую машину извне через устройство ввода, и выдает во все другие устройства информацию, необходимую для протекания вычислительного процесса. Память машины в большинстве случаев состоит из двух существенно отличающихся по своим характеристикам частей: быстродействующей основной или оперативной (внутренней) памяти (ОП) и сравнительно медленно действующей, но способной хранить значительно больший объем информации внешней памяти (ВП).

Оперативная память содержит некоторое число ячеек, каждая из которых служит для хранения машинного слова. Ячейки нумеруются, номер ячейки называется адресом.

В запоминающих устройствах, реализующих в ЭВМ функцию памяти, выполняются операции считывания хранимой информации для передачи в другие устройства и записи информации, поступающей из других устройств. При считывании слова из ячейки содержимое последней не меняется и при необходимости слово может быть снова взято из той же ячейки. При записи хранившееся в ячейке слово стирается и его место занимает новое.

Непосредственно в вычислительном процессе участвует только ОП, и лишь после окончания отдельных этапов вычислений из ВП в ОП передается информация, необходимая для следующего этапа решения задачи.

Управляющее устройство (УУ) автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия. В частности, УУ указывает ОП, какие слова должны быть переданы в АЛУ и в другие устройства, включает АЛУ на выполнение нужной операции и помещает полученный результат в ОП.

Алгоритмом решения задачи численным методом называют последовательность арифметических и логических операций, которые надо произвести над исходными данными и промежуточными результатами для получения решения задачи. Поэтому алгоритм можно задать указанием, какие следует произвести операции, в каком порядке и над ка-

кими словами. Описание алгоритма в форме, воспринимаемой ЭВМ, называется программой.

1.5. Аппаратные и программные средства реализации алгоритмов

Вычислительная машина (ВМ) - это искусственная инженерная система для автоматической обработки информации по заданному алгоритму.

Как известно, средства реализации алгоритмов вычислений делятся на аппаратные и программные. Любая вычислительная структура (ВС) это совокупность указанных средств. Их соотношение определяется требованиями к производительности и стоимости ВС.

Аппаратные средства реализуют какие-либо действия алгоритма одномоментно, без возможности дробления со стороны программиста. (Примеры аппаратной реализации: сумматоры, быстрые умножители, устройства для преобразования сигналов в реальном времени и т.д.).

Программные средства - это совокупности инструкций по реализации вычислительного процесса с помощью аппаратных средств в соответствии с алгоритмом. Традиционно под программированием обычно понимают процедурное программирование - задание последовательности действий по реализации алгоритма, причем действия происходят последовательно во времени. В то же время «программировать» решение задачи можно и структурно, пользуясь заданным набором аппаратных средств, в этом случае программирование - это указание путей следования потоков данных от одних аппаратных средств к другим. (Термин «структурное программирование» в литературе по вычислительной технике обычно используется для указания на определенную методологию разработки программного обеспечения, подразумевающую нисходящее проектирование системы, использование только основных управляющих конструкций, отказ от операторов GOTO и т.д. В данном контексте «структурное программирование» означает программирование в пространстве аппаратных структур.) Структурное программирование еще называют «аппаратурно-ориентированным».

Программирование структуры и процедурное программирование не являются взаимоисключающими подходами, как правило, они дополняют друг друга.

При программной реализации алгоритма вычислительный процесс организуется как последовательность процедур, выполняемых поочерёдно во времени на одном операционном устройстве (ОУ). Такое процедурное представление алгоритма удобно оформлять в виде блок-схемы алгоритма. При аппаратурной реализации алгоритма вычислительный процесс разворачивается в пространстве операционных блоков, соединённых между собой в соответствии с потоковым графом алгоритма и работающих параллельно во времени.

На рис. 1.2 показаны два варианта представления алгоритма, а на рис. 1.3 -два варианта его реализации. Очевидно, что во втором случае отпадает необходимость в программной памяти, так как программа вычислений заменяется схемой соединений операционных блоков.

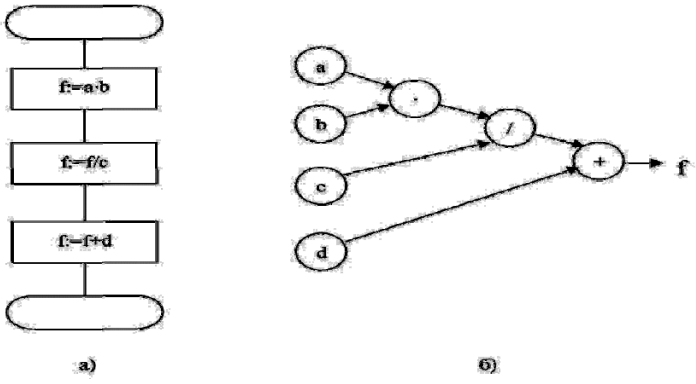


Рис.1.2. Процедурное (а) и потоковое (б) представление алгоритма вычисления значения f = ab/c+d

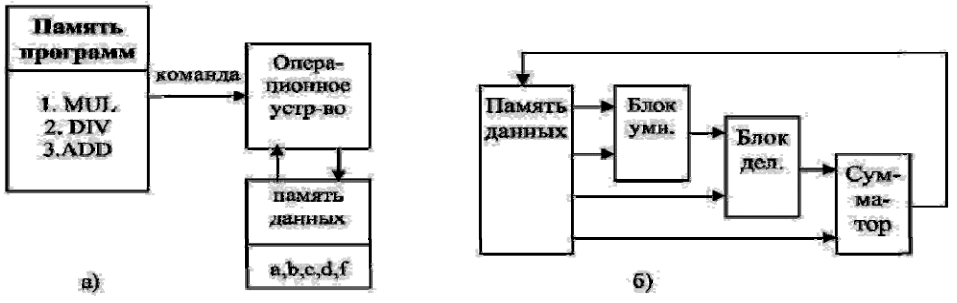


Рис.1.3.Программная (а) и аппаратурная (в) реализации алгоритма f = ab/c+d

Сравнивая рассмотренные варианты, можно отметить, что при программной реализации сложность алгоритма влияет на длину программы, а при аппаратурной реализации - на количество оборудования. Время вычисления одного результата в обоих случаях может быть одинаковым (без учёта времени обращения к памяти). Однако аппаратурная реализация позволяет организовать конвейерную обработку, что существенно повысит её производительность. Это обстоятельство и определяет современные тенденции развития вычислительной техники, учитывая блестящие успехи современной микроэлектроники.

Очень быстрый рост степени интеграции современных микросхем, когда сверхбольшие интегральные схемы (СБИС) могут содержать сотни миллионов транзисторов в одном корпусе, диктует необходимость проектирования мощных аппаратурных средств реализации алгоритмов. Однако разработанные алгоритмы решения прикладных задач, в

том числе и цифровой обработки сигналов, мало ориентированы на аппаратурную реализацию. Необходимы специальные -аппаратурно-ориентированные алгоритмы, синтезированные с учётом требований технологии СБИС. В идеальном случае процесс разработки алгоритма должен быть совмещён с проектированием СБИС, так как топология СБИС изоморфна потоковому графу аппаратурно-реализуемого алгоритма.

Рассмотрим процесс решения задачи на универсальной ЭВМ, включающий ряд этапов, показанных на рис. 1. 4. На начальном этапе задача, возникающая в некоторой прикладной

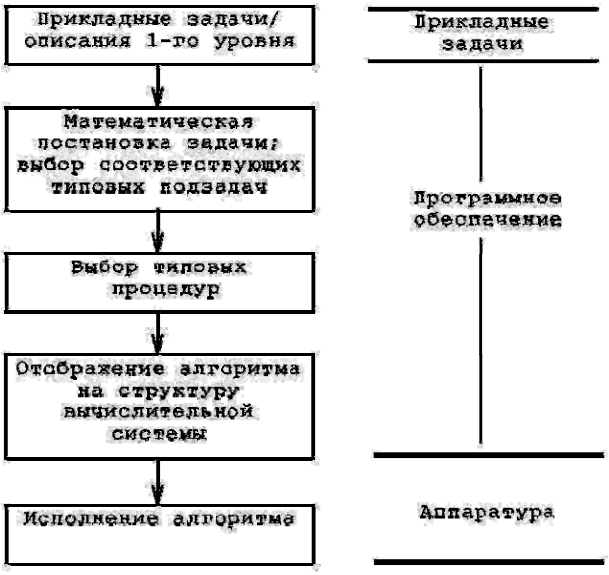


Рис. 1.4.

области, формулируется на естественном языке (составляется описание задачи).Затем осуществляется математическая постановка задачи выбор

соответствующих типовых подзадач. Следующим этапом является выбор типовых вычислительных процедур для реализации необходимых подзадач и отображение алгоритма на структуру вычислительной системы. После этого алгоритм исполняется на имеющейся вычислительной структуре.

логическое устройство (АЛУ) строится, как правило, на основе универсального сумматора. Такое АЛУ выполняет лишь элементарные операции типа сложения, сдвига и некоторые другие. Поэтому для исполнения алгоритма на таком АЛУ необходима программа, состоящая из таких операций.

Таким образом, выбор типовых подзадач, типовых процедур и отображение алгоритма на структуру вычислительной системы осуществляется в универсальной ЭВМ на этапе процедурного программирования.

Время решения задачи на такой ЭВМ прямо зависит от длины программы. Чем крупнее будет математическая функция, выполняемая аппаратурно, тем меньше будет длина программы, меньше обращений к памяти, а, следовательно, меньше время решения и выше производительность ЭВМ. По мере роста возможностей интегральной электроники увеличивается сложность задач, для которых возможна аппаратная реализация их решения, и граница между задачами, реализуемыми аппаратурно и программно, сдвигается. То есть, аппаратурно реализуются уже типовые вычислительные процедуры, крупные математические и даже прикладные алгоритмы. С другой стороны, рост производительности процессоров позволяет решать более сложные задачи, традиционно решавшие-

ся аппаратным способом, программными средствами (примером может служить появление т.н. Winmodem'ов). Таким образом, граница между программными и аппаратными средствами при реализации алгоритмов постоянно перемещается.

2 ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ ЭВМ И СИСТЕМ

2.1 Основные технические характеристики ЭВМ

К числу основных относятся: операционные ресурсы, емкость памяти, быстродействие устройств, надежность, стоимость.

Операционные ресурсы – это перечень действий (операций), которые может делать (выполнять) аппаратура ВК в плане обработки информации (исходных данных). В этот перечень прежде всего включается **система машинных операций** – список F={+,-,*,/,…}. Кроме того, это порождающая ее (систему операций) система машинных команд К={$К_1$, …, $К_N$}. В понятие операционные ресурсы включаются также способы представления информации в ЭВМ, способы представления чисел, текстов, логических значений. Чем шире перечень действий, чем шире многообразие способов представления данных – тем шире операционные ресурсы ЭВМ и, следовательно, возможности ВК в плане обработки информации.

Емкость памяти – очевидная техническая характеристика, которая характеризует вместимость хранилища программ и данных ВК. Единицы измерения – бит, байт В, килобайт КВ = 2^{10}В, мегабайт МВ = 2^{20}В, гигабайт ГВ = 2^{30}В, терабайт ТВ = 2^{40}В. Емкость памяти Е обычно кратна степени 2: $E = 2^m$, m – длина адреса.

Быстродействие – это характеристика, которая отвечает на вопрос, как быстро действует (работает) аппаратура ЭВМ. Эта характеристика определяет потенциальные возможности устройств, указывает на верхнюю границу. Относится к отдельным устройствам, а не ВК в целом. Так, быстродействие АЛУ характеризует скорость, с которой это устройство может выполнять операции: $V_{АЛУ}$={V_+, V_-, V_*, $V_{дел}$, …}. Быстродействие определяется количеством операций в единицу времени и зависит от времени выполнения операции: V=1/t – чем меньше время выполнения операции t, тем выше быстродействие. Быстродействие – это паспортная характеристика, указывается в документе на устройство либо в виде вектора скоростей V, либо в виде набора времен: t_+, t_-, t_*, $t_/$, … Быстродействие процессора определяется временем выполнения команд. Следует отметить, что время выполнения команды $t_к$ зависит от многих факторов – быстродействия памяти (т.к. выборка команды и данных осуществляется из памяти, результаты также засылаются в память), от

быстродействия АЛУ, а также организации ВК. В простейшем случае

$$t_к = t_{вк} + t_{во} + t_{алу} + t_{зр},$$

где первое слагаемое определяет время выборки команды из памяти, второе – время выборки операнда(ов), третье – время выполнения операции в АЛУ, четвертое – время засылки результата операции. Быстродействие процессора принято измерять миллионами операций в секунду или миллионами операций с плавающей запятой в секунду.

Память ЭВМ предназначена для хранения, записи и чтения информации. Быстродействие памяти принято характеризовать количеством операций чтения/записи в единицу времени. Память ЭВМ строится на базе ЗУ (БИС ОЗУ, ППЗУ). Быстродействие памяти зависит от быстродействия ЗУ и ее внутренней организации.

Итак, быстродействие устройств ВК характеризует потенциальные возможности отдельных устройств ВК. Быстродействие же ВК в целом зависит от многих факторов: от быстродействия устройств, внутренней организации самого комплекса, от операционной системы, под управлением которой работает аппаратура, т.е. от организации вычислительных процессов и др. факторов. Поэтому понятие быстродействие на ВК не распространяется. Вместо него используется понятие **производительность ВК**. Назначением ЭВМ является обработка информации, т.е. решение различных задач. Поэтому производительность ВК естественно оценивать количеством задач в единицу времени. Но решаемые задачи разные. Как оценить производительность ВК? Ответ на этот вопрос даёт метрическая теория ВС: процессы в ЭВМ (ВК+ПО) описываются в виде некоторой математической модели, исследование которой и дает все ответы. В качестве такой модели, в частности, используется теория массового обслуживания. Она позволяет получить значения разных характеристик ВС: времени решения задач $T_{реш}$, производительности ВС $\Lambda=1/T_{реш}$, загрузки процессора ρ, времени ожидания в очереди и др. Следует отметить, что время решения конкретной задачи можно приближенно оценить по формуле: $T_{реш} = N(p_1t_1+p_2t_2+\ldots)$. Здесь N – длина программы (количество команд), p_1, p_2, …-вероятности (частоты), а t_1, t_2 …- времена выполнения операций. Было замечено (Гибсоном, в частности), что эти вероятности обладают устойчивостью при решении задач разных классов.

Для сравнения различных ВК по производительности в ВТ обычно используют один и тот же набор программ, который прогоняют на ВК различных типов. Например, т.н. Бенч-Марковские программы и др.

Надежность ВК – это свойство ВК выполнять возложенные на него функции в течение заданного отрезка времени. Надежность ВК отлична от 100% (т. е. от абсолютной). Почему? Дело в том, что элементы, из которых строится ЭВМ, рано или поздно перестают

(отказываются) нормально работать. В результате отказа элемента работоспособность ВК нарушается. Отказы аппаратуры – случайные события, частоту которых принято характеризовать интенсивностью отказов λ, т.е. количеством отказов в единицу времени. Другая характеристика надежности – т. н. наработка на отказ: T=1/λ - это промежуток времени между двумя соседними (по времени) отказами. Для увеличения надежности ВК используется резервирование аппаратуры, например, дублирование – двукратное резервирование, если недостаточно, то трехкратное и т. д.

Стоимость ВК – интегральная характеристика, определяется всеми перечисленными характеристиками. Чем лучше характеристика, тем выше стоимость.

2.2 Общая классификация вычислительных машин

Вычислительные машины можно классифицировать по разным признакам, в том числе: по производительности, назначению, элементной базе и т.д. К основным из них можно отнести, например, следующие:

1. ***По способу представления информации :***

- ВМ непрерывного действия (аналоговые ВМ);
- ВМ дискретного действия (цифровые ВМ) ;
- гибридные ВМ (смешанного типа).

2. ***По назначению (степени специализации):***

- ВМ общего назначения ;
- специализированные и проблемно-ориентированные ВМ.

3. ***По физическому эффекту, используемому для представления ,кодирования и обработки информации:***

- электронные ВМ;
- магнитные ВМ;
- механические ВМ;
- электромеханические;
- криогенные ВМ;
- оптические ВМ;
- пневматические ВМ;
- гидравлические ВМ и др.

4. ***По количеству вычислительных устройств и степени распределенности:***

- автономные ВМ;
- вычислительные системы ;
- вычислительные комплексы ;

- вычислительные сети.

Вычислительная система - сложная совокупность аппаратных средств, в том числе двух и более процессоров, соединенных внутренними шинами и реализующих общие программы вычислений. (Например, параллельные ЭВМ.)

Вычислительный комплекс - совокупность двух и более ЭВМ одного или различных типов, предназначенных для решения общего класса задач и соединенных между собой посредством общей (внешней) памяти (с косвенной связью) или через каналы ввода/вывода (с прямой связью). (Например, две ЭВМ, подсоединенные к одному массиву дисков.)

Вычислительная сеть - множество ЭВМ, соединенных стандартными телекоммуникационными каналами связи или стандартными каналами передачи данных (например, ЛВС).

5. ***По производительности*** можно достаточно условно разделить все ВМ на суперкомпьютеры (ВМ наибольшей производительности на данный момент), большие ВМ (супермини-ВМ), мэйнфреймы, то есть мини-ВМ, высокопроизводительные рабочие станции, микро-ВМ, в том числе - автономные микро-ВМ и встраиваемые ВМ различной производительности, которая может быть сравнительно небольшой. Деление это весьма условно, и границы между машинами одного и другого класса размыты, кроме того, по мере роста общей производительности средств ВТ количественные показатели, отмечающие эти границы, естественно, меняются.

6. ***По сфере применения*** можно выделить ВМ, предназначенные для выполнения научных и инженерных расчетов, управляющие и индустриальные ВМ, встраиваемые ВМ, ВМ, специализированные для обработки сигналов, персональные ВМ и др.

7. Важной характеристикой, непосредственной связанной с определением архитектуры машины, является ***количество процессоров в ВМ.*** Соответственно можно выделить однопроцессорные и многопроцессорные ВМ.

8. ***По способу управления.*** Наряду с традиционными ВМ, управляемыми потоком инструкций (команд), выделился достаточно обширный класс ВМ, управляемых потоком данных (потоковые архитектуры). Классификация не ограничивается приведенными признаками, мы привели только некоторые из возможных, так как многообразие средств ВТ достаточно велико.

2.3 Основные пути повышения производительности ЭВМ

Одной из наиболее важных прикладных характеристик ВМ является ее производительность. Повышение производительности - зачастую главное требование, стоящее перед разработчиками ВТ. Можно выделить несколько основных путей решения этой задачи:

1 Совершенствование технологии производства ЭВМ («физический» путь) - повышение быстродействия логических элементов.

2. Распараллеливание вычислений.

3. Конвейеризация вычислений.

4. Специализация вычислений.

5. Аппаратная реализация сложных функций.

Совершенствование технологии в основном направлено на уменьшение геометрических размеров, снижение потребляемой мощности и уменьшение времени переключения логических вентилей. (В настоящее время геометрические размеры порядка 0.1 мкм, время переключения - порядка 0.1нс). Но - этот путь ограничен физическими пределами.

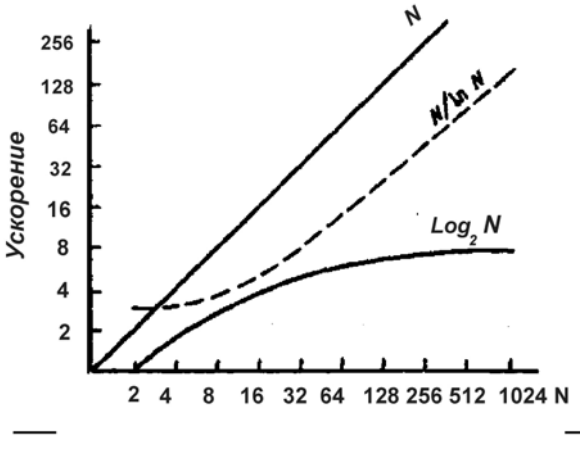


Рис.2.1

Альтернативные пути - архитектурные (логические), и прежде всего - параллельные и конвейерные вычисления. *Параллелизм* можно определить как возможность одновременного выполнения разных математических или служебных операций. *Распараллеливание* - нахождение алгоритма решения задачи, использующего параллелизм и реализация этого алгоритма в ВС.

Параллельные ВС - это многопроцессорные ВС, в которых параллелизм используется для повышения производительности при решении задач за счет одновременного выполнения разных операций на разных процессорах или обрабатывающих устройствах одного процессора..

В идеальном случае для ВС из N процессоров производительность : $P_N = P0N$, где P0 - производительность одного процессора. Однако на практике ситуация зачастую оказывается иной.

В литературе по ВС часто приводят зависимости ускорения в ВС от числа процессоров в системе, приведенные на рис. 2.1. Наиболее оптимистичная оценка соответствует ожиданиям (ускорение в N раз), но встречается только для некоторых ВС (например, систолического типа).Более пессимистические оценки Амдаля (N/(ln N)) и Минского (log2N) объясняются рядом обстоятельств.

В частности, можно выделить три проблемы распараллеливания:

1. Распараллеливание алгоритма (математическая проблема).
2. Распараллеливание вычислительной структуры (архитектурная и схемотех-

ническая, системная проблема)

3. Перенос алгоритма на структуру.

Эти проблемы возникают на разных этапах построения параллельной ВС. Первая проблема связана со спецификой решаемой задачи и выбранного метода решения. Некоторые задачи очень хорошо распараллеливаются (например, многие задачи обработки матриц), некоторые - очень плохо. Вторая - с решением вопросов взаимодействия параллельных процессоров (например, обмен с памятью, межпроцессорный обмен, синхронизация и др.) Третья - с возможностью реализации выбранного распараллеленного алгоритма на данной параллельной структуре.

Конвейеризация вычислений - разбиение вычислений на последовательные этапы с целью реализации этих этапов на отдельных ступенях конвейера для повышения производительности.

Конвейер - устройство, состоящее из N последовательно соединенных частей (ступеней конвейера), каждая из которых выполняет очередной шаг вычислений за время t (такт конвейера). Таким образом, для решения задачи, требующей N шагов, потребуется время, равное $t\,N$, однако производительность конвейера может быть достаточно велика, поскольку освобождающиеся ступени могут заполняться новыми данными. В результате на каждом такте конвейер может выдавать очередной результат.

Время решения одной задачи на конвейере:

$T\,реш = N\,t$.

Производительность конвейера:

$P = N'/(T_0 + Nt);$

где N' - количество задач, поступающих на вход конвейера.

T_0 - время подготовки данных.

Из последней формулы видно, что при $T_0 \ll N\,t$ и $N' \to N$ производительность конвейера стремится к величине $1 / t$ (пиковая производительность конвейера). В то же время, при T_0 сопоставимом с $N\,t$ (большое время загрузки и подготовки конвейера), или при малом N' производительность конвейера будет ниже. Так же, как и в случае распараллеливания, при конвейеризации возникают проблемы создания алгоритма, создания структуры и перенесения алгоритма на структуру.

При этом к алгоритму предъявляют требования:

- отсутствие (минимальное количество) циклов и развилок ;
- возможность разбиения на шаги одинаковой длительности (и сложности);

- последовательное и равномерное перемещение данных по алгоритму и др.

Часто конвейерный режим используют в векторных ВС. Векторные ЭВМ и ВС выполняют не только скалярные операции (над числами), но и операции над векторами (массивами чисел), при этом векторные операции выполняются параллельно для всех элементов вектора, либо - на конвейере. В последнем случае производительность повышается за счет одновременного выполнения на конвейере нескольких векторных операций :

Ускорение = Тск (m-1) Q / (mt + Тп),

где Тск - время скалярной операции, m- длина вектора, t - такт конвейера, Q - количество одновременно выполняемых векторных операций на конвейере (количество задач), Тп - время подготовки конвейера (заполнение конвейера).

Специализация вычислений позволяет повысить производительность за счет аппаратной реализации решения задачи в целом, либо какой-либо сложной части этой задачи, то есть методом программирования аппаратной структуры. Выигрыш в производительности достигается за счет сокращения расходов на управление, а также за счет снижения универсальности устройства.

В то же время, *аппаратная реализация сложных функций* (арифметических, матричных операций и т.д.) может применяться и в универсальных ВС без снижения функциональности, за счет только увеличения стоимости устройств и придания им дополнительных функций.

В дальнейшем мы увидим примеры применения этих подходов в различных ВС.

3. ОРГАНИЗАЦИЯ СИСТЕМ ПАМЯТИ

3.1. Характеристики и классификация запоминающих устройств. Иерархия систем памяти

Под *запоминающими устройствами* (ЗУ, память) будем понимать совокупность устройств для запоминания, хранения и выдачи информации. Память является одним из основных ресурсов компьютера, влияющим как на производительность, так и на функциональность вычислительной машины.

К основным характеристикам устройств памяти можно отнести:

1) Временные характеристики :

- ***быстродействие*** - определяется временем выборки, временем обращения и другими параметрами. Время обращения складывается из различных составляющих, например:

$$t_{обрЧТ} = t_{дост} + t_{чт} + t_{рег},$$

где $t_{обрчт}$ - время обращения при чтении, $t_{дост}$ - время доступа к данным, $tp_{ег}$ -время регенерации (для динамической памяти), *Ь,,* - время собственно чтения;

$$t_{обрЗП} = t_{дост} + t_{подг} + t_{зп}$$

где $t_{обрЗП}$ - время обращения при записи, $t_{подг}$ - время подготовки данных, $t_{зп}$- время собственно записи. Таким образом, процесс чтения/записи ЗУ в общем случае включает ряд этапов разной сложности и длительности.

- ***производительность*** - определяется пропускной способностью ЗУ, то есть - объемом информации, который можно считать/записать из/в ЗУ в единицу времени. Для оценки производительности часто используют показатель длительности цикла обращения к памяти $t_{ц}$, под которым понимают минимальное время между сменой информации на выходе/ входе ЗУ. Длительность цикла не всегда совпадает с временем обращения, в частности, при конвейеризации ЗУ можно добиться увеличения производительности при достаточно большой величине $t_{обр}$ за счет разделения общей задачи чтения/записи на последовательные ступени конвейера.

2) Важнейшей потребительской характеристикой ЗУ является его ***объем,*** или ***емкость памяти*** *(Е),* то есть количество запоминаемой информации. В зависимости от типа ЗУ, его места в вычислительной системе, объем может меняться от десятков байт (для регистровой памяти ЦП) до десятков и сотен гигабайт (для массивов накопителей на магнитных дисках).

Наряду с характеристикой емкости памяти применяют также удельную емкость по отношению к единице площади или объема кристалла. Такая характеристика в большей степени характеризует технологические особенности ЗУ.

3) Третьей важнейшей потребительской характеристикой ЗУ, как и любого вычислительного устройства, является его стоимость, которая также может меняться в самых широких пределах в зависимости от объема, производительности и других характеристик. Распространенной характеристикой является удельная стоимость в расчете на единицу информации (стоимость одного бита/байта, кило- и мегабайта и т.д.)

Помимо перечисленных можно отметить множество других характеристик ЗУ, в том числе: технологию изготовления, потребность во внешнем источнике питания для хранения информации, длительность хранения, количество циклов чтения и записи, геометрические размеры, и так далее.

С учетом приведенных характеристик, а также - назначения ЗУ, места, занимаемого ЗУ в вычислительной системе, можно привести, например, следующую классификацию ЗУ:

1. По удаленности от процессора :

- сверхоперативная (регистры процессора, КЭШ память);

- основная (оперативная) память ;
- дополнительная память (внешняя) ;
- вторичная память (также внешняя) ;
- массовая память (внешняя, как правило, на доступных сменных носителях).

2. По организации записи :

- постоянное запоминающее устройство - ПЗУ (ROM - read-only memory) - однократно программируемое изготовителем устройство только для чтения;
- перепрограммируемое запоминающее устройство - ППЗУ (PROM) -возможно перепрограммирование, которое, однако, требует специальной процедуры, кол-во циклов записи намного меньше циклов чтения;
- оперативное запоминающее устройство - ОЗУ (RAM - random access memory) - количество циклов чтения может совпадать с количеством циклов записи.

Строго говоря, приведенные отечественные и импортные сокращения для двух основных типов памяти не вполне точно отражают приведенное деление памяти по организации записи, но являются исторически сложившимися и общепринятыми.

3. По организации доступа :

- с последовательным доступом ($t_{дост}$ меняется для различных адресов или участков памяти - чем старше адрес, тем больше время доступа);
- с прямым доступом ($t_{дост}$ = const для различных адресов или участков памяти).

4. По организации поиска ячеек в памяти:

- «М-поиск» - поиск по месту (например, в адресном ОЗУ);
- «В-поиск» - поиск по времени (например, при работе с накопителем на магнитной ленте).

5. По физическому эффекту (технологии), используемому для запоминания и хранения информации :

- полупроводниковая память;
- магнитная;
- магнитооптическая;
- оптическая;
- электростатическая и др.

6. ОЗУ по способу хранения делится на :

- статическое (на триггерах);
- динамическое (на конденсаторах).

7. По способу адресации:

- адресная память;
- стековая память;

- ассоциативная память.

8. По организации памяти в систему:

- память с расслоением;
- виртуальная память;
- кэш-память;
- различные варианты блочно-конвейерных систем.

9. По зависимости от источника питания:

- энергозависимая;
- энергонезависимая.

Как и ранее, при классификации вычислительных машин, отметим, что выбранные классификационные признаки не являются всеобъемлющими или обязательными, просто они отражают некоторые важные особенности классифицируемых систем.

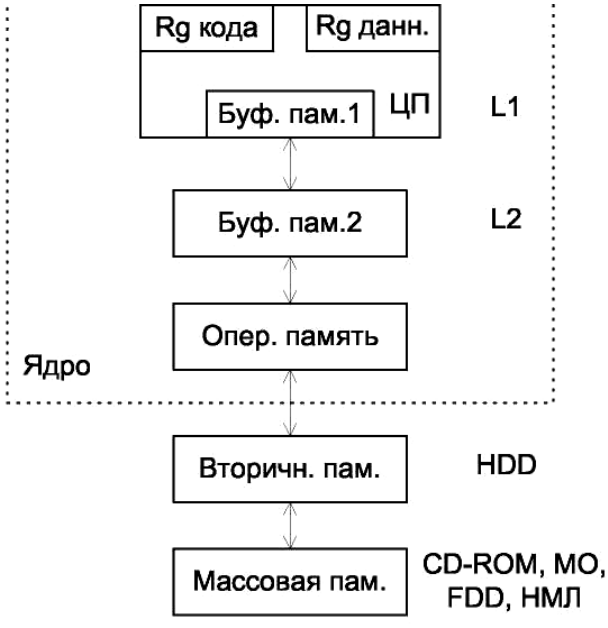


Рис. 3.1

Рассматривая характеристики и классификацию ЗУ, с учетом их многообразия нельзя не упомянуть об иерархии систем памяти в составе вычислительной системы. Как мы помним, принцип иерархического построения систем памяти заложен еще в фон-неймановской архитектуре, в те годы, когда большинства современных ЗУ и их типов не существовало. Однако и тогда существовала относительно быстрая и дорогая энергозависимая оперативная память, и внешняя память – более дешевая, намного более медленная, но при этом энергонезависимая. Сейчас иерархия выглядит намного сложнее, но общий принцип ее построения остается в основном неизменным (Рис.2.1).

На верхнем уровне иерархии располагается наиболее быстрая и дорогая регистровая память процессора, а также - буферная кэш-память первого уровня, расположенная в кристалле процессора. К ней примыкает кэш-память второго уровня, выполняемая в одном корпусе с процессором, либо - на системной плате. На следующем уровне находится оперативная (чаще всего - динамическая) память достаточно большого объема. Эти уровни вместе с процессорами образуют ядро ВС в архитектуре фон-Неймана. На более низких

уровнях располагается внешняя память - внешние устройства, взаимодействующие с ядром по каналам ввода-вывода. В качестве вторичной памяти можно указать НЖМД (HDD) - накопители на жестких магнитных дисках - пожалуй, наиболее быстродействующую внешнюю память, при этом со значительным объемом. К массовой памяти можно отнести разнообразные сменные носители информации, различающиеся как по объему, так и по времени доступа (накопители на гибких магнитных дисках, магнитной ленте, CD-ROM - диски и т.д.), которые объединяет, пожалуй, относительно низкая удельная стоимость.

Легко заметить, что при движении по иерархии сверху вниз происходит снижение удельной стоимости хранения информации, рост объемов ЗУ и -падение производительности.

Подобное построение систем памяти в ВС объясняется, с одной стороны, различной функциональной направленностью ЗУ (оперативное хранение небольших объемов информации в ОЗУ, либо - долговременное хранение больших объемов данных на дисковой памяти), а с другой - попыткой достичь более-менее приемлемого соотношения между ценой и производительностью (а также функциональностью) вычислительной системы, что являлось актуальным как на заре вычислительной техники, так и сейчас.

3.2. Организация адресной памяти

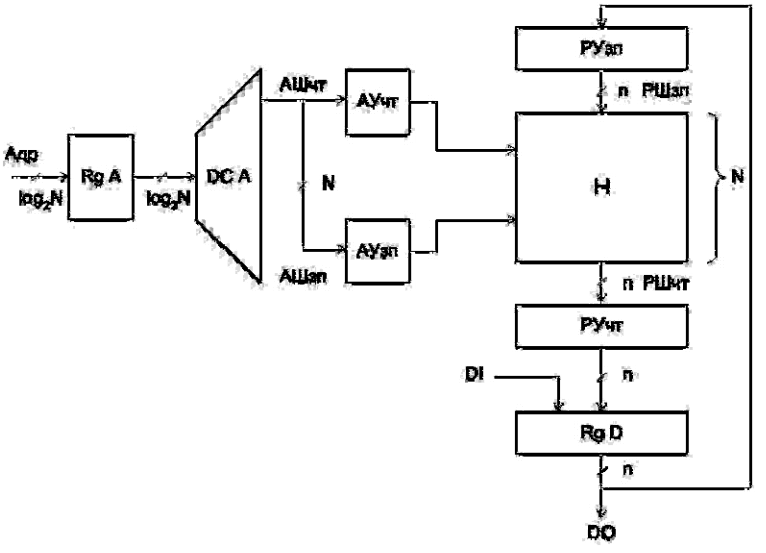
Отличительным признаком адресной памяти является организация доступа к ячейкам памяти по адресам, то есть - по номерам, которые поступают на вход ЗУ в закодированном виде, затем - декодируются тем или иным образом для выбора определенного запоминающего элемента (ЗЭ) или их группы. Подобная схема соответствует в большей степени устройствам с М-поиском, для которых время доступа является постоянной величиной, не зависящей от адреса.

Адресная память с М-поиском (под которой чаще всего подразумевают полупроводниковую память) на самом общем уровне включает в себя массив запоминающих элементов (триггеров, регистров, управляемых конденсаторов и т.д.), адресные дешифраторы для декодирования адреса ячейки в управляющие импульсы по шинам управления, усилители адресных и разрядных линий, а также все остальные необходимые логические схемы для осуществления выборки, считывания и записи и управления ЗУ. Различные варианты организации памяти с М-поиском связаны, прежде всего, с различными способами построения массива ЗЭ и декодирования адреса. С этой точки зрения выделяют память типа 1D, 2D, 2,5D, 3D, 4D - по количеству измерений массива ЗЭ. В памяти типа 1D массив имеет 1 измерение, то есть адресуется каждый бит памяти. При достаточно большом объеме ЗУ это приводит к сложным схемам дешифраторов и огромному количеству служеб-

ных линий, трассировка которых внутри кристалла вызывает проблемы, а площадь, занимаемая ими, сопоставима с площадью массива самих ЗЭ. В 2D-памяти (рис. 3.2) адресуются не отдельные биты, а слова, что улучшает общую картину.

Рис. 3.2

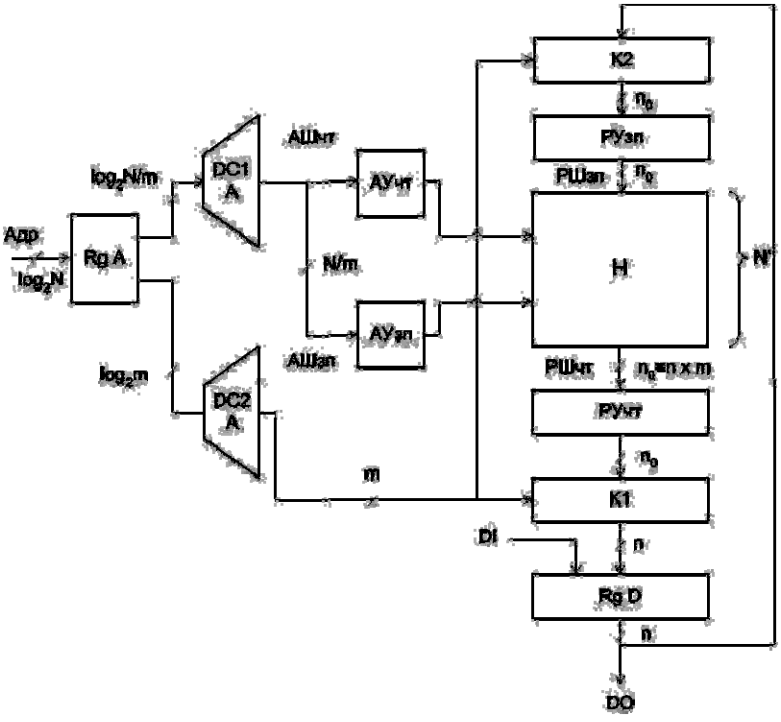
Для большей экономии кристалла необходимо использовать слова еще большей разрядности, что входит в противоречие с разрядностью шин данных ВС, и создает дополнительные неудобства. Для их преодоления используют организацию типа 2,5D (рис.



3.3), при которой слова системной разрядности (16, 32, 64 и т.д.) объединяются в группы, адрес ячейки при декодировании в ЗУ делится на 2 части, большая из них используется для выбора группы, а меньшая - для выбора слова внутри группы.

При 3D организации (рис. 3.4) массив ЗЭ имеет два измерения, то есть выбор ячейки (слова) осуществляется по двум координатам, при этом адрес ячейки делится на две равные части, каждая из которых используется для выбора одной из линий по одной из двух координат. В результате и количество линий, и сложность адресных дешифраторов

уменьшается. Дальнейшее развитие такого подхода приводит к памяти с организацией 4D и т.д.



Ри
с. 3.3.

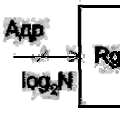


Рис. 3.4.

Память с В-поиском также использует адресный способ выборки ячейки (блока) информации, однако в таких устройствах адрес выступает не как источник кода для дешифратора, а скорее - как инициализирующее значение для счетчика, который отсчитывает количество последовательно считанных, либо -просмотренных блоков. При обнулении счетчика последний блок записывается/считывается из массива накопителя. Физически к памяти такого типа можно отнести дисковую память (с определенной долей условности, если рассматривать подсистему «головка чтения/записи - дорожка»), а также, в более явном виде - накопители на магнитной ленте.

ЗУ с В-поиском в процессе подсчета блоков могут использовать внешнюю синхронизацию, либо - внутреннюю, то есть быть самосинхронизируемыми. В последнем случае синхронизация осуществляется с помощью адресных меток, которыми снабжен каждый блок данных, и которые подсчитываются устройством при выполнении последовательного доступа.

3.3. Безадресная стековая память

В стековой памяти (памяти магазинного типа, организованной по принципу «Последним вошел - первым вышел» - LIFO - "Last In - First Out") все операции чтения и записи

осуществляются относительно указателя стека (SP-stack pointer). Указатель стека указывает на ячейку памяти, содержащую последнее внесенное в стек слово. Стековая память может организовываться программно-аппаратным или аппаратным способом. Команды обращения к стеку не содержат адресной части, либо эта часть является относительной величиной, прибавляемой к указателю. Это позволяет сократить длину программы, так как нет необходимости указывать достаточно длинные адреса, а также - упростить схему ЗУ при аппаратной реализации стека.

В то же время при работе со стековой памятью приходится осуществлять фактически последовательный доступ, кроме того, может происходить т.н. переполнение стека - при попытке записать в полностью заполненный стек очередное значение, либо при считывании из пустого стека.

Использование стековой памяти будет более эффективным, если процессор, работающий со стеком, будет поддерживать специальные стековые команды - не только «занести в стек» и «считать из стека», но и такие, как -«сложить два числа на вершине стека», «переставить элементы стека» и т.д. Такие команды часто используются в RISC-процессорах, в микроконтроллерах, управляющих ЭВМ.

3.4. Ассоциативная память

Под ассоциативной памятью (АП) подразумевают вариант организации памяти, при котором адресная информация, используемая для выборки слова из памяти, содержится в самих словах памяти. Чтение/запись осуществляется для тех слов, адресная часть которых (так называемый «тэг») полностью или частично совпадает с заданной. Ассоциативная память может быть организована как программным, так и аппаратным путем. При программной реализации понятие АП используется в основном как модель взаимодействия программы (процессора) с источником данных. Например, в реляционных базах данных для ускорения поиска нужной информации широко используются т.н. ключевые поля, которые входят в состав каждой записи БД. Для быстрого поиска по ключам используют специальные индексные файлы, построенные, например, по принципу двоичных деревьев. Адресной информацией в данном случае является не номер записи, а содержимое, например, поля кода товара, или - фамилии человека. Индексные файлы же позволяют укорить процедуру поиска.

При аппаратной организации АП большую роль играют, во-первых, аппаратные средства поиска, различные быстродействующие компараторы (схемы сравнения), а во-вторых- вариант организации поиска. В частности, в АП часто используется принцип «вертикальной» обработки и разрядных срезов (рис. 3.5). При обычной «горизонтальной» обработке (рис. 3.5а) для отыскания нужного слова в массиве

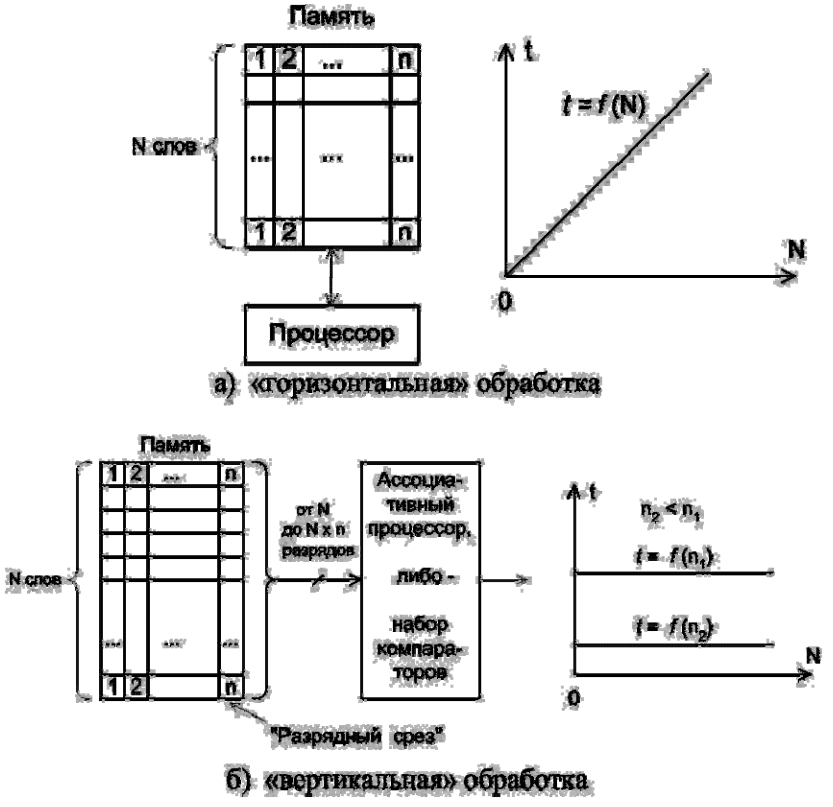


Рис 3.5

ячеек слова просматриваются последовательно, по адресам, то есть как бы горизонтально, если представить себе массив ячеек как вертикальный столбец. При вертикальной обработке (рис. 3.5б) все слова просматриваются одновременно. При этом, если осуществлять сравнение искомого тэга со всеми разрядами всех тэгов слишком накладно, то используются вертикальные разрядные срезы (РС) всех слов накопителя. После первого сравнения отсекаются все слова, имеющие первый бит, несовпадающий с заданным тэгом, затем анализируется следующий РС и т.д.

Таким образом, отличительные особенности АП:

1. Операции в памяти выполняются не над определенной ячейкой. а относятся сразу к группе или ко всем элементам.

2. Основной операцией в АП является операция поиска или сравнения.

3. Время поиска в АП может не зависеть от числа ячеек в памяти. При аппаратной организации АП выделяют 4 варианта :

1. Память с полным параллельным доступом (осуществляется параллельное сравнение всех тэгов с заданным по всем разрядам) - самый высокопроизводительный и самый дорогой вариант.

2. Память с последовательной обработкой разрядных срезов (РС). Время поиска (доступа) в такой памяти пропорционально разрядности тэгов.

3. Память с последовательной обработкой слов («горизонтальная обработка») - время поиска пропорционально числу слов в памяти. Фактически этот вариант только условно можно отнести к АП, и то в случае, когда сравнение каждого тэга с заданным осуществляется аппаратным способом.

4. Частично-ассоциативная память. Компромиссный вариант, в котором выделяются несколько групп слов (блоков слов), в каждой из которых производится последовательный поиск, но все группы обрабатываются параллельно, либо - наоборот, группы обрабатываются последовательно, а внутри группы ведется полностью ассоциативный поиск, или поиск по срезам.

На рис. 3.6 приведен пример структура блока ассоциативной памяти. На рисунке использованы следующие обозначения :

RgАП - регистр адресного признака,

RgM - регистр маски,

RgD - регистр данных,

КС - комбинационная схема,

RgC - регистр совпадений,

ФС - формирователь сигналов (1 - нет совпадений; 2 - одно совпадение; 2 - более одного совпадения),

Н - накопитель;

N - количество слов в устройстве памяти (накопителе);

n - количество адресных разрядов в слове (тэге);

n - ый разряд используется для указания занятости ячейки; m - разрядность собственно информационной части слова, не используемой для адресации. Маска используется для выделения тех разрядов, которые должны участвовать в сравнении.

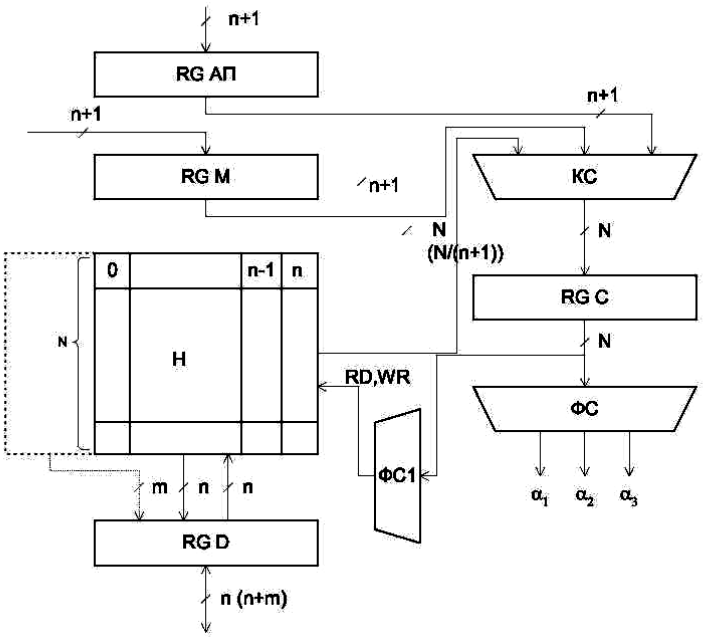


Рис. 3.6

Ассоциативная память применяется в основном в ВС, в которых решаются задачи распознавания образов, необходим быстрый поиск информации (например - в системах с аппаратной поддержкой БД).

Также АП применяется в системах виртуальной памяти и кэш-памяти для определения необходимости подкачки страниц и для поиска страниц, подлежащих замене.

3.5. Системы памяти с расслоением

Принцип организации систем памяти с расслоением рассчитан на повышение быстродействия устройств памяти, состоящих из нескольких медленных устройств, за счет распределения адресного пространства между этими устройствами. (Напомним, что адресное пространство - это количество независимо адресуемых ячеек памяти). Адресное пространство делится таким образом, что соседние по адресам ячейки располагаются в разных физических устройствах. Логический адрес ячейки состоит из физического адреса внутри устройства (блока) и номера блока.

Расслоение памяти осуществляют двумя основными способами.

1. Повышение производительности памяти за счет одновременного считывания/записи соседних ячеек памяти из разных физических устройств по общей шине данных. Производительность увеличивается за счет параллельного подключения устройств и их одновременной работы на общую шину данных. Недостатком такого подхода являет-

ся необходимость использования широкой шины данных, часто - превышающей по ширине разрядность слов, используемых в системе.

2. Использование конвейера памяти. Конвейер строится из нескольких медленных устройств с большим временем доступа. Если система может обратиться с небольшой задержкой ($t_ц$) к нескольким медленным устройствам, то каждое из них начнет процесс выборки, и к тому моменту, как система закончит обращение к последнему устройству, первое уже выдаст считываемые данные на шину данных, затем - второе и т.д. В результате реальная производительность системы будет определяться не большим $t_{дост}$, а маленькой величиной $t_ц$.(см. Рис. 3.7)

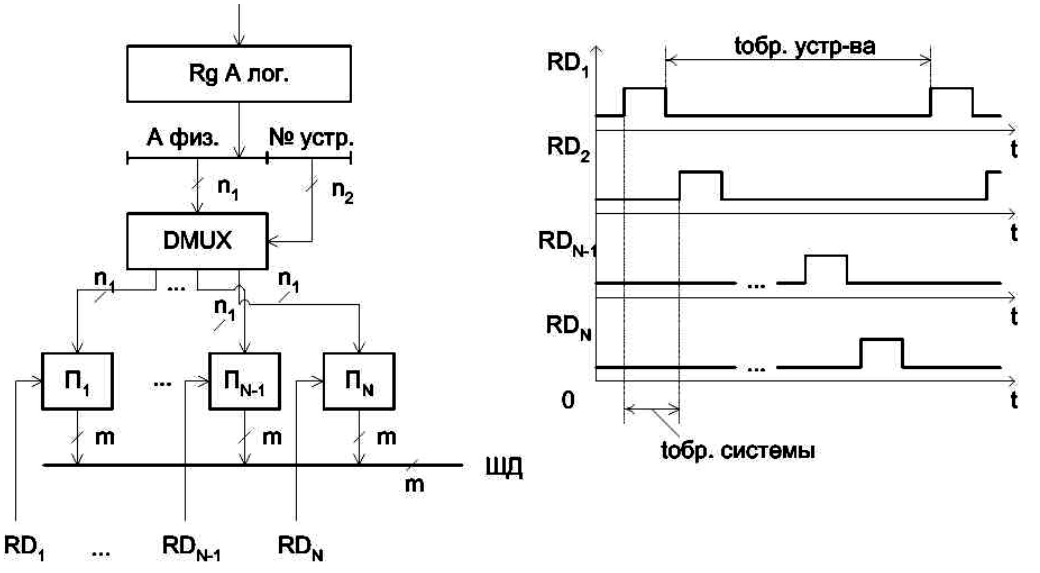


Рис. 3.7

Системы расслоения памяти применялись и применяются в различных ВС, в частности - в свое время в ПК типа IBM PC/XT, в супер-ЭВМ Cray -1,2, в других векторных процессорах. Похожие принципы конвейеризации работы памяти применяются в современных устройствах синхронной динамической памяти (SDRAM).

3.6. Понятие о виртуальной памяти

Виртуальная (от virtual - "кажущийся") память (ВП) - это система организации памяти, при которой процессору (программе) предоставляется адресное пространство, превышающее физическое адресное пространство ОЗУ системы за счет внешней памяти. Задачей построения ВП является сведение к минимуму потерь производительности при вынужденном обращении к внешней памяти.

ВП может быть организована программно, программно - аппаратно и аппаратно. Как правило, в современных ВС программно-аппаратная организация ВП заключается в использовании операционной системой аппаратной поддержки ВП, заложенной в процессорах общего назначения.

ВП может иметь страничную, сегментную или странично – сегментную организацию. При страничной организации память представляется совокупностью страниц фиксированной длины (2-16 Кбайт). При сегментной организации память представляет собой набор сегментов, то есть логически связанных блоков памяти различного размера.

Для виртуальной памяти большое значение имеет алгоритм подкачки, то есть способ замены страниц в ОЗУ на страницы во внешней памяти, к которым произошло обращение. При аппаратной организации ВП система подкачки использует ассоциативную память страниц. Стратегии замены страниц в ВП могут быть самыми различными:

1. Наиболее давнее использование (по времени)
2. Наиболее редкое использование.
3. По очереди (по принципу FIFO)
4. Случайным образом.
5. "Наилучший" выбор - гибкое сочетание различных стратегий.

2.7. Варианты организации КЭШ-памяти

Обособленным вариантом ВП можно считать т.н. кэш-память (от фр. «cache» - скрывать). Это вариант организации системы памяти, предназначенный для ускорения обмена между процессором и оперативной памятью. С виртуальной памятью кэш-память роднит общий принцип - ускорение за счет размещения наиболее активно используемых данных и кода в более быстрой памяти, но между ВП и кэш-памятью существует также множество различий, которые можно проиллюстрировать следующей таблицей:

Сравнение виртуальной и кэш-памяти.

Таблица 3.1

Виртуальная память	Кэш-память
1 Организуется для ускорения обмена между процессором и внешней памятью (ОЗУ и ВнП) 2. Обмен страницами по 2-16Кб 3. Ускорение до 1000 раз 4. При подкачке ЦП может переклю-	1. Организуется для ускорения обмена между ЦП и ОЗУ 2. Обмен строками (сотни байт) 3. Ускорение до 10 раз 4. При подкачке ЦП ожидает ее завершения

чаться на другую задачу 5. Адресное пространство ВП равно сумме адресного пространства ОЗУ и ВнП 6. В ОЗУ хранятся копии или оригиналы страниц ВП 7. ВП. программно. доступна	5. Адресное пространство кэш-памяти равно адресному пространству ОЗУ 6. В буферной памяти хранятся копии строк ОЗУ 7. Кэш-память программно недоступна.

(Можно заметить, что под кэш-памятью иногда понимают не систему организации памяти, а саму буферную память (БП), используемую для ускорения обмена процессора с ОЗУ.)

Небольшое значение ускорения из-за использования кэш-памяти по сравнению со значительным ускорением при использовании виртуальной памяти можно объяснить большой разницей между временем доступа к дисковой памяти (10-ки микросекунд) и оперативной (10-ки наносекунд), и сравнительно небольшой - между временем доступа к оперативной памяти и к буферной памяти (наносекунды). Заметим, что буферная память в составе кэш-памяти обычно строится на базе быстродействующего статического ОЗУ на триггерах.

Системы кэш-памяти можно классифицировать следующим образом:

1. По способу отображения строк основной памяти на строки буферной памяти:

- полностью ассоциативная кэш-память (любая строка основной памяти может размещаться в любой строке буферной памяти - самый дорогой и самый производительный вариант);
- кэш-память с прямым отображением (каждая строка основной памяти может размещаться только в одной определенной строке основной памяти - самый простой и наименее производительный вариант);
- частично - ассоциативная или множественно-ассоциативная кэш-память (компромиссный вариант, при котором основная память делится на множества строк, каждое множество отображается на группу строк в буферной памяти, при этом внутри группы действует принцип полной ассоциативности; при количестве групп = 1 получаем полностью ассоциативную кэш-память, при количестве групп = количеству строк -кэш-память с прямым отображением).

2. По способу переноса информации из кэш-памяти в основную (т.н. «своппинг»):

- простой своппинг (Write Through - когда информация, записанная процессором в кэш-память, переносится в основную только при необходимости замены строки);

- сквозной своппинг (Write Back - когда информация записанная процессором в кэш-память, одновременно переносится в основную, то есть кэш работает только на чтение; этот вариант менее производительный, но более надежный).

Рассмотрим подробнее варианты отображения строк основной оперативной памяти на буферную память на примере условной системы памяти с 16-ю строками в основной памяти (ОП) и 4 строками буферной памяти (БП). Такое небольшое количество строк выбрано для простоты изложения.

1. ***Полностью ассоциативная кэш-память*** (кэш-память с произвольным отображением). При таком варианте построения кэш-памяти в любой строке БП может располагаться любая строка из ОП (рис. 3.8).

На рис. 3.8 : RGАлог -регистр логического адреса, RG D - регистр данных, хранящий всю строку из БП.

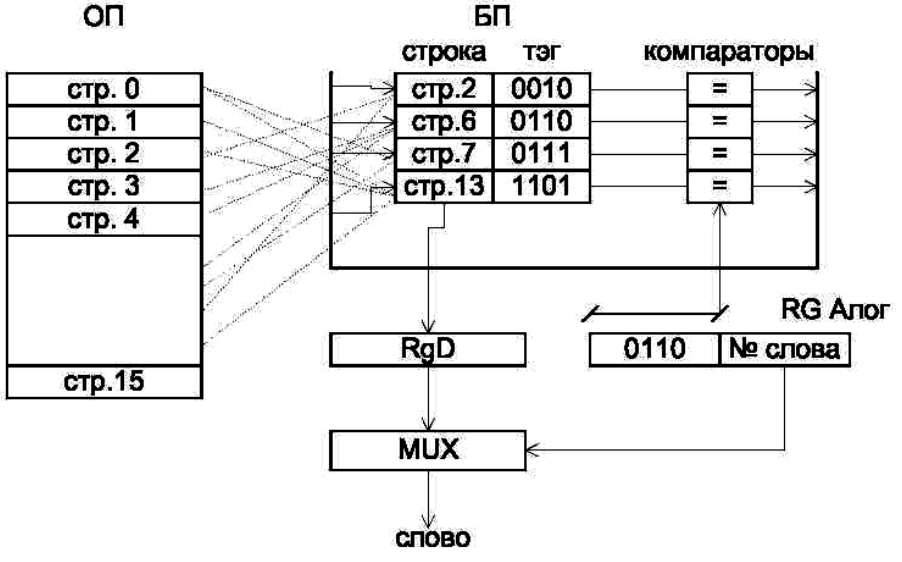


Рис. 3.8

Производительность системы с кэш-памятью, или величина ускорения при использовании кэш-памяти зависит в том числе от вероятности попадания искомой строки в БП :

$$P_{кп} = f(P_{hit}),$$

(P_{hit} - вероятность попадания, P_{miss} - вероятность промаха).

В свою очередь, вероятность попадания зависит как от объема буферной памяти (или - от соотношения объема БП и ОП), так и от количества комбинаций раз-

личных строк ОП, которые могут размещаться в БП. Для рассматриваемого варианта отображения такое количество комбинаций равно:

$$K1 = 2^N!/(2^N-2^n)!,$$

где 2^N - количество строк ОП, 2^n - количество строк БП (в данном случае N = 4,n=2, 2^N=16,2^n = 4).

2. ***Кэш-память с прямым отображением.*** При таком варианте построения кэш-памяти любая строка из ОП может располагаться только в одной конкретной строке БП (рис. 3.9).

Такой вариант является самым дешевым, но и самым медленным вариантом реализации, поскольку количество комбинаций различных строк ОП, которые могут размещаться в БП, существенно меньше, чем для полностью ассоциативной КП:

$$K_2 = 2^N.$$

3. ***Множественно-ассоциативная*** (частично-ассоциативная, ассоциативная по множеству) кэш-память. При таком варианте построения кэш-памяти (рис.3.10) все множество строк основной памяти разбивается на несколько подмножеств (количество которых равно 2^s). Каждое подмножество отображается на группу строк в буферной памяти, внутри группы действует принцип полной ассоциативности, то есть любая строка из данного подмножества может располагаться в любой строке данной группы.

Такой вариант является промежуточным, компромиссным вариантом между полностью ассоциативной кэш-памятью и кэш-памятью с прямым отображением. Количество комбинаций:

$K_3 = 2^n(2^{N-S}!)/(2^{N-S}-2^{n-S})!$,

При s=0 получаем полностью ассоциативную кэш-память (единственное подмножество), при s=n получаем вариант кэш-памяти с прямым отображением (количество подмножеств равно количеству строк в БП).

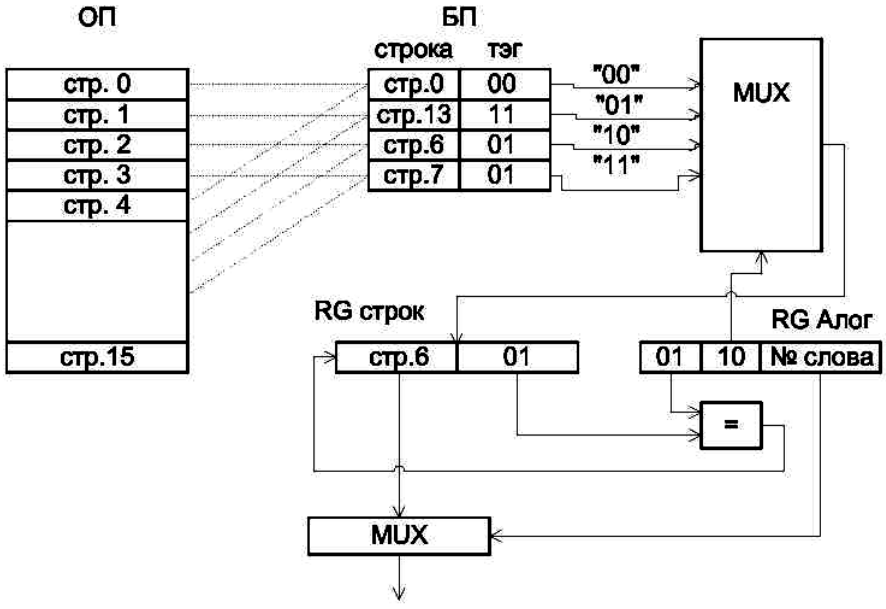


Рис.3.9

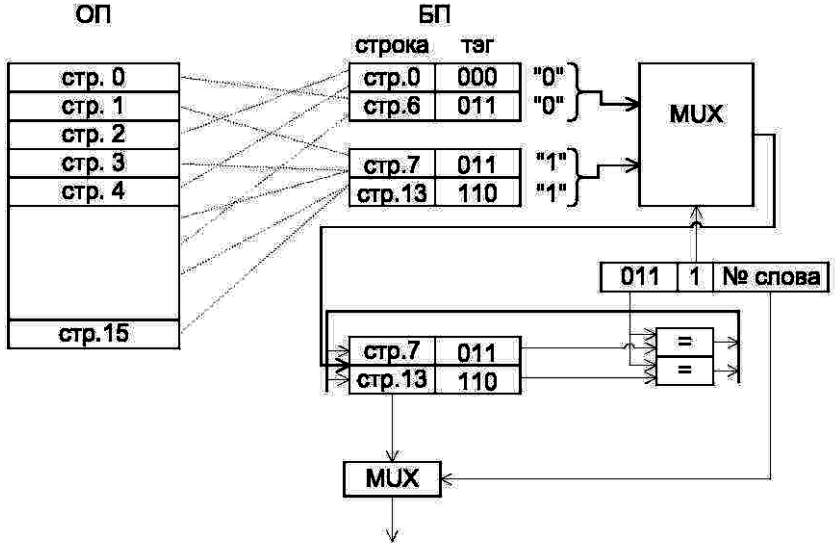


Рис 3.10

В общем случае на ускорение кэш-памяти, которого она позволяет достичь, влияет ряд параметров:

- прежде всего, - размер кэш-памяти;
- способ отображения строк памяти;
- соотношение быстродействия устройств ОЗУ и буферной памяти;
- вариант свопинга.

Первые три параметра влияют на вероятность попадания слова в кэшпамять (т.н. "cache hit", при отсутствии попадания происходит кэш-промах -"cache miss", приводящий к необходимости подкачки из основной памяти), которая непосредственно влияет на ускорение в системе с кэш-памятью.

Эффективное время обращения к кэш-памяти:

t обрКП = tПАП + P tобрБП + (1-P) (tПАП + tобрБП + 2 t_oбрОП)

где tПАП − время поиска адресного признака ;

P − вероятность попадания в кэш ; tобрБП − время обращения к буферной памяти; t_oбрОП − время обращения к основной оперативной памяти.

В многопроцессорных системах с общей (разделяемой) памятью, в которых используется локальная для каждого процессорная кэш-память (буферная память), возникает проблема обеспечения непротиворечивого соответствия информации в разделяемой ОП и локальных копиях строк ОП в различных локальных блоках БП, известная как *проблема когерентности кэшей.*

В общем случае проблема сводится к тому, что запись одним процессором информации в свою буферную память не сразу приводит к изменению соответствующей ячейки в ОП, и, соответственно, другие процессоры, обращающиеся к этой ячейке ОП, либо - к ее копиям в своих модулях БП, видят «старую» информацию. В особенности это влияет на системы, использующие систему «почтовых ящиков» (ячеек ОП) для обмена заданиями между процессорами.

В таких системах могут использоваться различные методы разрешения указанной проблемы:

1 - запрещение переноса в кэш-память «почтовых ящиков» и другой служебной информации, используемой при обмене;

2 - фиксирование попадания в кэш-память подобных ячеек и их принудительное синхронное обновление во всех локальных копиях на аппаратном уровне;

3 - ограничение на максимальное количество чтений ячеек кэш-памяти (БП), подкачка из ОП при достижении максимума;

4 - информирование всех процессоров о попадании разделяемой информации в чью-либо БП.

5 - применение в многопроцессорных системах кэш-память со сквозным своппингом (сквозной записью).

4. ОРГАНИЗАЦИЯ ПРОЦЕССОРОВ

4.1. Назначение и классификация процессоров

Процессор - устройство, осуществляющее процесс автоматической обработки данных и программное управление этим процессом. Процессоры можно классифицировать, например, по следующим признакам:

1) По используемой системе счисления:
- работающие в позиционной системе счисления;
- работающие в непозиционной системе счисления (например, СОК).

2) По способу обработки разрядов:
- с параллельной обработкой разрядов;
- с последовательной обработкой;
- со смешанной обработкой (последовательно-параллельной).

3) По составу операций:
- процессоры общего назначения;
- проблемно-ориентированные;
- специализированные.

4) По месту процессора в системе:
- центральный процессор (ЦП);
- сопроцессор;
- периферийный процессор;
- канальный процессор (контроллер канала ввода/вывода);
- процессорный элемент (ПЭ) многопроцессорной системы.

5) По организации операционного устройства (ОУ):
- с операционным устройством процедурного типа (I-процессоры, -процессоры)
- с преимущественно микропрограммным правлением; процессоры с блочным операционным устройством;
- процессоры с конвейерным операционным устройством(с арифметическим конвейером) (последние два варианта предусматривают аппаратную реализацию большинства операций).

6) По организации обработки адресов:
- с общим операционным устройством;
- со специальным (адресным) операционным устройством.

7) По типу операндов:
- скалярный процессор;

- векторный процессор;
- с возможностью обработки и скалярных, и векторных данных.

8) По логике управления процессором:

- с жесткой логикой управления;

с микропрограммным управлением.

9) По составу (полноте) системы команд:

- RISC (Reduced Instruction Set computer - компьютер с сокращенным набором команд);
- CISC (Complete Instruction Set Computer- компьютер с полным набором команд);
- CISC - процессор с внутренними RISC-подобными инструкциями.

10) По организации управления потоком команд / способу загрузки исполнительных устройств:

- с последовательной обработкой команд;
- с конвейером команд;
- суперскалярные процессоры;
- процессоры с длинным командным словом (VLIW - Very Long Instruction Word) и т. д.

Как всякая классификация, приведенная выше классификация не может считаться полной, так как количество типов процессоров достаточно велико и по своим архитектурам процессоры весьма многообразны.

4.2. Логическая организация процессора общего назначения

Схема, отражающая логическую организацию некоего усредненного процессора общего назначения, представлена на рис. 4.1. В основе структуры процессора лежит взаимосвязь операционной и управляющих частей, что соответствует модели цифрового автомата. Операционные устройства процессора (средства обработки, исполнительные устройства) включают в общем случае ОУ с фиксированной запятой (целочисленное ядро, АЛУ), ОУ с плавающей запятой (числовой сопроцессор или ядро с плавающей запятой), устройство для реализации десятичной арифметики и возможно - устройства для обработки строк и массивов.

Отметим, что в некоторых процессорах отдельно реализуется специальное устройство для вычисления адресов (так называемая разнесенная - decoupled -архитектура), в других процессорах вычисление адресов происходит в общем операционном устройстве. Операционное устройство неразрывно связано с наиболее быстродействующей памятью ВМ - с локальной регистровой памятью процессора. Выделение регистров в отдельный блок на схеме призвано подчеркнуть самостоятельное значение, которое приобретают регистры в

универсальных процессорах - они не просто являются частью операционных устройств, а используются для хранения различной информации как при обработке, так и при вводе-выводе. Целочисленные регистры объединяются в блок регистров общего назначения (РОН), регистры с плавающей запятой - в отдельный блок (в некоторых процессорах эти многоразрядные регистры используются и как векторные регистры в специальных режимах, в других векторные регистры вынесены в отдельный блок).

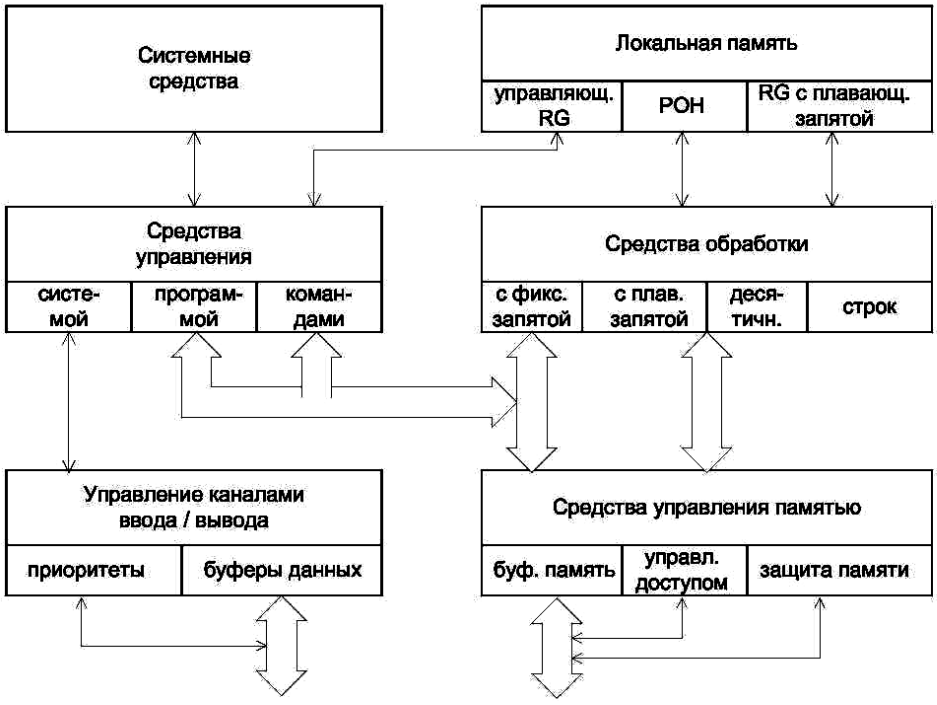


Рис. 4.1

Кроме упомянутых регистров можно выделить набор специальных управляющих регистров, используемых для управления режимами работы процессора, функционированием его различных подсистем, управления памятью и т.д. Средства управления процессором (или - устройство управления) выполняют разнообразные функции, которые включают: управление системой, программой и командами. Управление системой подразумевает управление прерываниями, остановом и запуском процессора, обеспечение отладочного режима и вообще выбор режимов работы процессора и т.д. Управление программой включает обеспечение выполнения ветвлений и циклов, вызовов и возвратов из подпрограмм и т.д. Средства управления командами обеспечивают выполнение машинных циклов работы процессора, то есть - выборки команды, ее дешифрации, собственно управления выполнением команды, управление записью результатов. В данной подсистеме могут реализовываться собственно управляющие

автоматы, отвечающие за реализацию алгоритмов, заложенных в командах процессора (то есть за реализацию микрокода процессора). В некоторых случаях подобные подсистемы относят к УУ, в других - включают в состав собственно средств обработки, то есть рассматривают как часть АЛУ и числового сопроцессора, что в принципе не так важно.

Помимо выполнения операций, вычисления адресов и программного управления этими процессами, процессор должен содержать средства для обеспечения интерфейса как с оперативной памятью, так и устройствами (интерфейсы ввода-вывода). В состав интерфейса с памятью могут включаться буферная память (кэш-память), средства управления доступом и защиты памяти. Интерфейс с каналами ввода-вывода включает буферы данных, систему управления приоритетами, входящую в подсистему прерываний процессора, и т.д.

Под системными средствами понимают встроенные схемы синхронизации, возможно - таймеры, какие-то дополнительные схемы управления, сброса и т.д.

4.3. Операционные устройства процессоров

4.3.1. Операционные устройства процедурного типа и с жесткой структурой. Понятие об I-процессорах и M-процессорах

Операционные устройства процессоров могут строиться с большей или меньшей степенью универсальности, могут быть более простыми, универсальными, требующими большого объема микрокода для реализации всех необходимых алгоритмов операций, либо - более сложными и специализированными, но за счет этого - более производительными и не требующими большого объема управляющего микрокода. Первые устройства можно назвать устройствами процедурного типа, так как они требуют для реализации какого-либо алгоритма арифметической операции выполнения последовательности действий, заданной во времени (то есть процедуры).

Устройства второго типа, рассчитанные на аппаратную реализацию алгоритмов вычислений, можно назвать устройствами с жесткой структурой. (Отметим, что гибкость устройств первого типа заключается не в возможности перестройки их структуры, а в возможности выполнения на заданной структуре большего числа различных алгоритмов.) Примером устройств процедурного типа могут являться, до некоторой степени, устройства для выполнения косвенного умножения. Такие устройства после небольшой доработки могут быть использованы и для реализации других операций (алгоритмов), например, для обычного сложения со знаком, для выполнения деления или операций с плавающей запятой. В предельном случае наиболее универсальной схемой может являться обычный нака-

пливающий сумматор, дополненный схемами выполнения логических операций. С другой стороны, специализированный аппаратный умножитель, например, матричный (матричные умножители подробнее рассматриваются в следующем пункте), является примером устройства с жесткой структурой, рассчитанного только на выполнение конкретной операции, зачастую - определенной разрядности и в определенной кодировке. Для создания более или менее универсального ОУ необходимо иметь набор таких схем для всех требуемых операций, либо - сочетание нескольких специализированных устройств с одним универсальным.

Операционные устройства процедурного типа могут быть построены различными способами. Примером процессоров с более жестким принципом построения операционной части процедурного типа являются так называемые I-процессоры, у которых за определенными регистрами закреплены определенные операции (рис.4.2). На рисунке 4.2 ША и ШД - соответственно шины адреса и данных, Acc - аккумулятор, КС - комбинационные схемы, ТП -триггеры признаков, УУ - устройство управления. Разные регистры соединены с разными операционными элементами (КС) и по-разному соединены друг с другом. Такое разнесение операций по регистрам за счет наличия нескольких операционных элементов в схеме позволяет распараллелить выполнение некоторых вычислений и тем самым повысить производительность. С другой стороны, такая организация подчас лишена необходимой гибкости и требует частых пересылок информации между регистрами.

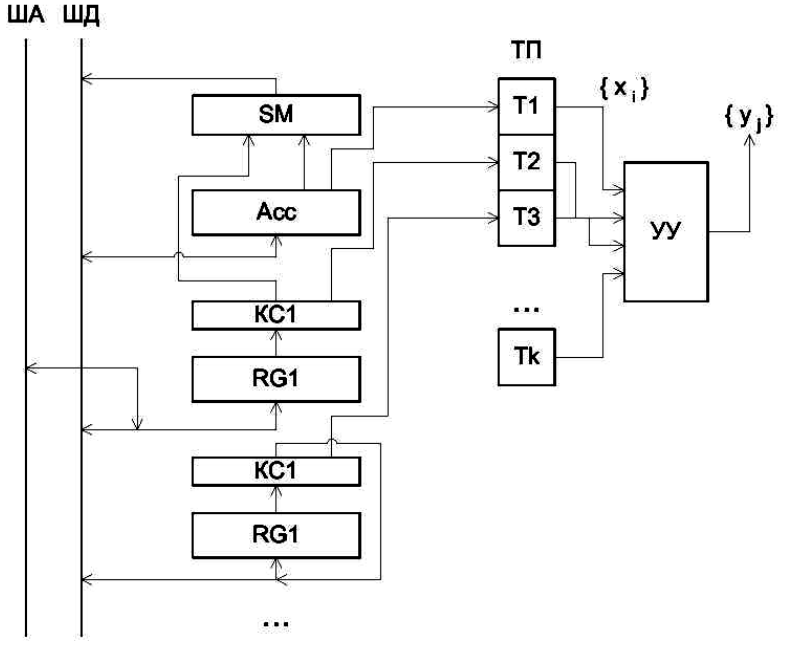


Рис. 4.2

В процессорах с магистральной архитектурой (процессоры М-типа, или процессоры с общим АЛУ) имеется одно обрабатывающее устройство -сумматор, либо АЛУ (например, табличное), с которым связаны все регистры из блока РОН (рис. 4.3).

Регистры являются в данном случае равноправными, каждая пара регистров может участвовать в любой операции. АЛУ связано с регистрами тремя магистралями - магистрали А и В служат для подачи операндов в АЛУ, а магистраль С - для записи результата в выбранный регистр из блока РОН.

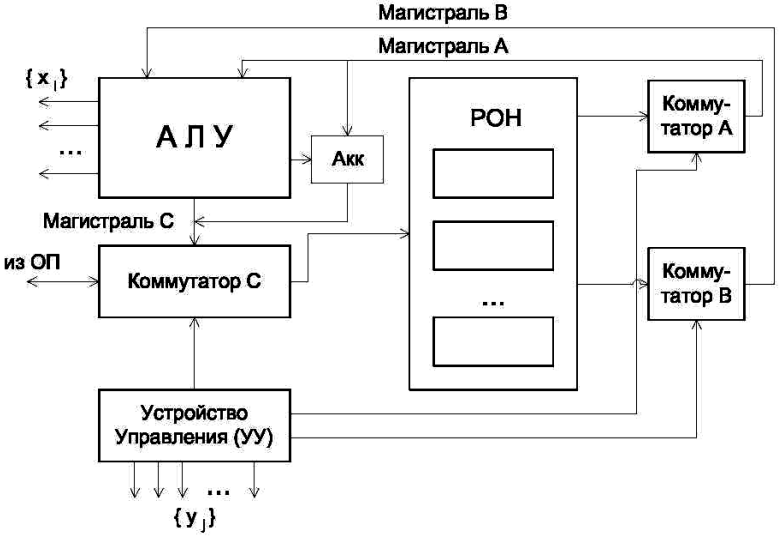


Рис. 4.3

Иногда один из регистров все же выделяется как особый, в котором могут выполняться специальные операции, недоступные для других регистров. В ряде случаев этот регистр всегда является приемником результата (а иногда -обязательно и одним из операндов). Тогда такой регистр называют аккумулятором, а процессор называют процессором на базе аккумулятора. В принципе, в АЛУ такого процессора можно разместить какое-то количество специализированных арифметических устройств жесткой структуры, тогда полученное ОУ будет чем-то промежуточным между процедурным и жестким.

4.3.2. Блочные операционные устройства

Для повышения производительности процессора при выполнении операций его операционное устройство может строиться по блочному принципу. В таких *блочных* О У реализуется несколько функционально независимых исполнительных устройств, выполняющих различные операции (или различные группы операций, например, три блока целочислен-

ного сложения, два - целочисленного умножения, по одному блоку деления, сложения и умножения с плавающей запятой и т.д.).

Эти устройства работают параллельно, обрабатывая каждое свои операнды. Управление этими устройствами осуществляется с помощью так называемых длинных командных слов (Very Long Instruction Word - VLIW) . Командные слова включают инструкции для каждого их исполнительных устройств, а также операнды или указатели на них.

Преимуществом блочных ОУ является более высокая производительность, достигаемая за счет распараллеливания вычислений. В то же время, использование таких устройств не всегда эффективно, поскольку не всегда есть возможность загрузить все исполнительные устройства в каждом такте, в результате часть из них простаивает. Более эффективными часто оказываются конвейерные операционные устройства, поскольку конвейеризовать вычисления в ряде случаев проще, чем распараллелить, что связано с повторением однотипных вычислений в алгоритмах.

4.3.3. Конвейерные операционные устройства

Для конвейеризации вычислений необходимо:

- разбить вычисления на последовательность одинаковых по времени этапов;
- реализовать каждый этап аппаратно в виде ступени конвейера;
- обеспечить фиксацию промежуточных результатов вычислений на выходе каждой ступени в регистрах-защелках.

Напомним, что эффективность конвейера будет тем выше, чем больше задач будет поступать на его вход.

Типичным примером конвейерных операционных устройств могут служить так называемые матричные умножители. Свое название они получили, во-первых, потому, что включают фактически матрицу операционных элементов (сумматоров), а во-вторых, поскольку одной из наиболее очевидных сфер их применения является умножение матриц.

Рассмотрим процесс умножения двух двоичных четырехразрядных положительных чисел:

$$
\begin{array}{cccccccc}
 & & & a_3 & a_2 & a_1 & a_0 & \\
 & & & \times\ b_3 & b_2 & b_1 & b_0 & \\
\hline
 & & + & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & \\
 & + & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & \\
+ & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & \\
+ & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0
\end{array}
$$

По косвенной схеме умножения на устройстве с одним сумматором и набором регистров для реализации этого умножения необходимо в общем случае выполнить 4 шага, на каждом их которых выполняется умножение А на очередной разряд *bi*, сложение А *bi* с текущей суммой частичных произведений и сдвиг новой полученной суммы на 1 разряд вправо. Таким образом, время на выполнение этого умножения можно приближенно оценить как :

$$T_{умн} = 4\left(t_{\&} + 4^{*} t_{sm} + t_{sh}\right),$$

где $t_{\&}$ -задержка на 1 логическом вентиле (при умножении А на *bi),* В формуле не присутствуют затраты на сдвиги, так как они задаются жестко путем соединений линеек сумматоров, кроме того, считаем, что все частичные произведения формируются за 1 логическое умножение. Для нашего случая время на умножение оказывается равным 13 $t_{\&}$. Таким образом, быстродействие умножителя по сравнению с обычной схемой примерно в 3 раза выше. Кроме того, умножитель может работать в режиме конвейера. В данном случае число его ступеней равно 6 (так как в сумматоре с последовательным переносом придется организовывать три отдельные ступени). Пиковая производительность конвейера при полной загрузке - 1 результат за $2t_{\&}$, то есть в 20 раз выше, чем в обычной схеме. Такой выигрыш достигается за счет дополнительных аппаратных затрат, которые выше, чем в первом случае примерно в 4-5 раз.

В умножителе Брауна используются несколько основных способов повышения производительности:

- распараллеливание вычислений (одновременное вычисление всех *Abi);*
- конвейеризация вычислений (цикл умножения разворачивается в последовательность ступеней, межразрядные переносы сохраняются и передаются на следующую ступень);
- аппаратная реализация и специализация вычислений позволяет избежать расходов на сдвиг, который задается жестко, сохранение переноса также диктуется выбранным для аппаратной реализации алгоритмом.

Как уже упоминалось, основным элементом матричного умножителя является сумматор с сохранением переноса (ССП или Carry Save Adder - CSA). Его используют не только в умножителях, но и везде, где необходимо ускорить сложение N чисел. Так, на рис. 4.4. показан сумматор для сложения 3 чисел на базе ССП. Остановимся на принципе построения подобных устройств. Полный сумматор (ПС) позволяет складывать 3 одноразрядных числа. Обычно в качестве третьего слагаемого выступает перенос, поступающий либо с предыдущего сумматора, либо со схемы передачи переноса. Но если в качестве третьего слагаемого использовать соответствующий разряд третьего n-разрядного числа и не передавать перенос в следующий одноразрядный сумматор, то на выходе сумматора сформируется сумма в данном разряде и перенос.

На выходе линейки таких сумматоров формируются два числа - собственно сумма разрядов трех n-разрядных слагаемых и сумма переносов при сложении этих слагаемых. Сумма этих двух чисел и представляет собой значение суммы трех слагаемых:

$$\boldsymbol{S = X + Y + Z = S_{xyz} + C_{xyz}}.$$

Линейка полных сумматоров, обведенная на рис. 4.5 пунктиром - это и есть сумматор с сохранением переноса (ССП). Данная схема имеет 3 входа и два выхода (имеются в виду n-разрядные входы и выходы), поэтому в литературе можно встретить для нее обозначение $ССП_{3\_2}$.

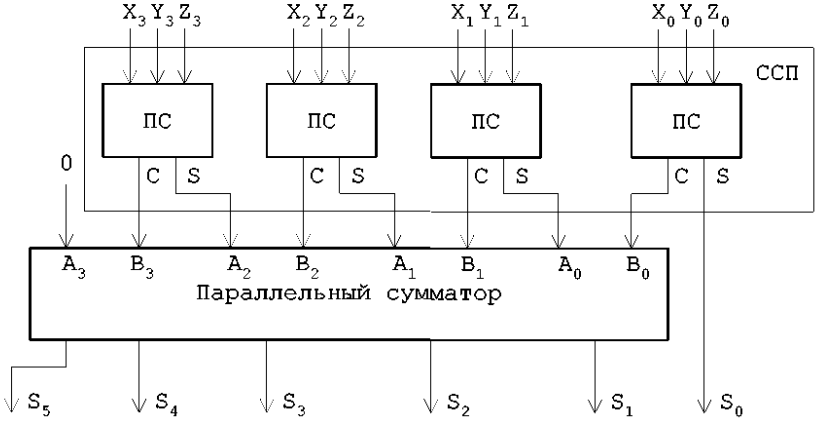


Рис. 4.4

Если подать два полученных числа на обычный параллельный сумматор, то на выходе мы получим сумму 3 чисел. Если использовать не один ССП3_2, а дерево таких сумматоров, как показано на рис.4.5 ($ССП_{8\text{-}2}$), то выполняется сложение 8 чисел, и так далее - для N чисел мы используем схему $ССПN_{\text{-}2}$. Фактически мы имеем схему, похожую на пирамидальную, но с одной общей

схемой передачи и ускорения переноса.

Ускорение схемы на базе ССП по сравнению с пирамидальным включением сумматоров зависит от времени задержки параллельного сумматора со схемой ускоренного переноса (СУП):

$$K_{уск} = \frac{t_{сум} \log_2 N}{t_{сум} + t_{эс} \log_{3/2} N}$$

- где $t_{сум}$ - время задержки параллельного сумматора с СУП,
- $t_{эс}$ - задержка полного одноразрядного сумматора.

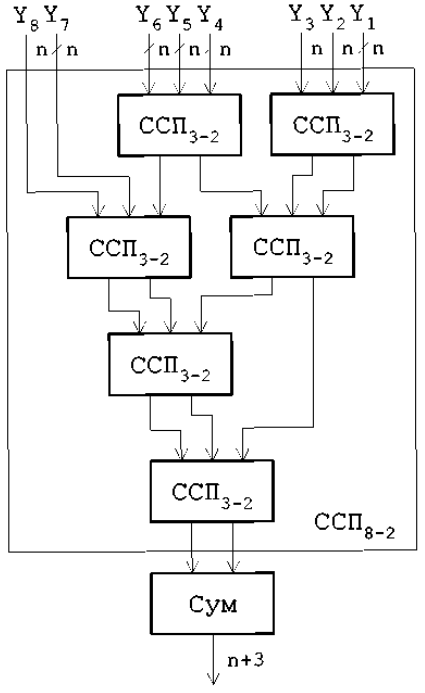


Рис.4.5

При этом необходимо отметить, что для большинства вариантов СУП ускорение схемы с ССП по сравнению с пирамидальной возрастает при увеличении разрядности слагаемых, так как, соответственно растет $t_{сум}$, а $t_{зсне}$ меняется. На базе быстродействующего сумматора на N чисел, аналогичного представленному на рис. 4.5, можно построить древовидный умножитель Уоллеса.

В таком устройстве умножение выполняется в 2 этапа - на первом формируются все частичные произведения вида A • bi • 2^i , на втором - полученные N частичных произведений (где N – количество разрядов множителя без учета знаковых) складываются на сумматоре с $ССПN_{-2}$, как показано нарос. 4.6 на примере умножения на 8-и разрядный множитель. По сравнению с умножителем Брауна мы имеем выигрыш в быстродействии за счет использования большего количества ССП, что позволяет в большей степени распараллелить процесс сложения частичных произведений.

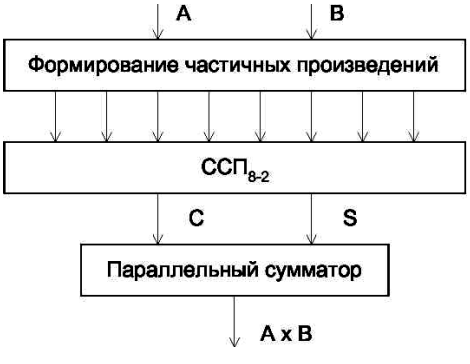


Рис.4.

Конвейерные ОУ могут использоваться самостоятельно, но чаще являются составной частью ОУ процедурного типа, либо - блочных ОУ как аппаратные ускорители выполнения операций.

4.4 Архитектура системы команд. RISC и CISC процессоры

Под архитектурой системы команд (ISA - Instruction Set Architecture) понимают состав и возможности системы команд, общий взгляд на систему команд (СК) и связанную с ней микроархитектуру процессора с точки зрения программиста. Во многом именно архитектура СК определяет трактовку архитектуры компьютера вообще как «…абстрактного представления о вычислительной машине с точки зрения программиста».

Исторически первые микропроцессоры, появившиеся в 70-х годах XX века, имели относительно простую систему команд, что объяснялось небольшими возможностями интегральной схемотехники. По мере увеличения степени интеграции ИМС разработчики МП старались расширять систему команд и делать команды более функциональными, «семантически нагруженными». Это объяснялось, в частности, двумя моментами - во-первых, требованиями экономить память для размещения программ, оставлять больше памяти под данные и т.д., а во-вторых - возможностью реализовать внутри кристалла процессора сложные инструкции быстрее, чем при их программной реализации.

В результате появились процессоры с большими наборами команд, причем команды эти также зачастую являлись достаточно сложными. В последствии эти МП назвали CISC - от Complete Instruction Set Computer - компьютер с полным набором команд или Complex ISC - со сложным набором команд. Типичным примером CISC-процессоров являются процессоры семейства x86 корпорации Intel и ее конкурентов (а также Motorola 68K и другие). Наряду с отмеченными преимуществами процессоры CISC обладали и рядом недостатков, в частности - команды оказывались сильно неравнозначными по времени выполнения (разное количество тактов), плохо конвейеризовывались, требовали сложного (и длительного) декодирования и выполнения.

Для повышения производительности стали использовать жесткую логику управления, что отразилось на регулярности и сложности кристаллов (нерегулярные кристаллы менее технологичны при изготовлении). На кристалле оставалось мало места для РОН и КЭШ.

Кроме того, исследования показали, что производители компиляторов и просто программисты не используют многие сложные инструкции, предпочитая использовать последовательность коротких.

Разработчики подошли к концепции более простого и технологичного процессора с некоторым откатом назад - к простым и коротким инструкциям. С конца 70-х до середины 80-х годов появляются проекты таких процессоров Стэндфордского университета и университета Беркли (Калифорния) - MIPS и RISC.

В основу архитектуры RISC (от Reduced Instruction Set Computer -компьютер с сокращенным набором команд) положены, в частности, принципы отказа от сложных и многофункциональных команд, уменьшения их количества, а также концентрация на обработку всей информации преимущественно на кристалле процессора с минимальными обращениями к памяти.

Основные особенности архитектуры RISC:

1. Уменьшение числа команд (до 30-40).
2. Упрощение и унификация форматов команд.
3. В системе команд преобладают короткие инструкции (например, часто в СК отсутствуют умножения).
4. Отказ от команд типа память-память (например, MOVSB в x86).
5. Работа с памятью сводится к загрузке и сохранению регистров (поэтому другое название RISC - Load-Store Architecture - архитектура типа «загрузка-сохранение»).
6. Преимущественно реализуются 3-х адресные команды, например : add r1, r2, r3 - сложить r2 с r3 и поместить результат в r1.
7. Большой регистровый файл - до 32-64 РОН.
8. Предпочтение отдается жесткой логике управления. Преимущества архитектуры RISC:

1. Облегчается конвейерная, суперскалярная и другие виды параллельной обработки, планирование загрузки, предвыборка, переупорядочивание и т.д.
2. Более эффективно используется площадь кристалла (больше памяти - РОН, кэш).
3. Быстрее выполняется декодирование и исполнение команд - соответственно, выше тактовая частота.

Примерами семейств процессоров с RISC-архитектурой могут служить DEC Alpha , SGI MIPS, Sun SPARC и другие. Большинство современных суперскалярных и VLIW-процессоров (в т.ч. и Intel) либо имеют архитектуру RISC, либо реализуют похожие на RISC принципы, либо - поддерживают CISC-инструкции, но внутри транслируют их в RISC-подобные команды для облегчения загрузки конвейеров и решения других задач.

4.5. Устройства управления процессоров

4.5.1 Назначение и классификация устройств управления

Как уже упоминалось ранее, устройство управления процессора отвечает за выполнение собственно команд процессора, включая основные этапы (загрузка, декодирование, обращение к памяти, исполнение, сохранение результатов), управление выполнением про-

грамм (организация ветвлений, циклов, вызов подпрограмм, обработка прерываний и др.), а также - управляет работой процессора в целом.

Устройства управления классифицируются в зависимости от типа процессора, или - типа управления исполнением команд, который в нем применяется :

- устройства управления процессора общего назначения или -спецпроцессора;
- устройства управления с поддержкой конвейера команд, без такой поддержки, или - с поддержкой многопотокового конвейера (в суперскалярных процессорах), а также - устройство управления процессора с длинным командным словом;
- устройство управления с упорядоченным исполнением команд, неупорядоченным исполнением, выдачей, или завершением команд (с поддержкой динамической оптимизации).

Кроме того, можно выделить устройства управления, построенные на базе памяти микропрограмм (с программируемой логикой), либо - на базе триггерных автоматов (с жесткой логикой).

Мы рассмотрим организацию устройства управления (а вернее - пары устройство управления - операционное устройство) для очень простого учебного RISC - процессора, а затем - рассмотрим способы ускорения работы процессора, основанные на конвейеризации и распараллеливании команд.

4.5.2 Архитектура простого RISC - процессора

Рассмотрим архитектуру простого RISC-процессора на примере некоторого процессора ARC («A RISC Computer») с системой команд, являющейся подмножеством системы команд процессора SPARC. / 16 /

Процессор является 32-разрядным (то есть обрабатывает 32-битовые слова в своем АЛУ), разрядность его команд - также 32 бита. Адресуемая память - 2^{32} байт или 2^{30} команд. Большинство команд процессора – трехадресные. Все команды можно разделить на следующие группы:

1. Команды работы с памятью : ld (load - загрузка) и st (store - сохранение).
2. Логические команды : and, or, nor, srl (сдвиг),

sethi rd, imm22 (установка старших 22 бит регистра в заданные значения).

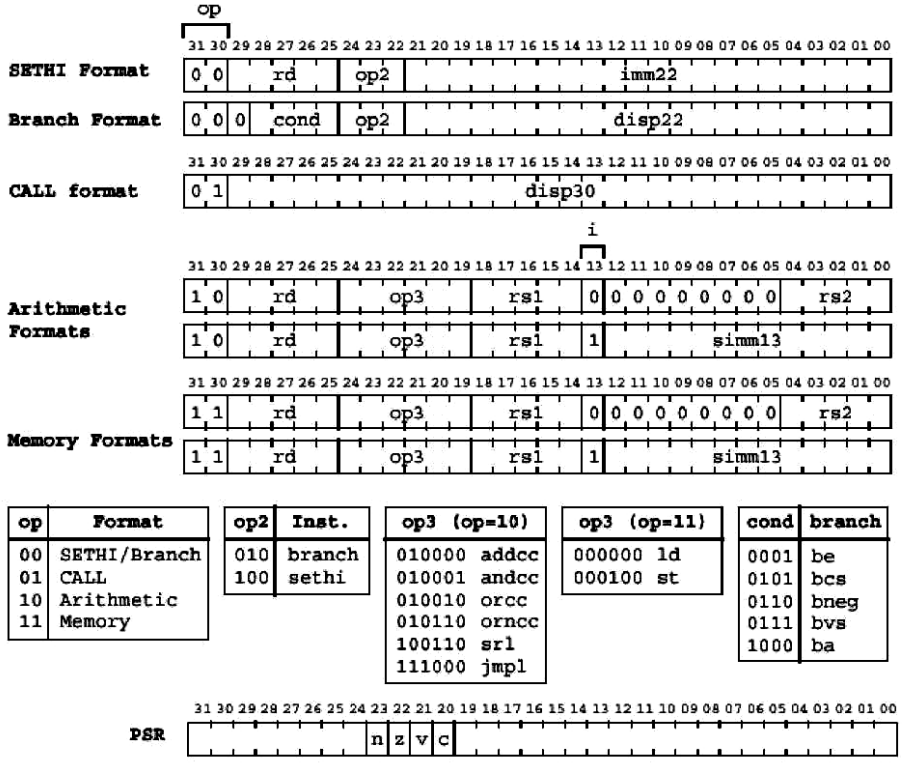
3. Арифметическая команда : add (сложение).
4. Команды управления: ветвления be, bneg, bcs, bvs, ba (безусловный переход), все ветвления в формате be imm22 (относительное смещение), команда call imm30 -вызов подпрограммы, jmpl (ret) - возврат из подпрограммы.

Регистры процессора: 32 РОН, IR (instruction register -регистр команды), PC (program counter - программный счетчик), PSR (Program Status Register - слово состояния программы - 4 флага). Все регистры - 32- разрядные.

В процессоре поддерживаются следующие режимы с адресации:

- непосредственная регистровая;
- косвенная регистровая;
- косвенная регистровая по базе (индексная).

Адресная арифметика в процессоре реализуется на том же АЛУ, что и основные операции. АЛУ построено на таблицах истинности, а также включает программируемый нетактируемый сдвигатель на базе мультиплексора. АЛУ выполняет до 16 арифметических или логических операций. Форматы команд приведены на рис. 4.7



op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
100	sethi

op3 (op=10)
010000 addcc
010001 andcc
010010 orcc
010110 orncc
100110 srl
111000 jmpl

op3 (op=11)
000000 ld
000100 st

cond	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

.

Рис. 4.7

Микроархитектура процессора представлена на рис. 4.8. На рисунке использованы следующие обозначения:

- **Data Section** - операционное устройство (ОУ);
- **Control Section** - устройство управления (УУ);
- **Main Memory** - основная память (ОП);
- **Scratchpad**-сверхоперативное ОЗУ;

– **C BUS MUX** - шинный мультиплексор С для выбора источника данных для регистра-приемника из памяти или с выхода АЛУ;

в регистре команд **ir:**

rd - адрес регистра-приемника, rs1, **rs2** - адреса регистров источников, **i** - флаг непосредственной адресации, **ops** - код операции;

– **MIR** - регистр микрокоманды **(РМК);**

мультиплексоры A, B, C - выбирают адрес соответствующего регистра либо из **ir,** либо - из соответствующего поля **РМК** в зависимости от флагов **MUXA, MUXB, MUXC;**

– **Control Store (CS)** - память микропрограмм **(ПМП);**
– **CSAI** - счетчик адреса микропрограммы;
– **CS Address MUX** - мультиплексор адреса микропрограммы (3 канала –**Next** следующий адрес из CSAI, **Jump** -переход по адресу, указанному в РМК, **Decode** - переход к микро-подпрограмме реализации команды);
– **CBL** - логика управления ветвлением; **%psr** - регистр состояния программы, хранит 4 флага результата последней операции: n-netgative (отрицательное число), z-zero (ноль), v-overflow (переполнение), c-carry (перенос);
– **АСК** - подтверждение о готовности памяти для инкремента адреса микрокоманды; в РМК также отметим поля: **RD/WR** - чтение/запись памяти, **ALU** - код операции АЛУ, **JUMP ADDR** - адрес перехода в микропрограмме.

Операционная часть ARC соответствует операционной части М-процесора.. Работу процессора коротко можно прокомментировать следующим образом.

Машинный цикл выполнения команды в общем случае (не для рассматриваемого процессора) включает:

1. Извлечение команды из памяти (IF - Instruction Fetch).
2. Декодирование команды (Instruction Decoding - ID).
3. Извлечение операндов из памяти или из регистров (MEM).
4. Выполнение (Execute - EX).
5. Запись результатов в память или регистр (Write Back - WB).

Для данного процессора обращение к памяти (MEM) и (WB) происходят только в 2 командах - ld и st. В остальных случаях все действия происходят с регистрами РОН. Поскольку у процессора ARC нет отдельного адресного операционного устройства, а режимы адресации предусматривают в том числе и косвенную адресацию, то этап выполнения EX в нем предшествует этапу обращения к памяти (MEM или WB) - на этом этапе необходимо вычислить окончательный адрес памяти, по которому будет обращение. В резуль-

тате среднее число тактов на команду (clocks per instruction - CPI) -около 3-4 на команду, и, кроме того, 1 загрузка команды из памяти. Производительность этого процессора можно оценить следующим образом. Среднее время выполнения (в тактах) :

$$Tк = 3t + 1{,}5tmem,$$

где t - длительность одного такта процессора, = tmem - длительность обращения к памяти. При тактовой частоте 100Мгц t=10нс. Пусть время обращения к памяти составляет даже 20нс. Получаем Тк = 3*10нс + 1,5*20нс = 60нс. Производительность = 1/Тк = 1/60нс = менее 20 MIPS.

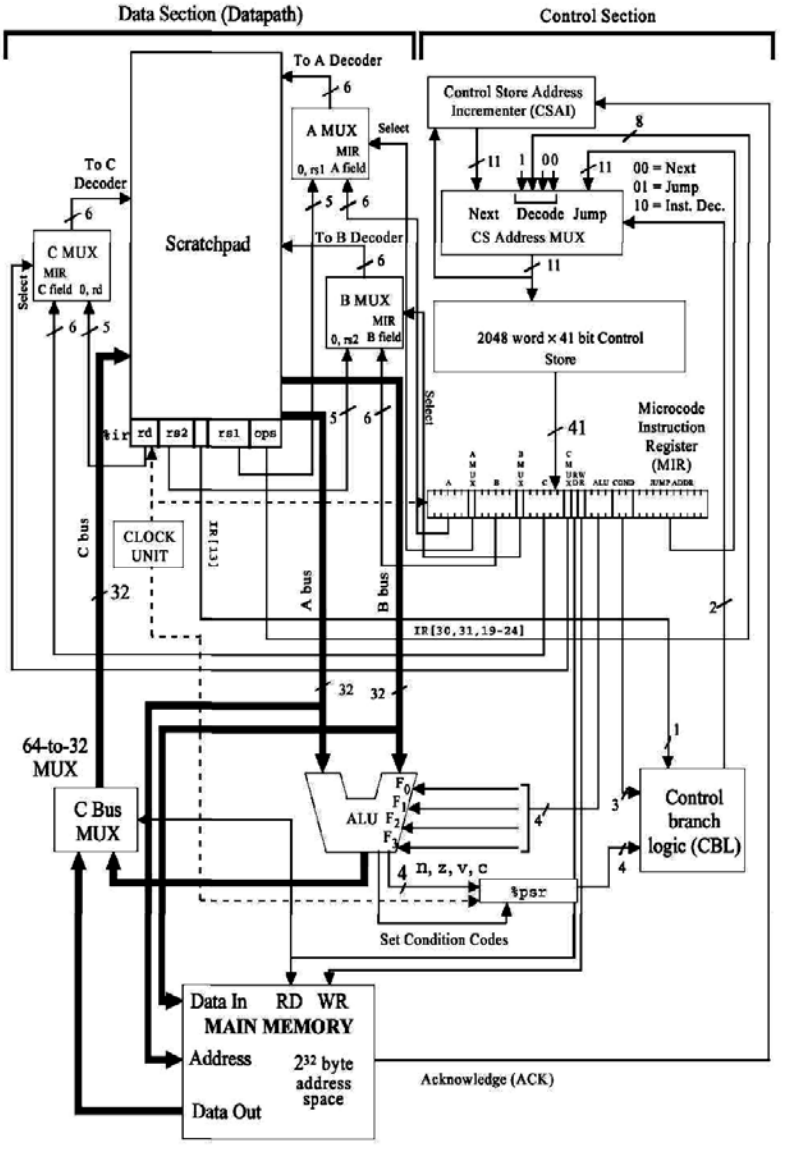


Рис. 4.8.

Производительность этого процессора можно оценить следующим образом. Среднее время выполнения (в тактах) :

Тк = 3t+ 1,5tmem,

где t - длительность одного такта процессора, = tmem - длительность обращения к памяти. При тактовой частоте 100Мгц t=10нс. Пусть время обращения к памяти составляет даже 20нс. Получаем Тк = 3*10нс + 1,5*20нс = 60нс. Производительность = 1/Тк = 1/60нс = менее 20 MIPS. Показатели производительности многих современных процессоров (и RISC и CISC) даже на той же частоте намного выше. (Например, Celeron 400 Мгц имеет производительность около 1000 MIPS - на частоте 100 МГц он бы имел производительность 250MIPS, то есть в 10 раз больше, чем у рассмотренного процессора). Как достигается повышение производительности ? Во-первых, можно несколько улучшить показатель CPI, если перейти к жесткой логике управления, то есть вместо микроподпрограммы выполнения команды реализовать аппаратную схему, выполняющую алгоритм заданной команды.

С другой стороны, можно использовать КЭШ-память для ускорения доступа к основной памяти. Однако, этих мер недостаточно для повышения производительности в 10 и более раз.

В современных процессорах для повышения производительности применяют, в том числе, 2 основных подхода: конвейеризацию команд и суперскалярное выполнение команд (многопотоковые конвейеры команд).

4.5.3 Конвейер команд

В общем случае приведенные ранее основные пять этапов выполнения команды процессора общего назначения требуют разного времени, но -сопоставимого. Если добиться (введением фиксаторов и синхронизацией), чтобы каждый этап занимал одинаковое время, можно организовать конвейер команд, в котором одновременно на разных этапах выполнения будут находиться несколько команд (Рис.4.9).

ADD R1, R2, R3
SUB R4, R5, R1

IF	ID	MEM	EX	WB	
	IF	ID	MEM	EX	WB

Рис. 4.9.

Даже при условии некоторого увеличения времени выполнения одной команды (небольшое снижение быстродействия) производительность при полном заполнении конвейера будет близка к величине $1/T_к$, где $T_к$ - такт конвейера, в данном случае - время выполнения одного этапа. Это позволило бы сразу увеличить производительность процессора в 5 раз ! Однако на практике добиться этого оказывается сложно. И препятствуют этому так называемые конфликты при конвейеризации.

Конфликтом при конвейеризации команд называют ситуацию, которая препятствует выполнению очередной команды из потока команд в предназначенном для нее такте.

Конфликты делятся на три основные группы:

1. Структурные или ресурсные.

Возникают в результате того, что аппаратные средства не могут поддерживать все комбинации команд в режиме их одновременного выполнения с совмещением на конвейере. Это происходит в случае, если какие-то устройства в процессоре не конвейеризованы, либо - присутствуют в единственном экземпляре (не распараллелены). Например, могут возникать конфликты при обращении к общей КЭШ-памяти: одну команду необходимо извлечь из памяти (на первом этапе выполнения), а другая пытается записать результат в память на заключительном этапе. Для борьбы с ресурсными конфликтами в основном применяют три способа:

- приостановка конвейера (pipeline stall, pipeline "bubble" - конвейерный "пузырь") до разрешения конфликта (до завершения первой конфликтующей команды);
- дублирование аппаратных средств, вызывающих конфликт, например, разделение КЭШ-памяти на КЭШ команд и КЭШ данных;
- ускорение или конвейеризация проблемного устройства, что позволяет снизить затраты времени на приостановку. Решение об увеличении аппаратных затрат в последних двух случаях принимают, если конфликт возникает часто, так как дополнительные затраты могут быть существеннее, чем потери производительности от приостановки конвейера, если она происходит редко.

2. Конфликты программные или информационные. Делятся на две подгруппы:

а) конфликты по данным, возникающие в случае, если выполнение следующей команды зависит от результата предыдущей.

б) конфликты по управлению, возникающие при нарушении естественного порядка следования команд (условная передача управления).

3. Выделяют несколько вариантов конфликтов по данным:

1) Конфликт типа «чтение после записи» (Read After Write - RAW). Допустим, имеются две команды - команда Ai и команда Aj, причем команда Ai предшествует команде Aj. Конфликт RAW возникает, если команда Aj использует результаты работы команды Ai, то есть должна прочитать регистр, либо память после записи туда результата командой Ai, но к моменту чтения данные еще не записаны, поскольку команды следуют друг за другом на конвейере и сдвинуты всего на один этап.

На рисунке 4.10. обведены этапы, на которых будет записан результат в первой команде и потребуется считать результат для второй команды. Очевидно, что нужный результат еще не будет находиться по месту, адресуемому второй командой, и произойдет конфликт.

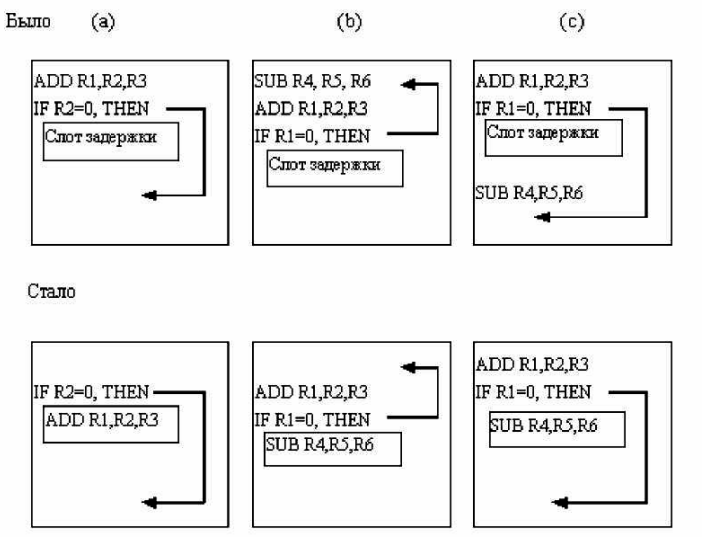


Рис. 4.10

2) Конфликт типа «запись после чтения» (WAR). Происходит, если команда Aj записывает результат до того, как он считывается командой Ai (предшествующей Aj). Такой тип конфликта может возникать только в случае, если команда Aj обгоняет команду Ai на конвейере в конвейерах с неупорядоченной обработкой, выдачей или завершением команд (out-of-order execution, out-of-order completition).

3) Конфликт типа «запись после записи» (WAR). Возникает, если последующая команда Aj записывает результат до того, как запишет его команда Ai, что может привести к нарушению логики программы, если, например, между этими командами стоит еще какая-нибудь команда, проверяющая этот адрес (регистр) (пример). Такой конфликт также может происходить в случае неупорядоченного выполнения команд.

Методы борьбы с конфликтами по данным:

а) остановка конвейера;

б) реализация механизмов обхода и продвижения данных (data bypassing & forwarding);

в) планирование загрузки конвейера компилятором (статическая оптимизация);

г) неупорядоченное выполнение команд в процессоре (динамическая оптимизация);

Статическое планирование позволяет еще на этапе компиляции переупорядочивать команды таким образом, чтобы они по возможности не конфликтовали друг с другом. Например, рассмотрим 2 оператора языка:

A:=B + C; D:=E + F.

Неоптимизированный код, реализующий эти операции, представлен в первой колонке таблицы 3.1. В данном потоке команд возникают конфликты типа RAW после второй и седьмой команд, а также после 4-ой и восьмой. Конфликты при записи в память могут быть устранены с помощью механизма обходов, а конфликты при чтении памяти - с помощью переупорядочения потока команд, как показано во второй колонке таблицы 4.3.

При динамической оптимизации команды, вызвавшие конфликт, могут задерживаться на конвейере, а следующие за ними команды, не вызывающие конфликтов и не зависящие от конфликтных команд, пускаются в обход них. Анализ конфликтных ситуаций и выбор команд для переупорядочивания возлагается на логику управления конвейером в самом процессоре.

Таблица 4.3.

Исходная последовательность команд			Переупорядоченная последовательность
LW Rb,B LW Rc, C +--- --- ADDRa, Rb, Rc ----- SW A,Ra *— LW Re,E <«------1 LW Rf,F ADDRd,Re,Rf *~ SW D,Rd			LW Rb,B LW Rc, C LW Re,E ADD Ra, Rb, Rc LW Rf,F SW A,Ra ADD Rd, Re, Rf SW D,Rd

При переименовании регистров логические имена регистров, присутствующие в командах, динамически отображаются на физические регистры процессора, которых, как правило, больше (если речь идет о RISC-подобной архитектуре). В результате различные данные, относящиеся к одному и тому же логическому регистру, помещаются в разные физические регистры. Соответствие между регистрами задается таблицей отображения, которая динамически обновляется после декодирования каждой команды. Это облегчает процессору планирование загрузки конвейера.

Конфликты по управлению возникают при конвейеризации команд условных переходов. По статистике до 15%-20% всех команд в программе - это команды условных перехо-

дов. Конфликт проявляется в том, что адрес условного перехода определяется только в конце выполнения команды, в то время, как конвейер уже должен быть заполнен командами из какой-то одной ветви. Если не обрабатывать конфликт, то производительность конвейера может снижаться в 2 и большее число раз. Способы борьбы с конфликтами по управлению можно разделить на 2 группы: статические и динамические.

К статическим методам можно отнести:

1) Возврат, т.е. статическое прогнозирование перехода как всегда выполняемого, либо - всегда не выполняемого с последующей очисткой конвейера в случае неправильного прогноза и возвратом к нужной команде.

2) Разворачивание циклов. Поскольку многие условные переходы связаны с определением условия выхода из цикла, то в случае, когда число шагов цикла заранее известно и невелико, можно «развернуть» цикл, то есть продублировать тело цикла столько раз, сколько необходимо, отказавшись от проверки условия вообще. Это приведет к увеличению кода программы, но избавит от конфликта по управлению.

3) Задержанные переходы (использование «слотов задержки»).

Под слотами задержки понимают участки программы, которые будут занесены в конвейер и успеют выполниться до выяснения адреса перехода в условной команде. Идея их использования состоит в том, чтобы заполнять слоты задержки «полезными» или «невредными» командами, то есть такими, которые либо не зависят от условия, либо - не приведут к нарушению логики программы, даже если их выполнение окажется лишним. (Рис. 4.9)

4) Предсказание переходов на основе профиля программы. Профилирование (profiling) предусматривает составление прогноза о выполнении тех или иных переходов на основе статистических наблюдений за программой по результатам ее многократного прогона.

К динамическим методам можно отнести:

1) Приостановку конвейера до выяснения адреса перехода.

2) Реализацию команд условного перехода в процессоре таким образом, чтобы адрес перехода выяснялся на начальных этапах выполнения команды.

3) Динамическое предсказание ветвлений в процессоре (branching predict).

Динамическое предсказание ветвлений в процессорах осуществляется с помощью буферов предсказания перехода (БПП - Branch Predicting Buffer -BPB). Чаще всего в них используется счетчик прогнозов, который представляет собой обычный n-

разрядный двоичный счетчик. При каждом выполненном переходе счетчик прогнозов для данного перехода увеличивается, а при невыполненном - уменьшается на единицу. Если текущее значение счетчика > 2^{n}-^{1}, то переход прогнозируется как выполняемый, иначе - как невыполняемый. На практике ограничиваются либо 1-битным, либо 2-битными счетчиками, которые при этом обеспечивают вероятность правильного прогноза соответственно до 70% и 85%.

Для еще большего ускорения предсказания используют буфер целевых адресов переходов (Branch Target Buffer - BTB), представляющий собой ассоциативную кэш -память, в которой в качестве тегов используются адреса команд ветвления в текущей части программы, а в ячейках содержатся счетчики прогнозов и целевые адреса перехода при условии его выполнения. Процессор при выборке команды проверяет, не хранится ли ее адрес в BTB, считывает счетчик прогнозов и в зависимости от его значения принимает решение о выборке команд по следующему адресу или по адресу, указанному в BTB.

4.5.4 Суперскалярные архитектуры

Итак, использование конвейера команд позволяет в лучшем случае снизить показатель CPI до 1, то есть на каждом такте с конвейера должна «сходить» новая обработанная команда. В этом случае производительность нашего процессора ARC должна увеличиться в 4 раза, при его длительности такта в 10 нс (тактовая частота 100 Мгц) имеем производительность в 100 MIPS. Но во-первых, у Celeron такой показатель равняется, как мы выяснили, где-то 250, а во-вторых - как показано ранее, достижение показателя 1 CPI не всегда возможно из-за конфликтов при конвейеризации. То есть реально мы будем иметь в лучшем случае 1,5-2 CPI. Как же достигается такая высокая производительность в Celeron и других процессорах с архитектурой P6? Для этого в них используется суперскалярная обработка, то есть обработка с многопотоковым конвейером команд, когда процессор может выполнять больше 1 команды за такт (CPI < 1, или - IPC > 1).

Фактически в суперскалярном процессоре несколько потоков проходят через несколько исполнительных устройств, а остальные ступени так или иначе работают с одним потоком. Для согласования разных скоростей потоков декодирования, выборки, трансляции в RISC - подобные инструкции, переупорядочивания и потоков в исполнительных устройствах применяют различные очереди инструкций (буферы FIFO), которые есть у каждого из исполнительных устройств (рис. 4.10)

Необходимо отметить, что в суперскалярных процессорах происходит очень сложное и нетривиальное преобразование последовательного потока команд исходной программы в параллельный поток триад (операция + операнды + назначение результата), параллельно продвигающихся по очередям команд в исполнительные устройства. Процессор ограничен в возможностях такого преобразования, а также в возможностях спекулятивного ис-

полнения (подготовки загрузки альтернативных ветвей ветвления) и прогнозирования ветвлений размером т.н. «окна исполнения», то есть той частью программного кода, который процессор «видит» в процессе выполнения в данном такте. Все это ограничивает возможности распараллеливания потоков команд до величин порядка 6-8.

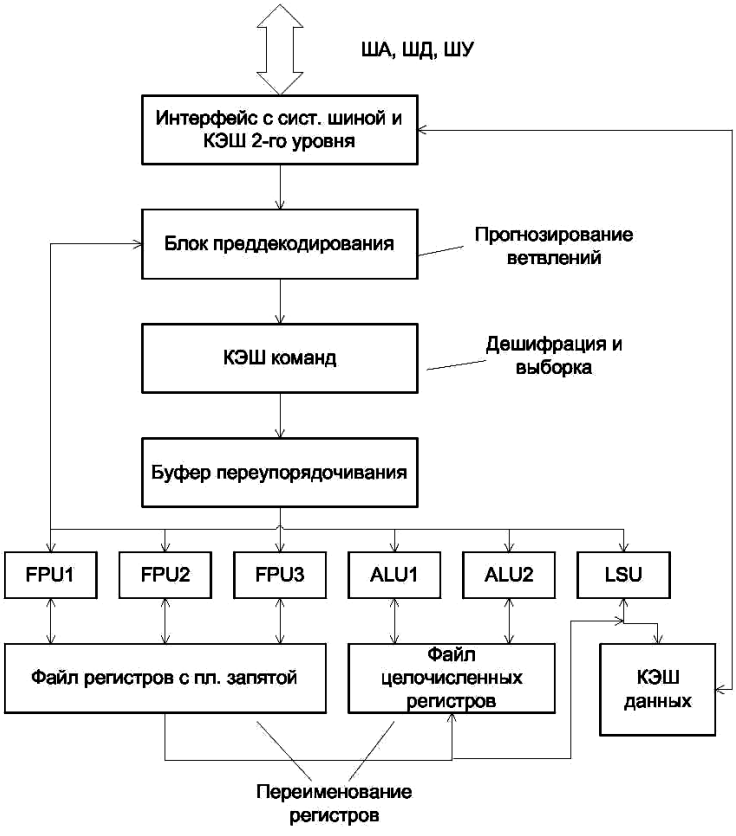


Рис. 4.10

4.5.5 Процессоры с длинным командным словом (VLIW).

В процессорах с длинным командным словом (Very Long Instruction Word) используется альтернативный суперскалярной обработке принцип распараллеливания последовательного алгоритма. В основном вся тяжесть планирования загрузки большого числа исполнительных устройств в таком процессоре (а у него блочное операционное устройство) ложится на программиста, или - на оптимизирующий компилятор. В процессор поступают уже сформированные триады для всех исполнительных устройств, так что ему только остается выполнять эти длинные команды. В результате он не ограничен размером окна исполнения, так как и программист, и компилятор видят весь код программы, и могут извлечь из него максимальный параллелизм.

Такой подход позволяет достичь принципиально более высокой производительности (например, тестирование процессоров Itanium с архитектурой IA-64, использующей принципы VLIW, указывает на 10-кратное ускорение при выполнении ряда вычислений), но такие процессоры обладают и рядом недостатков:

- в целом менее эффективная загрузка исполнительных устройств, так как не всегда можно сформировать достаточное количество команд для параллельного исполнения;
- сложности обработки условных переходов;
- сложность программирования и др.

Последнее обстоятельство ограничивает применение процессоров VLIW, даже Intel, в персональных ЭВМ, так как для этого придется кардинально переписывать все программное обеспечение, поскольку в существующем виде оно не даст прироста производительности на этих процессорах. Сфера применения VLIW-процессоров пока ограничена серверами, производительными рабочими станциями, а также многопроцессорными ЭВМ.

Что касается обработки условных переходов, то тут можно отметить широкое использование в процессорах VLIW так называемых условных (conditional) команд. Это команды, использующие предварительно рассчитанные логические значения (предикаты), для выполнения, либо пропуска какого-то действия (наподобие операторов языков высокого уровня c := iif (a>b, a, c)), что позволяет избавиться от нескольких ветвей при коротких условных переходах и использовать один поток команд без необходимости предсказывать адрес для следующей выборки.

4.6 Обзор архитектур процессоров Intel

Корпорация Intel является "законодателем мод" на рынке микропроцессоров, а ее продукты стали де-факто стандартом в компьютерной индустрии. Конечно, существует большое количество других производителей и распространенных семейств МП (те же процессоры Motorola и др.), однако наиболее распространенными во всем мире, и особенно в России, являются все же процессоры Intel.

Кроме того, в течении многих лет другие разработчики МП (AMD, NextGen, VIA и др.) выпускают свои аналоги процессоров, совместимых по системам команд с МП Intel. Поэтому, анализируя эволюцию процессоров Intel, мы проследить историю развития микропроцессоров общего назначения вообще. А история развития процессоров Intel подтверждает в целом закон Мура, сформулированный одним из основателей империи Intel Гордоном Муром еще в 1965 году: "каждые 1,5-2 года выпускается новый процессор, степень интеграции которого (и производительность) вдвое выше, чем у предыдущего."Согласно информации Intel /11/, за 24 года количество транзисторов в кристалле МП увеличилось более чем в 3700 раз от 29 тыс. в процессоре i8086 (выпущен в 1978 г.) до 108 млн в Intel

Pentium IV. При этом производительность процессоров возросла более чем в 6000 раз (от 0.8MIPS для i8086 до приблизительно 5000 MIPS для Pentium IV 2,6 ГГц) !

Наряду с прогрессом интегральной технологии в ходе эволюции процессоры Intel претерпевали и значительные архитектурные изменения. Если говорить об архитектуре, известной как x86, то она ведет начало от процессора i8086 до наших дней (Pentium III и IV).

Существенной особенностью всех процессоров x86 является их совместимость снизу вверх, что позволяет до сих пор пользоваться программами, написанными 20 лет назад!

В таблице 4.4 приведены характеристики основных процессоров x86 вплоть до Pentium III, а в таблице 4.5 - характеристики последнего поколения процессоров Pentium IV. Проследим кратко историю эволюции этих процессоров.

Таблица

Характеристики процессоров Intel (i8086-Pentium III)

Intel Processor	Date Introduced	Max. Clock Frequency at Introduction	Transistors per Die	Register Sizes[1]	Ext. Data Bus Size[2]	Max. Extern. Addr. Space	Caches
8086	1978	8 MHz	29 K	16 GP	16	1 MB	None
Intel 286	1982	12.5 MHz	134 K	16 GP	16	16 MB	Note 3
Intel386 DX Processor	1985	20 MHz	275 K	32 GP	32	4 GB	Note 3
Intel486 DX Processor	1989	25 MHz	1.2 M	32 GP 80 FPU	32	4 GB	L1: 8KB
Pentium Processor	1993	60 MHz	3.1 M	32 GP 80 FPU	64	4 GB	L1:16KB
Pentium Pro Processor	1995	200 MHz	5.5 M	32 GP 80 FPU	64	64 GB	L1: 16KB L2: 256KB or 512KB
Pentium II Processor	1997	266 MHz	7 M	32 GP 80 FPU 64 MMX	64	64 GB	L1: 32KB L2: 256KB or 512KB
Pentium III Processor	1999	500 MHz	8.2 M	32 GP 80 FPU 64 MMX 128 XMM	64	64 GB	L1: 32KB L2: 512KB

Первый процессор семейства x86 (или - Архитектуры Intel) - i8086 - был 16-разрядным, имел 16-разрядную внешнюю шину данных и 20-разрядную шину адреса, что позволяло адресовать до 1Мб внешней памяти. Память имела сегментную организацию с сегментами до 64К. Аналогичный процессор i8088 имел внешнюю шину в 8 бит, что удешевило популярные персональные системы IBM PC/XT. В процессоре i80286 был реализован "защищенный режим" работы, позволявший адресовать до 16Мб памяти, использовавший дескрипторные таблицы, систему с кольцами защиты памяти и аппаратной поддержкой переключения задач. Это новшество позволило перенести в среду

персональных ЭВМ элементы операционных систем больших ЭВМ и майнфреймов - многозадачность, защиту памяти и системных ресурсов.

Таблица 4.5

Intel Processor	Date Intro-duced	Micro-Architecture	Clock Frequency at Intro-duction	Transis-tors Per Die	Register Sizes[1]	System Bus Band-width	Max. Extern. Addr. Space	On-Die Caches[2]
Pentium III and Pentium III Xeon Processors[3]	1999	P6	700 MHz	28 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	Up to 1.06 GB/s	64 GB	3-KB L1; 256-KB L2
Pentium 4 Processor	2000	Intel NetBurst Micro-architecture	1.50 GHz	42 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Execution Trace Cache; 8KB L1; 256-KB L2
Intel Xeon Processor	2001	Intel NetBurst Micro-architecture	1.70 GHz	42 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Trace Cache; 8-KB L1; 256-KB L2
Intel Xeon Processor[4]	2002	Intel NetBurst Micro-architecture; Hyper-Threading Technology	2.20 GHz	55 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Trace Cache; 8-KB L1; 512-KB L2
Intel® Xeon™ Processor MP[4]	2002	Intel NetBurst Micro-architecture; Hyper-Threading Technology	1.60 GHz	108 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	3.2 GB/s	64 GB	12K µop Trace Cache; 8-KB L1; 256-KB L2; 1-MB L3

Процессор i80386 был 32-разрядным, адресовал до 4Гб памяти, включал аппаратную поддержку виртуальной памяти и возможность адресовать всю память в «плоском» режиме. На машинах с этим процессором можно было реализовывать операционную систему UNIX - классическую систему майнфреймов. В процессоре 386 уже использовалось распараллеливание при одновременной работе 6 устройств процессора.

В процессоре i80486DX появилась встроенная КЭШ-память 1 уровня (L1) объемом 8К, встроенный математический сопроцессор, а также -пятиступенчатый конвейер в устройствах декодирования и исполнения команд.

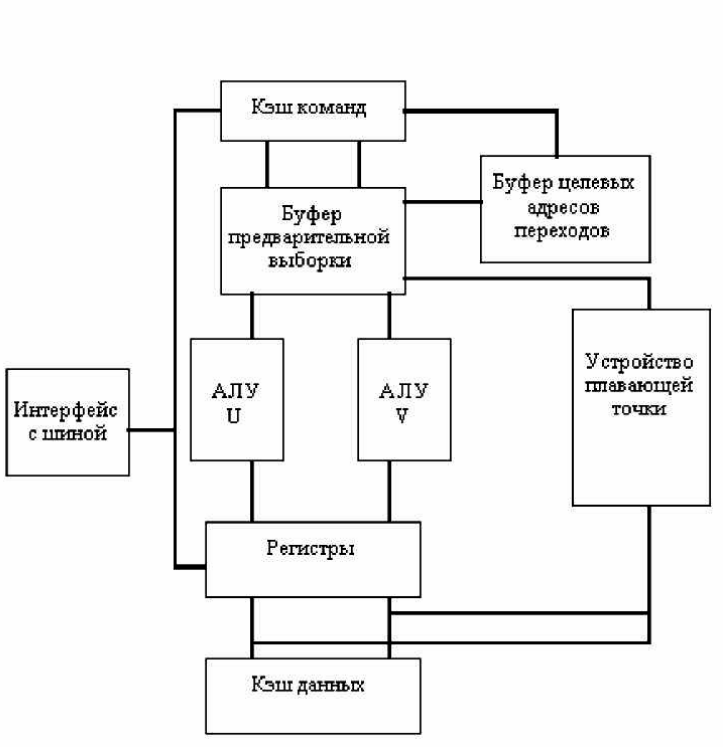


Рис 4.12

Процессор IntelPentium явился первым суперскалярным процессором (Рис.4.12). В нем был реализован двух потоковый конвейер, который позволял одновременно обрабатывать до двух команд в такте. (Правда, при этом существовало много ограничений, но тем не менее.) Процессор включал раздельную КЭШ-память для инструкций и данных (по 8К), поддерживающую режим обратной записи.

Внутренняя разрядность процессора осталась 32 разряда, но некоторые внутренние шины имели разрядность 128 и даже 256 разрядов, а внешняя шина - 64 разряда. В процессоре было реализовано динамическое предсказание переходов и поддержка мультипроцессорных конфигураций.

Появление процессора Pentium Pro дало начало новой модификации Intel Architecture - архитектуре P6.

Процессор (рис. 4.13) имеет 3-х потоковый конвейер, что позволяет достичь большей степени распараллеливания по сравнению с обычным Pentium. Главной отличительной особенностью процессора является, пожалуй, динамическое исполнение (Dynamic Execution) - реализация неупорядоченного выполнения, спекулятивного исполнения (исполнения по предположению) и усовершенствованного блока предсказаний. В процессоре реализована суперконвейерная архитектура, поскольку он содержит 13 более мелких ступеней конвейера по сравнению с 5 у Pentium, на которых исполняются спе-

циальные RISC-подобные инструкции процессора, получившие названия micro-ops. Три буфера декодирования параллельно формируют три потока таких инструкций, которые затем направляются в пять исполнительных устройств, результаты обработки в которых затем собираются в правильном порядке в блоке сборки.

Процессор содержит две КЭШ-памяти L1 по 8К, а также - КЭШ второго уровня (L2) объемом в 256К, реализованную в том же корпусе, что и основной кристалл процессора и обменивающуюся с ним по скоростной шине шириной в 64 разряда. Шина адреса процессора имеет 36 разрядов, что обеспечивает адресное пространство в 64 Гбайт.

Как мы видим, прогресс в области архитектур процессоров Intel во многом определялся последовательной реализацией способов борьбы с конфликтами при конвейеризации, рассмотренными нами ранее.

Уже после реализации Pentium Pro Архитектура Intel дополнилась реализацией векторных SIMD - инструкций, которая получила название технологии MMX (MultiMedia eXtensions - мультимедийные расширения). Суть новой технологии заключалась в реализации с помощью широких регистров математического сопроцессора целочисленных команд обработки векторов размерностью до 8 чисел по 8 бит. Эти команды позволяли ускорить обработку видео и аудио информации за счет ускорения векторных и матричных операций. Так появились процессоры Pentium MMX, с максимальной частотой до 233Мгц (300 Мгц в варианте для notebook).

Процессор Pentium II стал дальнейшим развитием архитектуры P6 (фирма Intel широко использует для своей основной архитектуры обозначение IA32). Архитектурно процессор стал симбиозом Pentium Pro с поддержкой MMX -инструкций. Кроме того, в нем реализована КЭШ-память с более быстродействующей шиной и большего объема, а также ряд технологических и конструктивных улучшений, связанных с выбором корпуса, процессорной шины и т.д.

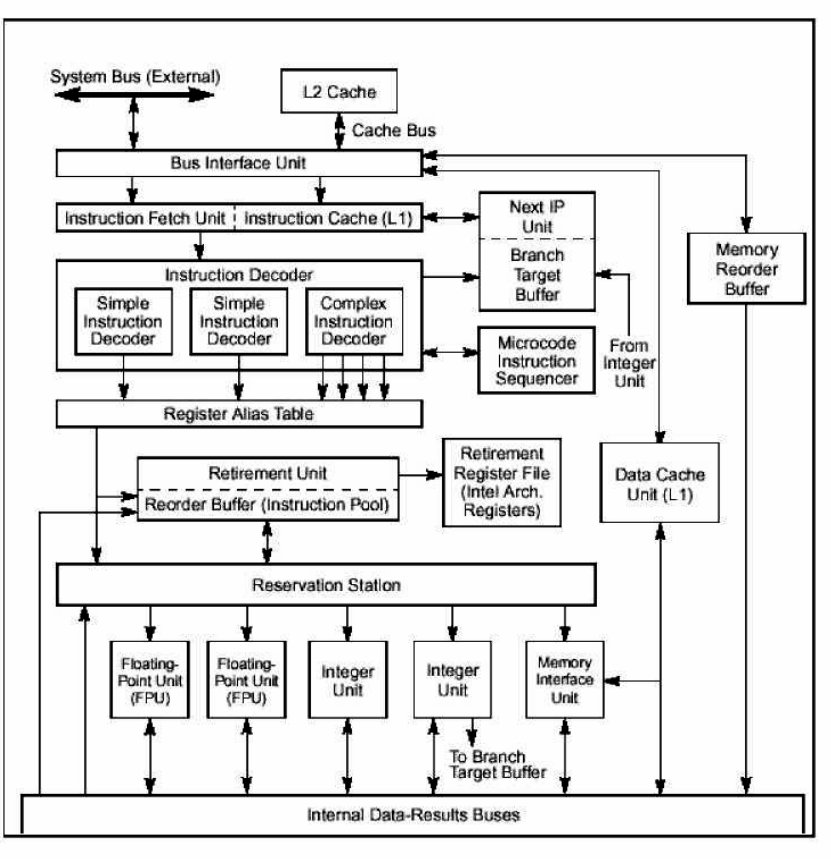


Рис. 4.13

Процессор Pentium III, заявленный как процессор, специально спроектированный для поддержки Internet-приложений, в дополнение к стандартному набору инструкций MMX включает набор из 70 новых инструкций SSE (Streaming SIMD Extensions - потоковые расширения SIMD). Кроме того, теперь каждый процессор снабжен уникальным серийным номером для его идентификации в сети Internet. Ряд серьезных изменений произошел в архитектуре процессора Pentium IV. Этот процессор уже не относится к архитектуре P6, хотя продолжает линейку IA-32. К его ключевым особенностям можно отнести кэш второго уровня, помещенный на кристалл процессора, увеличенную до 400(533)МГц частоту передней шины, 144 новых векторных инструкций SSE2, хранение так называемых трасс декодированных внутренних инструкций в специальном КЭШе и другие изменения.

5 ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА

5.1 Назначение и состав системы ввода-вывода.

Система ввода-вывода (СВВ) предназначена для выполнения трех из пяти основных функций ВМ: собственно ввода, вывода и долговременного хранения информации (две другие функции - это обработка информации и управление этой обработкой).

В состав системы ввода-вывода входят:

1. Периферийные устройства, которые можно подразделить на:

- устройства ввода информации;
- устройства вывода (отображения) информации;
- устройства вывода информации на исполнительные устройства (управление техническими и производственными объектами и т.д.);
- устройства долговременного хранения информации.

2. Контроллеры (адаптеры) периферийных устройств, отвечающие за управление периферийными устройствами и обмен между ними и ядром ЭВМ.

3. Внешние параллельные и последовательные каналы обмена и соответствующие контроллеры этих каналов.

4. Системные шины и их контроллеры.

5. Контроллеры прерываний и прямого доступа к памяти (ПДП).

6. Подсистемы процессора, отвечающие за ввод/вывод, организацию ПДП и реакцию на прерывания и др.

Традиционно выделяют три основных варианта обмена процессора с внешними устройствами:

- программный обмен (program polling);
- обмен по прерываниям (interrupts);
- обмен в режиме прямого доступа к памяти - ПДП (direct memory access - DMA).

Наиболее наглядно и полно можно проследить и прочувствовать проблемы и тенденции развития систем ввода-вывода при рассмотрении ретроспективы эволюции интерфейсов и структур систем ввода-вывода на примере персональных компьютеров типа IBM PC

В начале эры персональных компьютеров частота работы процессора составляла 10 МГц, при этом на выполнение даже самых простейших операций процессор затрачивал несколько тактов. В таких условиях для обеспечения бесперебойной работы процессора было достаточно всего 4 миллионов обращений к памяти в секунду, что соответствовало циклу работы в 250 нс. Этим условиям удовлетворяла одношинная структура систем ввода-вывода, когда все устройства компьютера, включая ОЗУ, общались с процессором через общую шину (рис.5.1а), которую называли системной. Все интерфейсы ПУ подключа-

лись к этой шине. Наиболее распространенной системной шиной в этот период стала сначала 8 разрядная, затем 16 разрядная шина ISA, работающая на частоте 8 МГц.

С ростом частоты работы ПК и изменения времени доступа к ОЗУ пропускная способность шины ISA стала тормозить работу процессора. Решение проблемы нашли в выделении канала передачи данных МП-ОЗУ в отдельную шину, построенную на базе внешнего интерфейса МП, и изолированную от медленной шины ISA посредством контроллера шины данных. Это повысило производительность работы центрального процессора. Все ПУ продолжали взаимодействовать с центральным процессором через системную шину (рис.5.1б).

С дальнейшем ростом частоты работы МП тормозом в работе стало ОЗУ. Тогда ввели дополнительную высокоскоростную кэш-память, что уменьшило простои МП. На определенном этапе развития компьютеров стали широко использовать мультимедиа. Сразу выявилось узкое место во взаимодействии центрального процессора и видеокарты. Имеющиеся системные шины ISA, EISA не удовлетворяли этим условиям.

Выход был найден с разработкой и внедрением высокоскоростных локальных шин, посредством которых можно было связаться с памятью, на этой же шине работали жесткие диски, что также повышало качество вывода графической информации. Первой такой шиной была шина VL-bus, практически повторявшая интерфейс МП i486. Затем появилась локальная шина PCI. Она была процессорно - независимой и поэтому получила наибольшее распространение для последующих типов МП. Эта шина имела частоту работы 33 МГц и при 32-х разрядных данных обеспечивала пропускную способность в 132 Мбайт/сек (см. рис.5.1в). Системная шина ISA по-прежнему использовалась в компьютерах, что позволяло применять в новых компьютерах огромное количество ранее разработанных аппаратных и программных средств.

В такой системе ввода-вывода различные ПУ подключались к разным шинам. Медленные - к ISA, а высокоскоростные - к PCI. С появление шины PCI стало целесообразным использовать высокоскоростные параллельные и последовательные интерфейсы ПУ (SCSI, ATA, USB). На этом этапе системной стали называть шину МП, через которую он взаимодействовал с ОЗУ. Шина PCI и ISA и подобные другие назвали шинами ввода-вывода или шинами расширения. Действительно, эти шины как бы расширяли число устройств, работающих с ЦПр, и их основной функцией стало обеспечение процессов ввода и вывода информации.

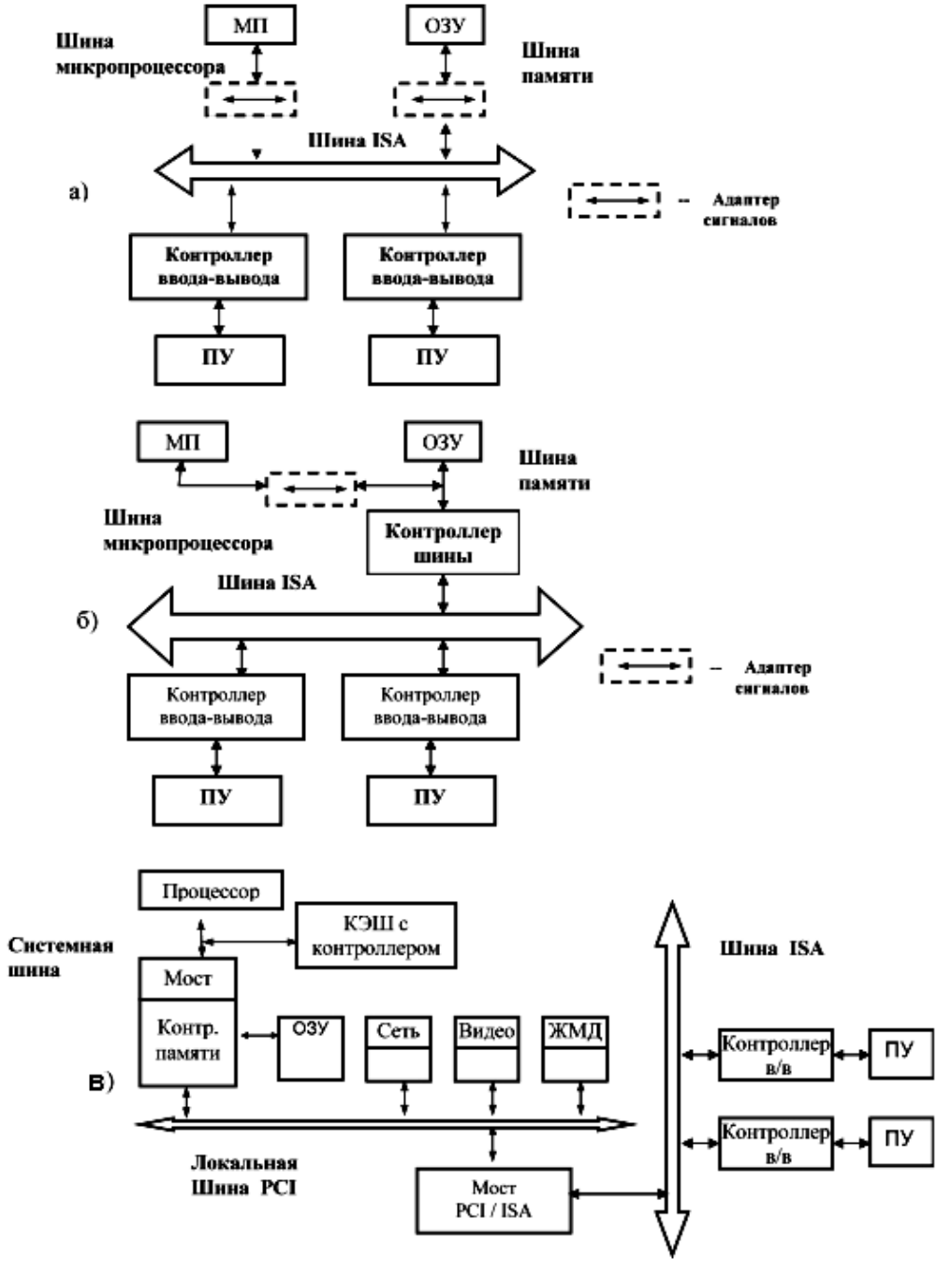


Рис. 5.1 Эволюция шинной архитектуры

Появление шины PCI не сняло всех проблем по качественному выводу визуальной информации для 3-х мерных изображений, "живого" видео. Здесь уже требовались скорости в сотни Мбайт/сек. В 1996г. фирма Intel разработала новую шину AGP, предназначенную

только для связи ОЗУ и процессора с видеокартой монитора. Эта шина обеспечивает пропускную способность в сотни Мбайт/сек. Она непосредственно связывает видеокарту с ОЗУ минуя шину PCI.

Таким образом, спустя годы снова пришли к многомагистральной структуре ввода-вывода с радиально-магистральными интерфейсами ПУ. Все шины систем ввода-вывода объединяются в единую транспортную среду передачи информации с помощью специальных устройств - мостов.

Мост – устройство, применяемое для объединения шин, использующих разные или одинаковые протоколы обмена. Мост – это сложное устройство, которое осуществляет не только коммутацию каналов передачи данных, но и производит управление соответствующими шинами. Для обеспечения выполнения функций интерфейсов, входящих в систему ввода-вывода, применяются специальные контроллеры и схемы. К ним можно отнести контроллеры прерываний и прямого доступа к памяти, таймер, часы реального времени, буферы шин данных, дешифраторы, мультиплексоры, регистры и другие логические устройства.

В первых компьютерах, построенных с использованием микропроцессоров, контроллер и другие устройства строились на базе набора интегральных схем малой, средней и большой степени интеграции. Адаптеры, таймер и др. выпускались в виде отдельных микросхем (8250, 8255, 8259, 8237 и т.д.)

С повышением производительности компьютеров и увеличением степени интеграции все вышеперечисленные устройства и схемы стали объединяться в микросхемы со сверхбольшой степенью интеграции, образуя специальные наборы интегральных схем, называемых «чипсет» (*ChipSet*).

В настоящее время управление потоками передаваемых данных производится с помощью мостов и контроллеров, входящих в ChipSet. Именно ChipSet определяет основные особенности архитектуры компьютера и, соответственно, достигаемый уровень производительности в условиях, когда лимитирующим фактором становится не процессор, а его окружение – память и система ввода-вывода. Принято называть две главные микросхемы южный мост и северный мост. Северный мост обслуживает системную шину, шину памяти, AGP и является главным контроллером PCI. Южный мост обслуживает работу с ПУ (шины PCI, IDE).

5.2. Структуры систем ввода-вывода.

Структура систем ввода-вывода представляет собой совокупность взаимосвязанных внутренних и внешних интерфейсов (шин), посредством которых все устройства (модули) объединены в единую систему, называемую компьютером.

Причем каждая шина имеет определенную скорость передачи информации, и к ней подсоединяются устройства с соответствующим быстродействием. Все шины, как правило, могут работать параллельно, обеспечивая высокую производительность вычислительной системы. Шины соединяются между собой с помощью специальных устройств – мостов.

Кроме того, в структуру систем ввода-вывода входят устройства управления шинами и схемы организации процессов передачи информации при различных режимах ввода-вывода.

В процессе развития вычислительной техники формировалась структура самого компьютера и его систем ввода-вывода, разрабатывались и внедрялись различные типы интерфейсов.

Из поколения в поколение менялась элементная база и архитектура компьютеров. К моменту появления микропроцессоров (МП), больших интегральных схем (БИС) и персональных компьютеров уже сформировались определенные принципы построения и структуры систем ввода-вывода.

В больших компьютерах класса «Мейнфрейм» (IBM-360/370, ЕС ЭВМ), работающих в мультипрограммном режиме и имеющих мощный процессор, большой емкости ОЗУ и много разнообразных ПУ, уже много лет успешно используется многомагистральная структура с выделенными каналами ввода-вывода и каскадно-магистральным подключением ПУ (см. рис. 5.2а). В таких машинах, как правило, используются специализированные каналы ввода-вывода: мультиплексный, работающий с медленными ПУ и селекторный, обслуживающий быстродействующие ПУ. Такая структура позволила максимально использовать вычислительную мощность компьютера за счет одновременного решения нескольких задач и параллельной работы процессора и каналов ввода-вывода.

В таких компьютерах аппаратно реализовывались все функции по управлению потоками данных. В них система ввода-вывода содержит оптимальный набор из нескольких типов интерфейсов. Высокоскоростные интерфейсы процессора и ОЗУ, через которые взаимодействуют основной процессор, специализированные процессоры, блоки оперативной памяти обеспечивают максимально эффективное использование процессорного времени. Интерфейсы ввода-вывода, аппаратно реализованные каналы ввода-вывода и контроллеры ПУ, освобождают центральный процессор от процедур управления вводом-выводом.

Интерфейсы ПУ предназначаются для подключения ПУ к компьютеру.

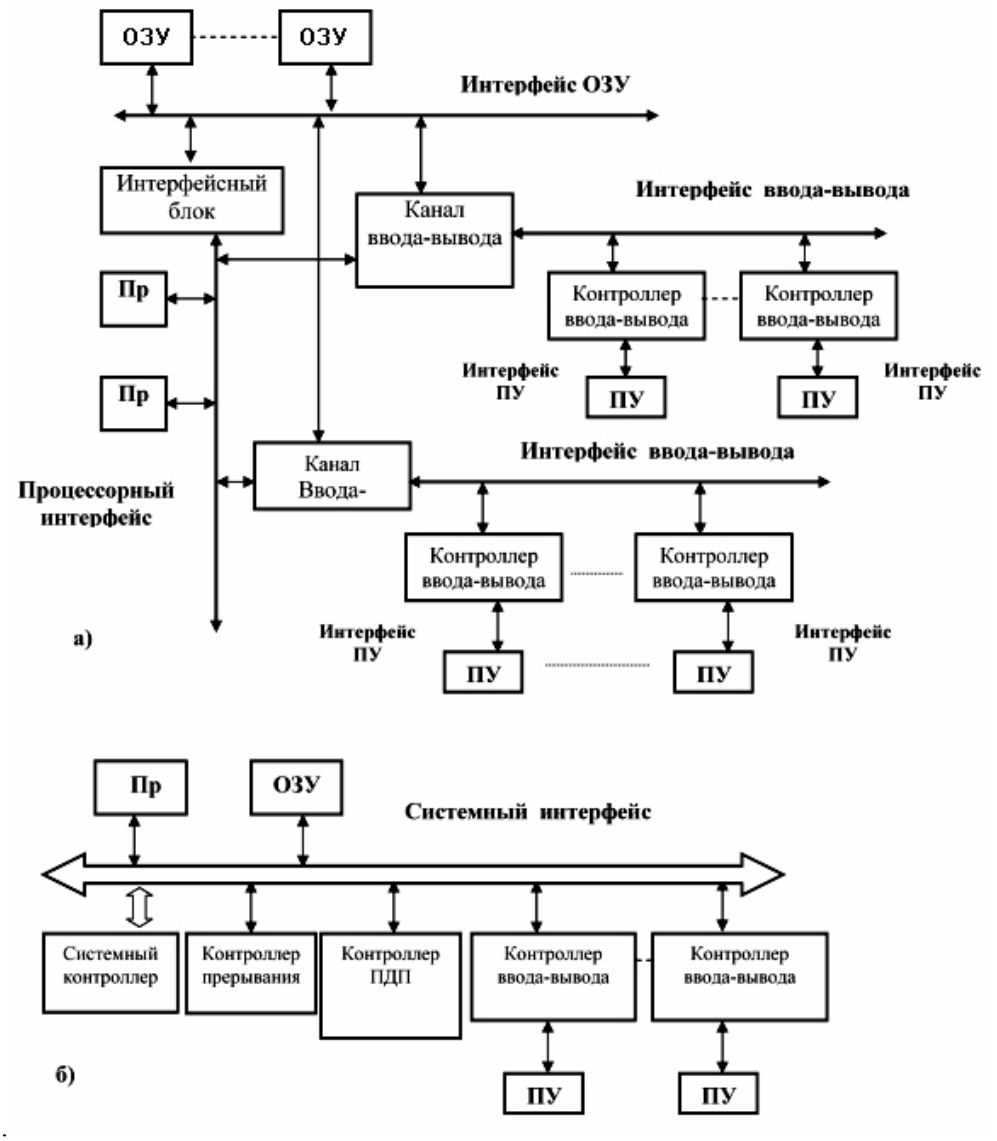


Рис. 5.2 Структура систем ввода-вывода

В то же время, система ввода-вывода малых вычислительных машинах типа DEC PDP-11, СМ ЭВМ, которые были намного дешевле больших, строилась по одномагистральной структуре с распределенным каналом ввода-вывода и радиально-магистральным подключением ПУ (см. рис.5.2б). Функции управления вводом-выводом выполнял процессор. В качестве высокоскоростного канала применялся контроллер прямого доступа к памяти. Существовали стандарты на используемые внутримашинные системные шины, например, Unibus фирмы DEC (отечественный аналог – «Общая шина»).

С появлением МП и БИС наступил новый этап развития структур систем ввода-вывода, обусловленный новым принципом построения вычислительных машин на основе модульности, микропрограммируемости и магистральности. Новый этап повторял стадии развития предыдущего, но на качественно новой элементной базе и других подходах к компоновке компьютеров.

С развитием элементной базы компьютеров, повышением скорости работы микропроцессоров и микросхем памяти, увеличением емкости ОЗУ совершенствовалась и изменилась структура системы ввода-вывода информации, повышалась скорость работы интерфейсов. Развитие интерфейсов и систем ввода-вывода было направлено на минимизацию потерь в производительности компьютера, вызванных задержками в передаче информации между его модулями (устройствами), т.е. передач процессор - ОЗУ, процессор - ПУ, ПУ – ОЗУ.

5.3 Основные режимы ввода-вывода.

Для учета особенностей реализации процессов ввода-вывода и специфики различного типа ПУ используются три режима ввода-вывода информации: программный ввод-вывод, ввод-вывод в режиме прерываний и с прямым доступом к памяти.

Интерфейсы должны учитывать возможность реализации всех 3-х режимов ввода-вывода.

Программный ввод-вывод. Здесь инициализация и управление процессом ввода-вывода осуществляет процессор. Существует три способа его выполнения (см. рис.5.3).

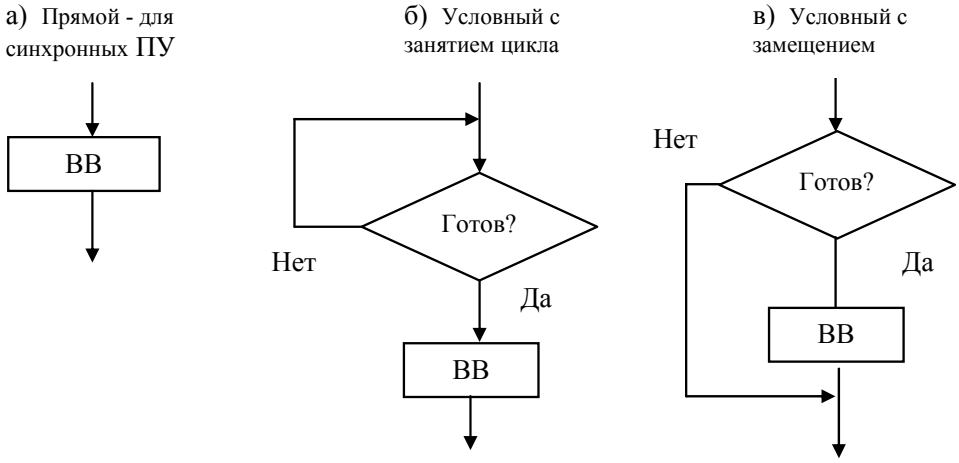


Рис.5.3 Программный ввод-вывод

Первый способ – прямой, используется для синхронных ПУ, т.е. устройств, которые всегда готовы к работе и циклов ожидания не требуется. Второй – условный с занятием цикла, когда при не готовности ПУ, процессор ждет до тех пор, пока наступит его готовность. Третий – условный с совмещением. В отличие от предыдущего, процессор не ждет готовности ПУ, а переходит к продолжению программы с периодической проверкой готовности ПУ.

Ввод-вывод в режиме прерываний. В этом случае инициатором начала процесса ввода-вывода является ПУ. Оно, когда готово, подает сигнал процессору "запрос на прерывание". Процессор, если ПУ разрешен такой режим, завершает текущую команду и переходит к выполнению процесса ввода-вывода (см. рис.5.4). Сначала он осуществляет контекстное переключение, т.е. запоминает свое состояние, чтобы можно было после продолжить программу, идентифицирует ПУ и

передает управление драйверу данного ПУ (ПП), который и осуществляет ввод или вывод информации. Идентификация ПУ производится с помощью адреса вектора прерывания, который содержит номер ячейки, где хранится первая команда этого драйвера. Адрес вектора прерывания ПУ передается процессору от контроллера прерываний.

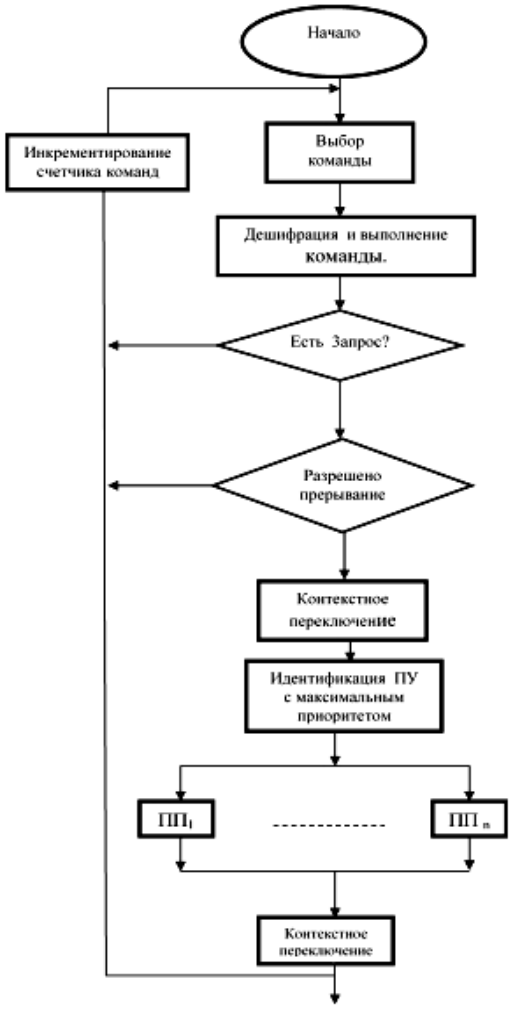


Рис.5.4 Ввод-вывод в режиме прерывания

Следует отметить два момента. Во-первых, ПУ должно иметь, предварительно установленное, разрешение на работу в режиме прерываний. Во-вторых, возможны коллизии, когда несколько ПУ выставляют процессору запрос прерывания. Эта коллизия разрешается с помощью механизма задания уровня приоритетов для каждого ПУ. Возможна организация вложенных прерываний, когда ПУ с большим приоритетом прерывает работу ПУ с меньшим приоритетом. Все эти моменты должен учитывать стандарт на интерфейс.

Прямой доступ к памяти (см. рис. 5.5). Этот режим используется для высокоскоростных ПУ.

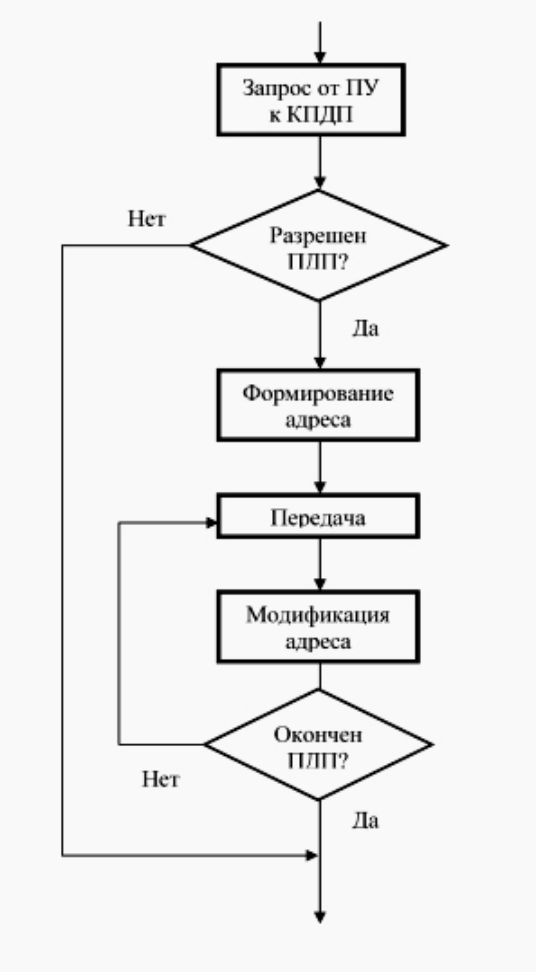


Рис. 5.5 Режим прямого доступа к памяти.

Процессор, получив от КПДП заявку на прямой доступ, прерывает свою работу и отключается от интерфейса, передавая его задатчику, т.е. КПДП. Процессор при этом не выполняет контекстного переключения, а может продолжать свою работу, если она не требует интерфейса. Управление интерфейсом переходит к КПДП, который посредством выполнения операций чтения и записи передает информацию между ОЗУ и ПУ с соответствующим заданием адресов памяти. В этом режиме используется механизм задания уровня приоритетов для тех ПУ, которые работают с прямым доступом к памяти. Этот режим также должен быть предусмотрен в интерфейсах.

Как следует из вышеизложенного, канал ввода-вывода (главный контроллер) реализует функции управления общие для всех ПУ, а контроллер внешнего интерфейса учитывает специфику интерфейса, связывающего его с соответствующим ПУ.

В компьютерах, которые работают с малой интенсивностью ввода-вывода, главный контроллер (канал) ввода-вывода обычно отсутствует, а его функции берет на себя процессор. В этом случае процессор работает непосредственно с контроллером ввода-вывода ПУ, что упрощает структуру компьютера.

5.4 Основные принципы организации передачи информации в вычислительных системах.

В процессе работы компьютера передача информации по одному и тому же интерфейсу в один и тот же момент времени идет только между двумя устройствами (*модулями*) по принципу «точка-точка». При этом одно из устройств является активным (*ведущим, задатчиком*), другое – пассивным (*исполнителем, ведомым*).

Активное устройство начинает процедуру обмена и управляет ею. Пассивное устройство выполняет предписания активного. В компьютере одни устройства всегда является задатчиками (*активными*), другие только исполнителями (*пассивными*), третьи в разные моменты времени могут быть как задатчиками, так и исполнителями. Процессор всегда активное устройство, оперативная память (*ОЗУ*) – пассивное устройство.

Периферийные устройства при работе с процессором являются исполнителями, а при работе с ОЗУ (прямой доступ к памяти) – задатчиками. Возможна передача между двумя ПУ, тогда одно – задатчик, другое – исполнитель (см. рис.5.6а).

Рис.5.6 Управление обменом информации

Таким образом, пассивными устройствами являются либо ОЗУ, либо ПУ. Со стороны процессора средства управления этими двумя типами устройств существенно различны. Это обусловлено тем, что для передач процессор-ОЗУ заранее известны все типы и параметры устройств, кото-

рые должны соединяться между собой, т.к. эти устройства однотипны, в то время как ПУ существенно различаются, как по задержке, так и по пропускной способности. Кроме того, процесс управления ПУ намного сложнее и требует большего времени и учета специфики работы.

Поэтому управление передачей процессор-ОЗУ реализуется в рамках одной компьютерной команды на уровне микрокоманд (см. рис. 4.6б), а управление процессом ввода-вывода с учетом специфики ПУ с помощью специальной подпрограммы, которая называется драйвером и содержит как команды компьютера, так и команды управления, специфичные для каждого типа ПУ (см. рис.4.6в). Поэтому интерфейс, связывающий устройства при передаче данных должен учитывать эти особенности.

Передача информации от задатчика к исполнителю реализуется операцией записи, а обратная – операцией чтения. Процесс передачи между ПУ и ЦПр называют вводом-выводом информации.

Ввод реализуется с помощью операции чтения, а вывод – операцией записи.

Если на процессор возложить функции управления вводом-выводом, то у него не хватит времени для выполнения своей главной функции – преобразования информации. Это обусловлено широким диапазоном скоростей работы ПУ, сложностью их управления и большим разнообразием и количеством.

6 ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

6.1. Классификация параллельных ВС

При построении параллельных вычислительных систем (ВС) разработчиками могут ставиться различные цели, такие, например, как:

- повышение производительности;
- улучшение показателя производительность / стоимость;
- повышение надежности функционирования (системы высокой готовности) и т.д.

Параллельные вычислительные машины и системы могут классифицироваться по различным признакам. К наиболее распространенным можно отнести следующие классификации:

1. По взаимодействию потоков команд (инструкций) и потоков данных.

Данная классификация предложена американским ученым Флинном (Flinn) в начале 70-х годов и используется до настоящего времени. Он предложил подразделять все ВС на 4 группы :

- ОКОД - Одиночный поток команд / Одиночный поток данных (SISD -Single Instruction / Single Data). Это ВС и ЭВМ обычного последовательного типа (фон-Неймановкой архитектуры). Для данных ЭВМ параллельная обработка реализуется в виде многозадачной обработки (системы с разделением времени и др.). При этом в данный момент времени ЦП или ОУ занято выполнением какой-то одной задачи.

- ОКМД - Одиночный поток команд / Множество потоков данных (SIMD -Single Instruction / Multiple Data). Такая архитектура характерна для векторных и матричных ВС, выполняющих специальные векторные и матричные операции как параллельные операции для разных потоков данных. Под потоками данных подразумеваются последовательности элементов векторов (для векторных ВС) или строки матриц (для матричных ВС). В последние годы SIMD-расширения реализованы в системах команд процессоров общего назначения (MMX, SSE, SSE2 - Intel, 3DNow! - AMD, AltiVec - Motorola и др.)

- МКОД - Множество потоков команд / Одиночный поток данных (MISD -Multiple Instruction / Single Data). Данная архитектура соответствует ВС конвейерного типа, в которых один поток данных проходит разные ступени обработки в разных процессорных элементах (ПЭ).

Архитектуры типа ОКМД и МКМД используются при построении высокопроизводительных систем разного уровня, начиная от простых конвейерных ВС до супер-ЭВМ с векторными и параллельными процессорами.

МКМД - Множество потоков команд / Множество потоков данных (MIMD - Multiple Instruction / Multiple Data). Такая архитектура характерна для ВС сверхвысокой производительности, в которых множество ПЭ, выполняющих каждый свою вычислительную подзадачу (процесс), обмениваются потоками команд и данных в разных направлениях (транспьютерные системы, системы с массовым параллелизмом и др. - рис. 6.1.) Помимо четырех выделенных групп, иногда выделяют дополнительные, находящиеся на границе между перечисленными, например, MSIMD или MMISD - соответственно Multi-SIMD, или Multi-MISD - системы с несколькими параллельно работающими SIMD или MISD - блоками.

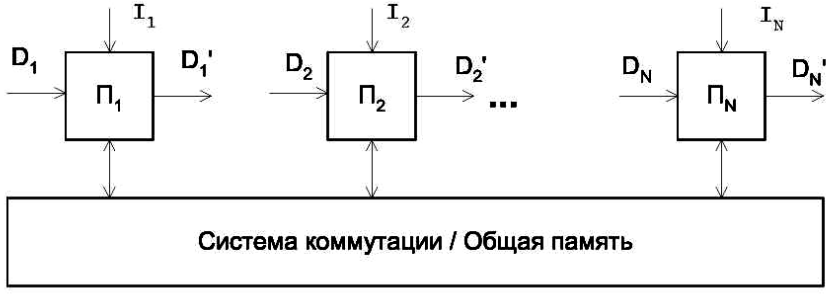


Рис 6.1

2. По управляющему потоку

управляемые потоком команд (IF- instruction flow) ;

- управляемые потоком данных (DF- dataflow) .

Системы с управлением потоком данных иногда называют просто потоковыми архитектурами .

3. По использованию памяти :

- с общей памятью («разделяемая память» - shared memory);
- с локальной памятью для каждого процессора.

Общая разделяемая разными процессорами память может быть физически распределена по вычислительной системе. С другой стороны, под распределенной памятью (distributed) понимают обычно систему, при которой каждый процессор имеет свою локальную память, с локальным адресным пространством. Для общей памяти, доступ к которой осуществляется разными процессорами в системе за одинаковое время, существует термин UMA -Unified Memory Access (память с одинаковым временем доступа). В случае, когда время доступа к разным адресам общей памяти для разных процессоров неодинаково (обычно это характерно для физически распределенной общей памяти), используют термин Non-UMA (память с различным временем доступа) .

4. По способу обмена между процессорами:

- через общую разделяемую память;
- через передачу сообщений.

5. По используемому типу параллелизма :

- естественный или векторный параллелизм (используется в векторных и матричных вычислительных системах);
- параллелизм независимых ветвей («крупнозернистый» или «Coarse Grain») - используется в симметричных многопроцессорных системах Symmetric MultiProcessor (SMP), а также в системах MIMD;
- «мелкозернистый» («Fine Grain») параллелизм - используется в многопроцессорных системах типа MIMD, в системах с массовым параллелизмом и др.;
- параллелизм смежных операций (Instruction Level Parallelism - ILP) - используется в ЭВМ с длинным командным словом - VLIW и др.

6. По способу загрузки данных:

- с последовательной загрузкой (последовательным кодом - по битам) ;
- с параллельной загрузкой (по словам);
- с последовательно-параллельной загрузкой.

7. По системе коммутации :

- полносвязанные МПВС (каждый процессор связан с каждым);
- с выделенным коммутатором ;
- с общей шиной ;

- линейный или матричный массивы (связаны друг с другом соседние процессоры) ;
- гиперкубы (связаны соседние ПЭ, но массивы многомерные);
- параллельные машины с изменяемой конфигурацией связей;
- с программируемыми коммутаторами .

8. По сложности системы коммутации (по кол-ву уровней иерархии) :

- с коммутирующей цепью (сетью) - один уровень коммутации ;
- с кластерной коммутацией (когда группы вычислительных устройств объединены с помощью одной системы коммутации, а внутри каждой группы используется другая).

9. По степени распределеннности ВС :

- локальные вычислительные системы ;
- вычислительные комплексы (в том числе - сильносвязанные - с обменом через общую память и слабосвязанные - с обменом через передачу сообщений).
- как вариант вычислительных комплексов - кластерные архитектуры;
- распределенные вычислительные комплексы, в том числе - сети ЭВМ и распределенные кластерные системы.

10. По организации доступа к общим ресурсам:

симметричные многопроцессорные системы (все процессоры имеют одинаковый доступ к общим ресурсам);

асимметричные (master-slave) многопроцессорные системы с разными возможностями доступа для разных процессоров.

6.2 Параллельные вычислительные системы типа SIMD. Векторные ВС.

К ВС типа SIMD относят, прежде всего**, *ассоциативные*** и ***векторные ВС. Ассоциативные*** ВС строятся, как можно заключить из названия, на базе ассоциативной памяти. Как уже отмечалось ранее, при рассмотрении организации систем памяти, для ассоциативной памяти характерны аппаратные механизмы поиска информации с заданным адресным признаком по различным мерам сходства и т.н. вертикальная обработка. В ассоциативных вычислительных системах помимо поиска могут аппаратно реализовываться различные механизмы обработки данных, применяемые к данным с подходящими тегами с использованием все той же вертикальной обработки, то есть - параллельно захватывающие все ячейки памяти, или по крайней мере - все строки массива накопителя ассоциативной памяти.

Тем не менее, чаще всего ассоциативные вычислительные системы строятся для решения задач невычислительного характера - поиска, распознавания и т.д. Примером могут

служить различные вычислительные системы для поддержки баз данных - в частности, реляционный ассоциативный процессор (RAP). Существуют и проекты собственно вычислительных ассоциативных систем -например, известная ВМ "STARAN".

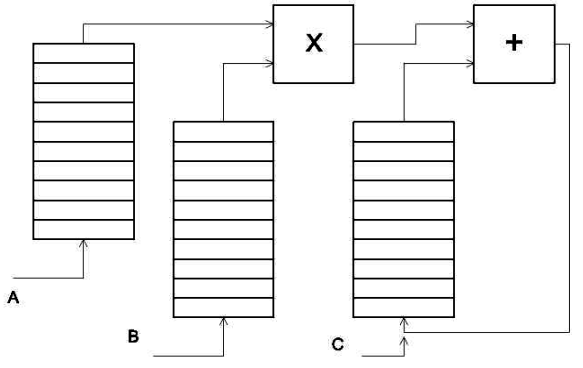
К ***векторным*** относят системы, включающие в свои системы команд специальные векторные (матричные) операции, такие, как векторное и матричное сложение, умножение вектора на матрицу, умножение матрицы на константу, вычисление скалярного произведения, свертки и т.д. Такие ВС относят к классу SIMD, иногда - MISD. Такая неоднозначность связана с наличием двух основных видов векторных систем :

- В векторно-конвейерных системах в основе ускорения вычислений лежит принцип конвейерной обработки последовательности компонент векторов и матриц.
- В векторно - параллельных системах различные компоненты векторов и матриц при выполнении векторных операций обрабатываются параллельно на параллельно работающих ПЭ.

Векторно-параллельные ВС иногда называют «настоящими SIMD», тем самым подчеркивая, что в векторно-конвейерных ВС имеется несколько иной механизм взаимодействия потоков команд и данных, больше характерный для MISD. В то же время логически или функционально и в векторно-конвейерных системах фактически реализуется принцип SIMD (одна инструкция - например, сложение, выполняется для нескольких потоков данных, но только эти потоки как бы выстраиваются в одну очередь на конвейере).

Например, если необходимо находить свертку, то есть вычислять выражение вида:

$$C = \sum_i a_i \cdot b_i ,$$



то можно использовать два арифметических конвейера - для умножения и для сложения (сложение тоже можно конвейеризовать, см. конвейеризацию параллельного сумматора на заключительном этапе формирования произведения в умножителе Брауна), на который подаются последовательно элементы обрабатываемых векторов из векторных регистров (рис. 6.2). Арифметические устройства сами по себе могут быть и неконвейеризованными, тогда либо реализуется конвейер из трех операций: умножение, сложение и запись в ре-

гистр (память), либо - три указанные операции просто выполняются как одна макрооперация, которая называется «зацеплением», причем аппаратно ускоряется ее вычисление и подготовка следующего зацепления по сравнению с обычными процессорами общего назначения. Часто зацепление реализуется с использованием трех банков оперативной памяти, а не регистров, поскольку векторные машины должны работать с векторами и матрицами большой размерности, которые не всегда можно разместить в векторных регистрах.

Для ускорения работы с памятью используют различные механизмы адресации, операции с автоинкрементом (автодекрементом) адреса, механизмы ускоренной выборки и записи (многопортовая память, память с расслоением и т.д.), отдельное адресное обрабатывающее устройство (что характерно для так называемой разнесенной архитектуры). Для выполнения скалярных операций в комплексе с векторным обрабатывающим устройством в векторной машине может использоваться скалярное устройство.

Таким образом, для векторно-конвейерных машин характерно :

1. Поддержка специальных векторных, матричных операций в системе команд.
2. Ускорение обработки векторов за счет конвейеризациии выборки и собственно обработки в конвейерных исполнительных устройствах..
3. Наличие векторных регистров.
4. Развитые механизмы адресации и выборки/записи в память.
5. Сочетание векторных и скалярных регистров и обрабатывающих устройств для эффективной реализации алгоритмов, требующих выполнения как векторных, так и скалярных вычислений.

Наряду с конвейеризацией операций в векторных ВС используется и конвейеризация команд, что требует такого построения системы команд векторной машины, чтобы команды легко могли выполняться на конвейере. Поэтому многие векторные процессоры имеют RISC-подобные команды.

Примерами векторно-конвейерных машин могут служить классические супер-ЭВМ серии Cray : Cray-1, Cray-2, Cray - X-MP, Cray - Y- MP и др. Примером векторно-параллельных машин могут служить машины ILIAC-IV, Cyber-205, отечественные супер-ЭВМ серии ПС2000.

Недостатком векторно-параллельных машин является сравнительно низкая эффективность в смысле загрузки процессорных элементов. Высокая производительность достигается только на векторных операциях, в то время как на скалярных операциях и при обработке векторов и матриц меньшей размерности значительная часть устройств может простаивать. В конвейерных ЭВМ при обработке векторов меньшей размерности конвейер, возможно, будет загружен полностью, так как меньшая размер-

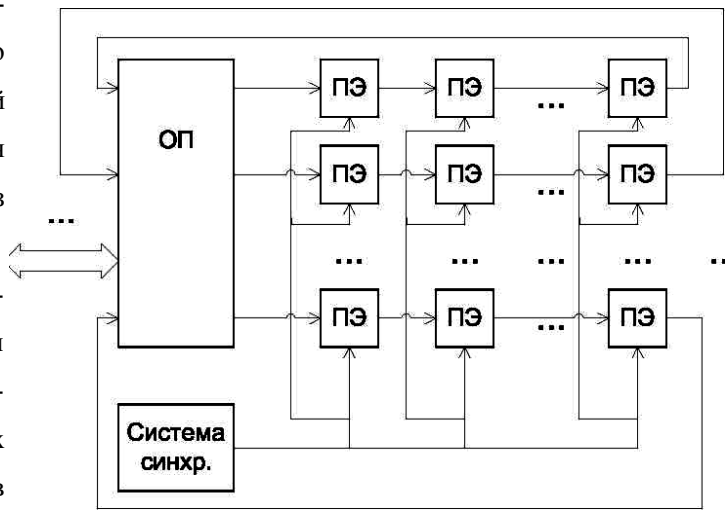
ность может компенсироваться большей интенсивностью следования последовательных по алгоритму векторных подзадач.

Кроме того, программирование векторно-параллельных ЭВМ осуществляется в целом сложнее, чем для векторно-конвейерных. Для загрузки и выгрузки данных требуется больше времени, чем в случае конвейерных систем, когда данные могут поступать последовательно. При этом, преимуществом векторно-параллельных машин является их потенциально более высокая производительность.

В целом векторные машины характеризуются высокой пиковой производительностью при полной загрузке их вычислительных устройств

6.3 Понятие о систолических структурах и алгоритмах

Под *систолической структурой* можно понимать массив (сеть) вычислительных «клеток» (ПЭ), связанных каналами обмена. Число соседних клеток ограничено. Каждая клетка работает по своей программе (что характерно для систем МКМД). Обмен данными и управляющими сигналами осуществляется только с соседними клетками. Обмен с оперативной памятью происходит только на границе массива клеток (рис. 6.3). Термин «систолическая» (systolic) указывает на синхронность, непрерывность и волнообразность продвижения обрабатываемых данных от одной границы массива к другой (систола это сердечная мышца, которая работает также синхронно, ритмично и без остановок в ходе всей жизни человека, систолическая система «прокачивает» через себя обрабатываемые данные).



Особенностью работы систолических структур является отсутствие накопления информации в локальных блоках памяти и возвратов потоков данных назад для циклической обработки. Упорядоченность этапов обработки информации указывает на родство с конвейерными архитектурами типа MISD, поэтому многие исследователи относят систолические системы именно к этому классу. В то же время наличие различных потоков команд и данных в системе и разных программ у разных вычислительных клеток указывает на родство с MIMD-архитектурами. В отличие от MIMD характер обмена носит заданный, однонаправленный характер, отсутствие циклов, возвратов, длинных пересылок, а также относительная простота задач, ре-

шаемых каждой вычислительной клеткой, позволяет говорить о своеобразии систолических структур.

Областью применения таких структур, прежде всего, являются многопроцессорные специализированные структуры для цифровой обработки сигналов, изображений, для решения матричных задач.

Для эффективной реализации вычислений в систолической структуре необходимы так называемые *систолические алгоритмы,* рассчитанные на аппаратную систолическую реализацию. Они должны удовлетворять определенным требованиям, среди которых :

1) Регулярность, однонаправленность графа вычислений (потокового графа) алгоритма.

2) Ацикличность алгоритма.

3) Возможность разбиения алгоритма на этапы одинаковой сложности и длительности выполнения для построения конвейера.

4) Возможность распараллеливания вычислений.

5) Отсутствие необходимости в больших объемах памяти для сохранения промежуточных результатов и накопления информации.

6) Локальность пересылок информации, отсутствие необходимости в длинных пересылках.

7) Минимальное количество развилок в алгоритме и т.д.

8) Минимальное количество входных и выходных точек алгоритма.

9) Минимальное количество разных типов вычислений и операций, используемых в алгоритме.

10) Возможность разбиения алгоритма на подалгоритмы меньшей размерности, и с другой стороны - наращивания алгоритма для решения задач большей размерности.

11) Гарантированная сходимость вычислений за заданное число шагов (итераций) и др.

Примером систолических алгоритмов являются алгоритмы CORDIC и родственные ему (так называемые ДЛП-алгоритмы или CORDIC-подобные), другие итерационные алгоритмы, алгоритмы обработки матриц, оптимизированные для аппаратной реализации и так далее. Рассмотренные ранее аппаратные умножители также являются примерами систолических структур.

6.4 Масштабируемые параллельные системы МКМД.

Современные параллельные вычислительные системы, включая аппаратные и программные средства, называются *масштабируемыми* (scalable), если их ресурс может быть изменен в зависимости от требований производительности/стоимости. Это понятие включает в себя следующие составляющие:

- Функциональность и производительность. Масштабируемая система должна повышать (понижать) свою производительность пропорционально повышению (понижению) своих ресурсов. В идеале эта зависимость должна быть линейной.
- Стоимость. Стоимость системы при масштабировании должна меняться пропорционально (но не быстрее, чем линейно).
- Совместимость. Одни и те же компоненты системы, включая аппаратные и программные средства, должны использоваться после масштабирования без больших изменений.

В простейшем случае масштабируемость системы может быть достигнута за счет изменения количества модулей - процессоров или вычислительных машин, входящих в нее. Масштабируемость характеризуется своим размером-максимальным количеством вычислительных модулей, которое может быть включено в нее без нарушения ее работоспособности. Например, Симметричная мультипроцессорная Система (SMP, 1997) имеет предел до 64 процессоров, а система IBM SP2 (1997) - до 512 процессоров.

В настоящее время наибольшее распространение получили пять вариантов МКМД систем:

- Системы с массовым параллелизмом (massively parallel processor- MPP);
- Симметричные мультипроцессорные системы (SMP);
- Кластеры рабочих станций (cluster of workstations-COW);
- Системы с распределенной памятью (distributed shared memory-DSM);
- Систолические структуры.

Системы с массовой параллельной обработкой (MPP) включают тысячи, десятки, иногда - сотни тысяч процессорных элементов. Подобные системы могут иметь различную организацию. Одни системы относят к категории SIMD с мелкозернистым параллелизмом. Они работают под управлением специальных управляющих ЭВМ, раздающих задания процессорам и контролирующих их коммутацию и обмен. Сами процессорные элементы могут быть достаточно простыми и иметь невысокую производительность. В частности, в одной из первых ЭВМ такого типа - Connection Machines - 1 (CM-1) использовались однобитные АЛУ в которых 32-битовое сложение выполнялось за 24мкс! АЛУ объединялись в ПЭ, состоявший из 16 таких ОУ и 4 блоков локальной памяти. Про-

изводительность 1 ПЭ - около 2MIPS. Количество процессорных элементов достигало 64 тысяч, разделенных на 4 блока, каждый блок управлялся ЭВМ-секвенсором, отвечавшей за обмен между блоками, а управление всей системой осуществляла фронтальная ЭВМ, транслировавшая высокоуровневый поток команд в поток инструкций для ПЭ. Одна фронтальная ЭВМ могла управлять до 2 млн. процессорных элементов ! Система СМ-1 достигала производительности в 100 Gflops (1984 г), а система Connection Machines - 5 с 256 тыс. процессоров -уже до 1 TFlops (1991 г).

В других системах с массовой параллельной обработкой процессорные элементы представляют собой достаточно мощные и автономные вычислительные устройства (например, высокопроизводительные процессоры общего назначения), выполняющие собственные программы вычислений. В таких системах может использоваться кластерная организация.

Важную роль в эффективной реализации вычислений в таких системах (как и в параллельных вычислительных системах вообще) играют организация обмена информацией между процессорными элементами, организация памяти и, конечно, программное обеспечение.

Наиболее естественным кажется использовать для обмена общую память (shared memory). При этом адресное пространство памяти доступно всем процессорам, которые помещают результаты своей работы, а также - считывают исходную информацию из общедоступного ресурса (в зависимости от характера и объемов обрабатываемой информации она может храниться в оперативной памяти, либо - во внешней). Как уже отмечалось ранее, общая память может быть физически распределенной (Non-UMA), либо - с одинаковым временем доступа (UMA). Проблемы возникают при синхронизации работы с общими областями памяти (здесь задействуются механизмы семафоров, транзакций, атомарных операций и т.д.), при организации виртуальной памяти, при ускорении доступа к памяти в физически распределенной системе памяти и так далее. Дополнительную сложность представляет синхронизация работы разделяемой оперативной памяти и локальной кэш-памяти процессоров (проблема когерентности кэш-памяти). Синхронизация достигается применением нетривиальных алгоритмов, например, MESI. Для упрощения синхронизации памяти и для ускорения обмена в целом предпочтительнее обмен крупными блоками данных, выполняемый реже, по сравнению с частым обменом мелкими порциями данных. К другим способам ускорения обмена с памятью в многопроцессорных системах можно отнести коммутацию типа «точка-точка», при которой память разбивается на банки, каждый из которых связан с каждым процессором через интеллектуальный коммутатор.

При использовании логически распределенной между процессорами памяти каждый процессор работает со своим адресным пространством, а обмен информации происходит путем передачи сообщений между процессорами. При этом аппаратная организация параллельной системы упрощается, для построения мультипроцессорных систем подойдут даже стандартные телекоммуникационные средства, например, аппаратура и протоколы локальных сетей. С другой стороны, программирование таких систем усложняется, поскольку синхронизация сообщений, управления, потоков данных выполняется в основном программными средствами. В настоящее время используются в основном два базовых программных интерфейса параллельных систем с передачей сообщений : PVM (Parallel Virtual Machine) и MPI (Message Processing Interface). В целом можно отметить, что степень «зернистости» при разбиении задачи в мультипроцессорных системах может варьироваться из соображений затрат на собственно вычисления и обмен между вычислителями. В некоторых случаях потенциально высокая степень параллелизма, диктующая мелкозернистость, входит в противоречие с большими накладными расходами на организацию такой мелкозернистости, связанными с обменом между процессорами, и тогда меньшее дробление задачи оказывается более предпочтительным.

Среди мультипроцессорных систем большое распространение получили архитектуры, называемые *кластерными.*

Термин «кластерная архитектура» трактуется в настоящее время достаточно широко. Ключевым является понятие кластера как группы каких-то относительно самостоятельных, но тесно взаимодействующих устройств. С этой точки зрения под кластерами понимают, во-первых, вычислительные системы, построенные по иерархическому принципу, использующие кластерную коммутацию, во-вторых - группы автономных вычислительных машин, работающих под управлением общих вычислительных программ (вычислительные кластеры, в том числе - на базе сегментов локальных сетей), в-третьих - серверные системы высокой готовности, включающие несколько (по крайней мере - 2) автономных подсистем и т.д.

Кластеры реализуются как системы с передачей сообщений. Кластеризация позволяет упростить организацию обмена и в целом повысить эффективность системы за счет разумной иерархии и оптимального разделения задач по группам вычислителей.

С точки зрения вычислительной мощности простейшие кластеры как вычислительные комплексы на базе автономных ВМ (в том числе - ПК) могут составлять конкуренцию дорогим вычислительным системам, при этом обладая существенным преимуществом - они намного дешевле. Здесь на первый план выходит эффективное программное обеспе-

чение для параллельной обработки. Чаще всего подобные вычислительные кластеры строятся на базе систем, работающих под управлением ОС UNIX.

Анализ списка 500 самых производительных вычислительных машин, который публикуется каждые 3 месяца в Интернете (адрес:http://www.top500.org), показывает, что верхние строчки списка, как правило, занимают мультипроцессорные MIMD - системы. Например, это компьютеры, строящиеся в рамках программы ASCI (Advanced Strategic Computer Initiative -Стратегическая компьютерная инициатива). Они включают, обычно, десятки тысяч процессоров (общего назначения, либо - специализированных), объединенных в группы, с развитой иерархической системой коммутации. Управляются такие машины специализированным программным обеспечением для поддержки параллельных вычислений.

6.5 Потоковые вычислительные системы

Под потоковыми ВС понимают обычно ВС, управляемые потоком данных (Data Flow). Это довольно интересный класс вычислительных систем, в которых очередной поток вычислений в соответствии с алгоритмом инициируется не очередными инструкциями программы, а готовностью к обработке необходимых данных. С каждой операцией в программе связан набор ее операндов, которые снабжаются флагами готовности к обработке. При установке всех флагов операция отправляется на выполнение. Потенциально такой подход позволяет достичь высокой степени параллелизма, так как потоковая архитектура пытается извлечь максимальный параллелизм из всех заданных вычислений.

Эффективность подобных структур во многом определяется эффективным программированием, задачей которого является формулировка задачи в терминах параллельных и независимых операций. На первый план выходит не эффективное построение процедуры вычислений, а выявление взаимосвязей между потоками данных в задаче. В значительной степени это задача соответствующего оптимизирующего компилятора.

Известно много проектов потоковых вычислительных систем, среди них можно отметить Манчестерскую ВС (Манчестерский университет), Tagged Token, Monsoon (Массачусетский технологический институт), Sigma, EMS, EMC-4 (Тсукуба, Япония), RAPID (проект Sharp - Mitsubishi - университет Осаки, Япония) и др. Как утверждает ряд авторов, многие интересные проекты Data Flow не были по достоинству оценены производителями вычислительной техники, в силу нетрадиционности подхода.

С другой стороны, потоковые вычисления нашли применение в современных суперскалярных процессорах и процессорах с длинным командным словом. Причем если в VLIW - структурах задача выявления параллельных потоков в большей степени ре-

шается программным обеспечением, то в суперскалярных процессорах логика управления многопотоковым конвейером как раз и реализует механизм, напоминающий управление потоком данных, но средствами самого процессора. Действительно, последовательный поток инструкций программы в суперскалярном процессоре транслируется в параллельные потоки внутренних инструкций, операнды которых снабжаются признаками готовности, которые, в том числе, зависят и от работы механизма динамической оптимизации команд.

Так что системы, управляемые потоком данных, так или иначе, находят применение в современной вычислительной технике.

Литература:

1. Пятибратов А. Вычислительные системы, сети и телекоммуникации - М., Финансы и статистика, 2002.
2. Каган Б.М. Электронные вычислительные машины и системы.- М.: Энергоатомиздат, 1991.
3. Нортон П., Гудмен Д. Внутренний мир персональных компьютеров, 8-е издание. Избранное от Питера Нортона: Пер. с англ. - К.: Издательство "ДиаСофт", 1999.-584 с.
4. Таненбаум Э.С. Архитектура компьютера, 4-е издание - С-Пб.:"Питер-пресс", 2002. -704с.
5. Столингс У. Структурная организация и архитектура компьютерных систем, 5-е изданеие. - М.: Изд. дом Вильямс, 2002. - 896с.
6. Корнеев В.В., Киселев А.В. Современные микропроцессоры. - М.: "Нолидж", 2000.- 320с., ил.
7. Корнеев В.В. Параллельные вычислительные системы. - М.: "Нолидж", 1999.- 320с., ил.
8. .Егунов В.А. Системы памяти. Учебное пособие; ВолгГТУ, 2000 г. 16.Murdocca M., Heuring V. Principles Of Computer Architecture, 1999., Prentice
9. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем. Учебник для вузов. – СПб.: Питер, 2004. - 668 с.
10. Хамахер К., Вранешич З., Заки С. Организация ЭВМ. 5-е изд. – СПб., Питер, 2003. – 848 с.
11. М. Гук, Аппаратные средства IBM PC. Энциклопедия. – СПб., Питер, 2001. – 928 с.
12. Павлов В.П. Организация ЭВМ и систем. – Самара: СГАУ, 2000. –182с.

Дополнительная литература

1. М. Гук. Процессоры intel от 8086 до Pentium 2. Архитектура, интерфейс, программирование. – СПб.: Питер, 1997. – 222 с.
2. Фролов А.В., Фролов Г.В. Аппаратное обеспечение персонального компьютера. М.: Диалог – Мифи, 1997.- 304 с.

3. М. Гук. Процессоры Pentium 2, Pentium Pro и просто Pentium. Архитектура, интерфейс, программирование. – СПб.: Питер, 1999.– 288 с.
4. Корнеев В.В., Киселев А.В. Современные микропроцессоры. – М.: Нолидж, 2000. – 320с.