А.Д. Чередов

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Учебное пособие

Томск 2000

У ДК 681.3

Чередов А.Д. Организация ЭВМ и систем: Учеб. пособие. — Томск: Изд. ТПУ, 2000. — 136 с.

В учебном пособии рассматриваются основные вопросы, связанные с организацией ЭВМ и систем: архитектуры, характеристики и классификация ЭВМ; функциональная и структурная организация ЭВМ и центрального процессора; принципы организации подсистемы памяти ЭВМ и вычислительных систем; принципы организации подсистемы ввода-вывода; архитектуры и способы организации многопроцессорных вычислительных систем.

Учебное пособие подготовлено на кафедре вычислительной техники ТПУ и предназначено для студентов специальностей 220100 «Вычислительные машины, комплексы, системы и сети» и 071900 «Информационные системы (в нефтегазодобыче)» Института дистанционного образования 'ГНУ.

Печатается по постановлению Редакционно-издательского совета Томского политехнического университета.

Рецензенты:

В. П. Бондаренко - д.т.н., профессор кафедры комплексной информационной безопасности электронных вычислительных систем, ТУСУР;

В. С. Фофонов - К.Т.Н., с.н.с., заведующий отделом НИИ АЭМ.

ВВЕДЕНИЕ

По принципу действия вычислительные машины делятся на три больших класса: аналоговые (АВМ), цифровые (ЦВМ) и гибридные (ГВМ).

Критерием деления вычислительных машин на эти три класса является форма представления информации, с которой они работают. АВМ - вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (аналоговой форме), т.е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения). ЦВМ - вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в цифровой форме. Аналоговые вычислительные машины весьма просты и удобны в эксплуатации; программирование задач для решения на них, как правило, нетрудоемкое; скорость решения задач изменяется по желанию оператора и может быть сделана сколь угодно большой, но точность решения задач очень низкая (относительная погрешность 2-5 %). На АВМ наиболее эффективно решать математические задачи, содержащие дифференциальные уравнения, не требующие сложной логики. ГВМ - вычислительные машины комбинированного действия, работают с информацией, представленной в цифровой и в аналоговой форме; они совмещают в себе достоинства АВМ и ЦВМ. Гибридные вычислительные машины целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

В последнее время широкое распространение получили ЦВМ с электрическим представлением цифровой информации, обычно называемые просто электронными вычислительными машинами (ЭВМ), без упоминания об их цифровом характере. В дальнейшем речь будет идти именно о таких вычислительных машинах.

Электронная вычислительная машина, компьютер - комплекс технических и программных средств, предназначенных для автоматической обработки информации в процессе решения вычислительных и информационных задач.

Под системой понимают любой объект, который одновременно рассматривается и как единое целое, и как объединенная в интересах достижения поставленных целей совокупность разнородных элементов.

Вычислительная система — взаимосвязанная совокупность средств вычислительной техники, включающая не менее двух основных процессоров либо вычислительных машин. Основным процессором называют составную часть ЭВМ, которая выполняет вычисления, предусматриваемые алгоритмами решаемых задач.

Информационная система - взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

В дальнейшем под термином «система» будем понимать вычислительную систему.

1. АРХИТЕКТУРЫ, ХАРАКТЕРИСТИКИ, КЛАССИФИКАЦИЯ ЭВМ

Под **архитектурой** ЭВМ понимается общая функциональная и структурная организация машины, определяющая методы кодирования данных, состав, назначение, принципы взаимодействия технических средств и программного обеспечения.

Можно выделить следующие важные для пользователя **группы** характеристик ЭВМ, определяющих ее архитектуру:

а) характеристики машинного языка и системы команд (количество и номенклатура команд, их форматы, системы адресации, наличие программно-доступных регистров в процессоре и т.п.), которые определяют алгоритмические возможности процессора ЭВМ;

б) технические и эксплуатационные характеристики ЭВМ;

в) характеристики и состав модулей базовой конфигурации ЭВМ;

г) состав программного обеспечения ЭВМ и принципы его взаимодействия с техническими средствами ЭВМ.

1.1. Однопроцессорные архитектуры ЭВМ

Исторически первыми появились однопроцессорные архитектуры. Классическим примером однопроцессорной архитектуры является архитектура фон Неймана со строго последовательным выполнением команд: процессор по очереди выбирает команды программы и также по очереди обрабатывает данные. По мере развития вычислительной техники архитектура фон Неймана обогатилась сначала конвейером команд, а затем многофункциональной обработкой и по классификации М.Флина получила обобщенное название SISD (Single Instruction Single Data — один поток команд, один поток данных). Основная масса современных ЭВМ функционирует в соответствии с принципом фон Неймана и имеет архитектуру класса SISD.

Данная архитектура породила CISC, MSC и архитектуру с суперскалярной обработкой (рис. 1.1).

Компьютеры с CISC (Complex Instruction Set Computer) архитектурой имеют комплексную (полную) систему команд, под управлением которой выполняются всевозможные операции типа «память-память», «память-регистр», «регистр — память», «регистр — регистр».

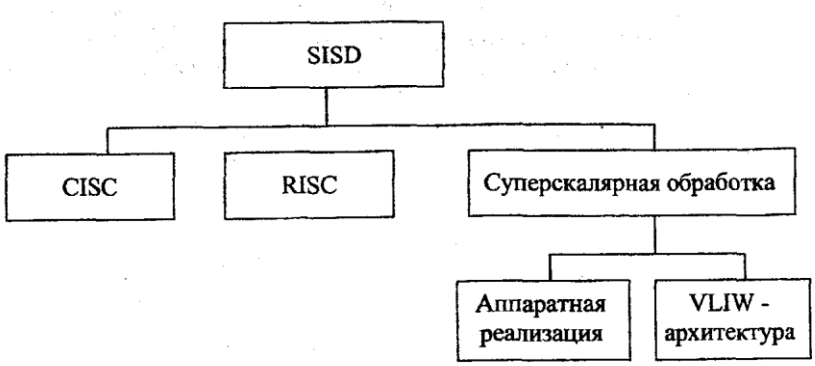


Рис. 1.1. Классификация архитектуры SISD

Данная архитектура характеризуется:

- большим числом команд (более 200);
- переменной длиной команд (от 1 до 11 байт);
- значительным числом способов адресации и форматов команд;
- сложностью команд и многотактностыо их выполнения;
- наличием микропрограммного управления, что снижает быстродействие и усложняет процессор.

Обмен с памятью в процессе выполнения команды делает практически невозможной глубокую конвейеризацию арифметики, т.е. ограничивается тактовая частота процессора, а значит, и его производительность.

Большинство современных компьютеров типа IBM PC относятся к CISC архитектуре, например, компьютеры с микропроцессорами (МП) 8080, 80486, 80586 (товарная марка Pentium).

Компьютеры с RISC (Reduced Instruction Set Computer) архитектурой содержат набор простых, часто употребляемых в программах команд. Основными являются операции типа «регистр — регистр».

Данная архитектура характеризуется:

- сокращенным числом команд;
- тем, что большинство команд выполняется за один машинный такт;
- постоянной длиной команд;
- небольшим количеством способов адресации и форматов команд;
- тем, что для простых команд нет необходимости в использовании микропрограммного управления;
- большим числом регистров внутренней памяти процессора.

Компьютеры с RISC-архитектурой «обязаны» иметь преимущество в производительности по сравнению с CISC компьютерами, за которое приходится расплачиваться наличием в программах дополнительных команд обмена регистров процессора с оперативной памятью.

В настоящее время практически все ведущие производители компьютеров прилагают большие усилия для производства RISC-процессоров (см. табл. 1.1).

Таблица 1.1

Название фирм и разработанных ими RISC-процессоров

Фирма	RISC-процессор
Sun Microsystems	Ultra SPARC II
	Ultra SPARC III
DEC	Alpha 21164
HP	PA-7150, PA-800
SGI	MIPS R-10000
IBM	PPC-601, PPC-60
Motorola	88000

Еще одной разновидностью однопотоковой архитектуры является **суперскалярная обработка.**

Смысл этого термина заключается в том, что в аппаратуру процессора закладываются средства, позволяющие одновременно выполнять две или более скалярные операции, т.е. команды обработки пары чисел. Суперскалярная архитектура базируется на **многофункциональном параллелизме** и позволяет увеличить производительность компьютера пропорционально числу одновременно выполняемых операций. Способы реализации суперскалярной обработки могут быть разными. **Первый способ** применяется как в CISC, так и в RISC - процессорах и заключается в чисто аппаратном механизме выборки из буфера инструкций (или кэша инструкций) несвязанных команд и параллельном запуске их на исполнение.

В табл. 1.2 для различных типов процессоров приведено максимальное и среднее число команд, выполняемых в одном машинном цикле.

Этот метод хорош тем, что он «прозрачен» для программиста, составление программ для подобных процессоров не требует никаких специальных усилий, ответственность за параллельное выполнение операций возлагается в основном на аппаратные средства.

Таблица 1.2

Максимальное и среднее число команд, выполняемых в одном машинном цикле

Процессор	Тактова	Число тр	Максимал на цикл	Среднее чи
Digital Alpha	500	9,3	4	1,0
Power PC 62	200	6,9	4	1,8
Power PC 60	225	5,1	4	1,5
Ultra SPARC	250	3,8	4	1,36
HP PA-8000	180	3,9	4	2,4
HPPA-7300I	160	*9,2*	2	1,35
Mips R1000	200	5,9	4	1,78
Mips R 5000	180	3,6	2	0,89
i486	25	1,2	-	0,45
Pentium Pro	200	5,5	3	1,76

Второй способ реализации суперскалярной обработки заключается в кардинальной перестройке всего процесса трансляции и исполнения программ. Уже на этапе подготовки программы компилятор группирует несвязанные операции в пакеты, содержимое которых строго соответствует структуре процессора.

Например, если процессор содержит функционально независимые устройства (сложения, умножения, сдвига и деления), то максимум, что компилятор может «уложить» в один пакет - это четыре разнотипные операции:
(сложение, умножение, сдвиг и деление). Сформированные пакеты операций преобразуются компилятором в командные слова, которые по сравнению с обычными инструкциями выглядят очень большими. Отсюда и название этих суперкоманд и соответствующей им архитектуры - VLIW (Very Large Instruction Word - очень широкое командное слово). По идее, затраты на формирование суперкоманд должны окупаться скоростью их выполнения и простотой аппаратуры процессора, с которого снята вся «интеллектуальная» работа по поиску параллелизма несвязанных операций. Однако практическое внедрение VLIW-архитектуры затрудняется значительными проблемами эффективной компиляции.

Архитектуры класса SISD охватывают те уровни программного параллелизма, которые связаны с одинарным потоком данных. Они реализуются многофункциональной обработкой и конвейером команд.

Параллелизм циклов и итераций тесно связан с понятием множественности потоков данных и реализуется векторной обработкой. В классификации компьютерных архитектур М. Флина выделена специальная группа однопроцессорных систем с параллельной обработкой потоков данных - SIMD (Single Instruction Multiple Date, один поток команд - множество потоков данных).

Возможны два способа построения компьютеров этого класса. Это матричная структура и векторно-конвейерная обработка. Суть матричной структуры заключается в том, что имеется множество процессорных элементов, исполняющих одну и ту же команду над различными элементами вектора, объединенных коммутатором. Основная проблема заключается в программировании обмена данными между процессорными элементами через коммутатор.

В отличие от матричной, векторно-конвейерная структура компьютера содержит конвейер операций, на котором обрабатываются параллельно элементы векторов и полученные результаты последовательно записываются в единую память. При этом отпадает необходимость в коммутаторе процессорных элементов, служащем камнем преткновения в матричных компьютерах.

Примером векторных супер-ЭВМ с матричной структурой является знаменитая в свое время система ILLIAC-IV.

Векторно-конвейерную структуру имеют однопроцессорные супер-ЭВМ серии VP фирмы Fujitsu; серии S компании Hitachi; C90, M90, T90 фирмы Cray Research; Cray-3, Cray-4 фирмы Cray Computer и т.д. Общим для всех векторных суперкомпьютеров является наличие в системе команд векторных операций, допускающих работу с векторами определенной

длины, допустим, 64 элемента по 8 байт. В таких компьютерах операции с векторами обычно выполняются над векторными регистрами.

Еще одним примером SIMD-архитектуры является технология MMX, которая существенно улучшила архитектуру микропроцессоров фирмы Intel. Она разработана для ускорения выполнения мультимедийных и коммуникационных программ. В MMX используются 4 новых типа данных и 57 новых инструкций. Команды MMX выполняют одну и ту же функцию с различными частями данных, например, 8 байт графических данных передаются в процессор как одно упакованное 64-х разрядное число и обрабатываются одной командой. MMX - команды используют восемь 64-разрядных регистров, «физически» размещенных в мантиссах регистров с плавающей запятой, и используются в том же режиме процессора, что и команды с плавающей запятой.

Все программное обеспечение, созданное для ранее выпущенных процессоров, без всяких изменений может выполняться на процессорах с технологией MMX.

1.2. Технические и эксплуатационные характеристики ЭВМ

Основным техническим параметром ЭВМ является её **производительность.** Этот показатель определяется архитектурой процессора, иерархией внутренней и внешней памяти, пропускной способностью системного интерфейса, системой прерывания, набором периферийных устройств в конкретной конфигурации, совершенством операционной системы и т.д.

Различают следующие виды производительности:

- **пиковая** (предельная) — это производительность процессора без учета времени обращения к оперативной памяти (ОП) за операндами;
- **номинальная** — производительность процессора с ОП;
- **системная** — производительность базовых технических и программных средств, входящих в комплект поставки ЭВМ;
- **эксплуатационная** — производительность на реальной рабочей нагрузке, формируемой в основном используемыми пакетами прикладных программ (ППП) общего назначения.

Методы определения производительности разделяются на три основных группы:

- расчетные, основанные на информации, получаемой теоретическим или эмпирическим путем;
- экспериментальные, основанные на информации, получаемой с использованием аппаратно-программных измерительных средств;
- имитационные, применяемые для сложных ЭВМ. Основные единицы оценки .производительности:
- **абсолютная,** определяемая количеством элементарных работ, выполняемых в единицу времени;
- **относительная,** определяемая для оцениваемой ЭВМ относительно базовой в виде индекса производительности.

Для каждого вида производительности применяются следующие традиционные методы их определения.

Пиковая производительность (быстродействие) определяется средним числом команд типа «регистр-регистр», выполняемых в одну секунду без учета их статистического веса в выбранном классе задач.

Номинальная производительность (быстродействие) определяется средним числом команд, выполняемых подсистемой «процессор-память» с учетом их статистического веса в выбранном классе задач. Она рассчитывается, как правило, по формулам и специальным методикам, предложенным для процессоров определенных архитектур, и измеряется с помощью разработанных для них измерительных программ, реализующих соответствующую эталонную нагрузку.

Для данных типов производительностей используются следующие единицы измерения:

MIPS (Mega Instruction Per Second) - миллион целочисленных операций в секунду;

MFLOPS (Mega Floating Operations Per Second) - миллион операций *с* числами с плавающей запятой в секунду;

GFLOPS (Giga Floating Operations Per Second) - миллиард операций НЕ С числами с плавающей загоггой в секунду;

TFLOPS (Tera Floating Operations Per Second) - триллион операций на числами с плавающей запятой в секунду.

Системная производительность измеряется с помощью синтезированны типовых (тестовых) оценочных программ, реализованных на унифицирова! ных языках высокого уровня. Унифицированные тестовые программы и< пользуют типичные алгоритмические действия, характерные для реальны применений, и штатные компиляторы ЭВМ. Они рассчитаны на использовг ние базовых технических средств и позволяют измерять производитеяьност для расширенных конфигураций технических средств. Результаты оценк системной производительности ЭВМ конкретной архитектуры приводятс относительно базового образца, в качестве которого используются ЭВМ, *m* ляющиеся промышленными стандартами систем ЭВМ различной архитекту ры. Результата оформляются в виде сравнительных таблиц, двумерных гра фиков и трехмерных изображений.

Существует большое количество различных тестовых программ, напри мер:

SPEC mt 95 и SPEC fp 95 - для интенсивной проверки процессоров п< целочисленным операциям и вычислениям с плавающей запятой (табл. 1.3);

Graphstone - набор из 125 подпрограмм, проверяющих компьютер на'1; различных типах графических элементов;

Khomevstone - методика испытаний с использованием 21 теста, интен сивно нагружающих центральный процессор, процессор с плавающей зал» той и дисковую подсистему.

Эксплуатационная производительность оценивается на основании ис пользования данных о реальной рабочей нагрузке и функционировании ЭВ^ при выполнении типовых производственных нагрузок в основных областя? применения. Расчеты делаются главным образом на уровне типовых пакете! прикладных программ текстообработки, систем

управления базами данных пакетов автоматизации проектирования, графических пакетов и т.д.

Очень часто при сравнении компьютеров пользуются отношением производительности к стоимости.

К другим технико-эксплуатационным характеристикам ЭВМ относятся:

- разрядность обрабатываемых слов и кодовых шин интерфейса;
- типы системного и локальных интерфейсов;
- тип и емкость оперативной памяти;
- тип и емкость накопителя на жестком магнитном диске;
- тип и емкость накопителя на гибком магнитном диске;
- тип и емкость кэш-памяти;
- тип видеоадаптера и видеомонитора;
- наличие средств для работы в компьютерной сети;
- наличие и тип программного обеспечения;
- надежность ЭВМ;
- стоимость;
- габариты и масса.
-

Таблица 1.3 Результаты тестирования процессоров

Фирма	Процессор	Результаты тестиро	
		SPECint95	SPECfp95
Digital Equipn	Alpha 21164/500	15,00	20,4
HP	PA-8000/180 Мгц	11,8	18,7
Intel	Pentium Pro/200 N	8,58	6,68
Motorola	Power PC604e/20(	8,00	6,31
Solicon Graph	Mips R10000/200	8,88	13,8
Sun Microsyst	Ultra SPARC2/20(	7,72	13,8

1.3. Классификация ЭВМ'

1.3.1. Классификация ЭВМ по назначению

По назначению ЭВМ можно разделить на три группы: **универсальные** (общего назначения), **проблемно-ориентированные и специализированные** (рис. 1.2).

Рис. 1.2. Классификация Э@М по назначению

Универсальные ЭВМ предназначены для решения самых различных видов задач: научных, инженерно-технических, экономических, информационных, управленческих и других задач. В качестве универсальных ЭВМ используются различные типы компьютеров, начиная

от супер-ЭВМ и кончая персональными ЭВМ. Решаемые на этих компьютерах задачи отличаются сложностью алгоритмов и объемами обрабатываемых данных. Причем одни универсальные ЭВМ могут работать в многопользовательском режиме (в вычислительных центрах коллективного пользования, в локальных компьютерных сетях и т.д.), другие — в однопользовательском режиме.

Проблемно-ориентированные ЭВМ служат для решения более узкого круга задач, связанных, как правило, с управлением технологическими объектами; регистрацией, накоплением и обработкой относительно небольших объемов данных; выполнением расчетов по относительно несложным алгоритмам. На проблемно-ориентированных ЭВМ, в частности, создаются всевозможные управляющие вычислительные комплексы.

Специализированные ЭВМ используются для решения еще более узкого круга задач или реализации строго определенной группы функций. Такая узкая ориентация ЭВМ позволяет четко специализировать их структуру, во многих случаях существенно снизить их сложность и стоимость при сохранении высокой производительности и надежности их работы.

1.3.2. Классификация ЭВМ по функциональным возможностям и размерам

По функциональным возможностям и размерам ЭВМ можно разделить (рис. 1.3) на супер-ЭВМ, большие, малые и микро-ЭВМ.

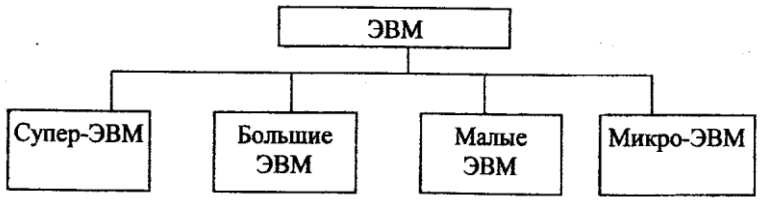


Рис. 1.3. Классификация ЭВМ по функциональным возможностям и размерам

Функциональные возможности ЭВМ обуславливаются основными технико-эксплуатационными характеристиками.

Некоторые сравнительные параметры названных классов современных ЭВМ приведены в табл. 1.4.

Исторически первыми появились большие ЭВМ, элементная база которых прошла путь от электронных ламп до интегральных схем со сверхвысокой степенью интеграции.

Таблица 1.4 Сравнительные параметры различных классов ЭВМ

Параметр	Супер-ЭВ	Большие	Малые Э	МикроЭ

Производит	1000—100	10-1000	1 100	1-100
Емкость ОП	2000—100	64—1000	4-512	4-256
Емкость ВЗ	500—5000	50—1000	2—100	0,5—10
Разрядности	64—128	32—64	16-64	8—64

Большие ЭВМ за рубежом часто называют **мэйнфреймами** (Mainframe). Они поддерживают многопользовательский режим работы (обслуживают одновременно от 16 до 1000 пользователей).

Основные направления эффективного применения мэйнфреймов — это решение научно-технических задач, работа в вычислительных системах с пакетной обработкой информации, работа с большими базами данных, управление вычислительными сетями и их ресурсами. Последнее направление — использование мэйнфреймов в качестве больших серверов вычислительных сетей - часто отмечается специалистами среди наиболее актуальных.

Родоначальником больших ЭВМ является фирма IBM. По её стандартам (IBM 360, 370, 380, 390) в последние несколько десятилетий развивались ЭВМ этого класса в большинстве стран мира. В нашей стране было создано семейство больших машин ЕС ЭВМ.

Среди лучших современных разработок мэйнфреймов за рубежом следует в первую очередь отметить: IBM ES/9000 (созданные в 1990 г.), IBM S/390 (созданные в 1997 г.), а также японские компьютеры Ml 800 фирмы Fujitsu.

Малые ЭВМ (мини-ЭВМ) - надежные, недорогие и удобные в эксплуатации компьютеры, обладающие несколько более низкими по сравнению с мэйнфреймами возможностями. В многопользовательском режиме поддерживаются 16 - 512 пользователей. Основные их особенности: широкий диапазон производительности в конкретных условиях применения, аппаратная реализация большинства системных функций ввода-вывода информации, простая реализация многопроцессорных и многомашинных систем, высокая скорость обработки прерываний, возможность работы с форматами данных различной длины.

К достоинствам мини-ЭВМ можно отнести: специфическую архитектуру с большой модульностью; лучшее, чем у мэйнфреймов, соотношение производительность / стоимость; повышенную точность вычислений.

Мини-ЭВМ ориентированы на использование в качестве управляющих вычислительных комплексов. Традиционная для подобных комплексов широкая номенклатура периферийных устройств дополняется блоками межпроцессорной связи, благодаря чему обеспечивается реализация вычислительных систем с изменяемой структурой.

Кроме этого, мини-ЭВМ успешно применяются для вычислений в многопользовательских вычислительных системах, в системах автоматизированного проектирования, в системах моделирования и искусственного интеллекта.

Родоначальником мини-ЭВМ можно считать компьютеры PDP-11 фир-

мы DEC (Digital Equipment Corporation) США. Они явились прообразом и наших отечественных мини-ЭВМ - Системы Малых ЭВМ (СМ ЭВМ): СМ1, 2,3,4,1400,1700 и др.
В настоящее время семейство мини-ЭВМ включает большое число моделей от VAX-11 до VAX 8000, супермкни - ЭВМ класса VAX 9000 и др. Модели VAX полностью перекрывают весь диапазон характеристик этого класса "компьютеров, а супермини-ЭВМ стирают грань с мэйнфреймами.

Супер-ЭВМ — мощные, высокоскоростные вычислительные машины (системы) с производительностью от сотен миллионов до триллионов операций с плавающей запятой в секунду. Супер-ЭВМ выгодно отличаются от больших универсальных ЭВМ по быстродействию числовой обработки, а от специализированных машин, обладающих высоким быстродействием в сугубо ограниченных областях, возможностью решения широкого класса задач с числовыми расчетами.

При производительности порядка нескольких GFLOPS можно еще обойтись одним векторно-конвейерным процессором (однопроцессорные суперЭВМ). Создание высокопроизводительной супер-ЭВМ с TFLOPS-ным быстродействием по современной технологии на одном процессоре не представляется возможным. Это связано с ограничением, обусловленным конечным значением скорости распространения электромагнитных волн (300 000 км/сек), так как время распространения сигнала на расстояние нескольких миллиметров (линейный размер стороны микропроцессора) при быстродействии 100 млрд. оп/с становится соизмеримым с временем выполнения одной операции. Поэтому супер-ЭВМ с такой производительностью создаются в виде высокопараллельных многопроцессорных вычислительных систем.

В настоящее время в мире насчитывается несколько тысяч супер-ЭВМ, начиная от простых офисных до мощных: Gray Y-MP C90 (фирмы Gray Research), Cyber 205 (фирмы Control Data), VP 2000 (фирмы Fujitsu), VPP500 (фирмы Siemens) и др., производительностью несколько десятков GFLOPS.

Самый мощный компьютер мира был проанонсирован в 1998 г. Это массивно-параллельный компьютер ASCI Blue (другое название - Blue Pacific) создан совместно корпорацией IBM и Ливерморской Национальной Лабораторией Департамента Энергетики США на основе распространенной архитектуры RISC процессора IBM RS/6000 SP2.

Система ASCI Blue построена на базе 4-процеССорных High-узлов с архитектурой SMP. Система состоит из 1464 таких УЗЛОВ или, в общей сложности, из 5856 процессоров, обеспечивая пиковую производительность в 3,88 Тфлопс. Общая сумма контракта на поставку системы Составляет 94 млн. дол.

ASCI Blue побьет современный «барьер скорости вычислений», выполняя 3,9 трлн. операций в секунду, т.е. примерно в 15 тысяч раз больше, чем типичный настольный компьютер. Суммарный объем оперативной памяти составляет 2,6 Тбайт, что примерно в 80 тысяч раз больше, чем у современных ПК.

Микро-ЭВМ по назначению можно разделить на универсальные и специализированные (рис. 1.4).

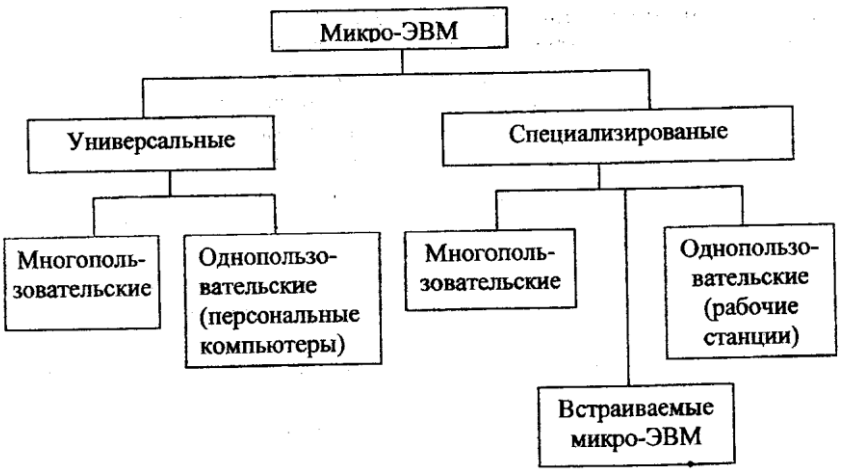


Рис. 1.4. Классификация микро-ЭВМ

Универсальные многопользовательские ЭВМ - это мощные микро ЭВМ, оборудованные несколькими видеотерминалами и функционирующие в режиме разделения времени, что позволяет эффективно работать на них сразу нескольким пользователям. Это **универсальные серверы** (Server) компьютерных сетей, обрабатывающие запросы от всех станций сети.

Персональные компьютеры (ПК) — однопользовательские микро-ЭВМ, удовлетворяющие требованиям общедоступности и универсальности применения.

Специализированные ЭВМ ориентированы на решение определенного (постоянного) класса задач в течение периода своей эксплуатации. Ориентация специализированных ЭВМ осуществляется различными способами: специальной аппаратурной организацией самих ЭВМ или их внешних связей;

созданием для ЭВМ специального программного обеспечения; введением дополнительных аппаратных блоков, .расширяющих те или иные функции, возлагаемые на ЭВМ, и др. Сферы использования таких ЭВМ как в нашей стране, так и за рубежом имеют устойчивую тенденцию к расширению. Можно выделить следующие основные области применения специализированных ЭВМ: промышленное производство и транспорт; военная техника и оборона; непромышленная сфера.

Примером специализированных однопользовательских микро-ЭВМ, ориентированных для выполнения определенного круга задач (графических, инженерных, издательских и др.), являются **рабочие станции** (Work Station).'

Специализированные серверы, осуществляющие управление базами и архивами данных, многопользовательскими терминалами, поддерживающими факсимильную связь, электронную почту и др., относятся к классу **специализированных многопользовательских микро-ЭВМ.**

Встраиваемые микро-ЭВМ входят составным элементом в промышленные и транспортные системы, технические устройства и аппараты, бытовые приборы. Они способствуют существенному повышению их эффективности функционирования, улучшению технико-экономических и

эксплуатационных характеристик.

Персональный компьютер для удовлетворения требованиям общедоступности и универсальности применения должен иметь следующие характеристики:

- малую стоимость, находящуюся в пределах доступности для индивидуального покупателя;
- автономность эксплуатации без специальных требований к условиям окружающей среды;
- гибкость архитектуры, обеспечивающую ее адаптивность к разнообразным применениям в сфере управления, науки, образования, в быту;
- «дружественность» операционной системы и прочего программного обеспечения для пользователя;
- высокую надежность работы (более 5000 ч. наработки на отказ).

Наибольшей популярностью в настоящее время пользуется ПК архитектурного направления (платформы) IBM с микропроцессорами фирмы Intel. Данное направление имеет большое количество клонов, т.е. аналогичных компьютеров, выпускаемых различными фирмами США, Западной Европы, России, Японии и др.

Существенно им уступают по популярности ПК направления DEC с микропроцессорами фирмы Motorola, занимающие 2-е место.

В начале 90-х годов мировой парк компьютеров составлял примерно 150 млн.шт., из них около 90 % — это персональные компьютеры. В их числе более 100 млн.шт. (около 75 % всех ПК) типа IBM PC, типа DEC около 5 млн.шт.

Классификация ПК по конструктивным особенностям показана на рис. 1.5.

Рис. 1.5. Классификация персональных компьютеров по конструктивным особенностям

Переносные компьютеры — быстроразвивающийся подкласс персональных компьютеров. По прогнозам специалистов в 2000 г. около 80 % пользователей будут использовать именно переносные машины.

Переносные компьютеры весьма разнообразны:

- мощные переносные компьютеры (рабочие станции) массой до 15 «от., носят жаргонное название Nomadic - кочевник;
- портативные (наколенные) компьютеры типа «LapTop» массой 5-Юг;
- компьютеры-блокноты (Note Book и Sub Note Book, их называют также и Omni Book - «вездесущие») массой 1,5-4 кг;
- карманные компьютеры (Palm Top - «наладонные») имеют массу до

300 г;

- электронные секретари (PDA - Personal Digital Assistent, иногда их называют Hand Help - ручной помощник) массой не более 0,5 кг, но с более широкими возможностями, чем у Palm Top.

Серверы — многопользовательские ЭВМ, используемые в компьютерных сетях. Эту интенсивно развивающуюся группу компьютеров обычно относят к микро-ЭВМ, но по своим характеристикам мощные серверы скорее можно отнести к малым ЭВМ и даже к мэйнфреймам, а супер серверы приближаются к супер-ЭВМ.

Универсальный сервер — это компьютер, выделенный для обработки запросов от всех станций вычислительной сети, предоставляющий этим станциям доступ к общим системным ресурсам (вычислительным мощностям, базам данных, библиотекам программ, принтерам, факсам и др.) и распределяющий эти ресурсы.

Специализированные серверы используются для устранения наиболее «узких» мест в работе сети: создание и управление базами и архивами данных, поддержка многоадресной факсимильной связи и электронной почты, управление многопользовательскими терминалами (принтером, плоттером и Профайл-сервер используется для работы с файлами данных, имеет объемные дисковые ЗУ, часто на отказоустойчивых дисковых массивах RAID.

Архивационный сервер (сервер резервного копирования) предназначен для резервного копирования информации, использует накопители на магнитной ленте (стриммеры) со сменными картриджами.

Факс-сервер, почтовый сервер - выделенные компьютеры для организации эффективной многоадресной факсимильной связи или электронной почты.

Рабочая станция (Work station), по определению экспертов IDC " (International Data Corporation), - это однопользовательская система с мощным процессором и многозадачной ОС, имеющая развитую графику с высоким разрешением, большую дисковую и оперативную память и встроенные сетевые средства.

Рабочие станции (WS) появились на рынке ЭВМ почти одновременно с ПК и находились впереди по своим вычислительным возможностям. Это в значительной мере и определяло их область применения и проблемную ориентацию за последние 10 лет: автоматизированное проектирование, банковское дело, управление производством, разведка и добыча нефти, связь, издательская деятельность и др. Стоимость таких рабочих станций на много превышала стоимость ПК.

Переломным моментом в развитии WS стало появление новой архитектуры микропроцессоров - RISC, позволившей резко поднять производительность ЭВМ. Прогресс технологии привел к тому, что цены на WS упали до уровня ПК, а порой даже ниже. Выяснилось, что работать за новыми WS проще, чем за персональными компьютерами (десятилетия упорного труда инженеров и программистов не прошли даром). Современная рабочая станция - это не просто большая вычислительная мощность. Это тщательно сбалансированные возможности всех подсистем машины, чтобы ни одна из них не стала «бутылочным горлышком», сводя на нет преимущества других. Рабочая станция стала **местом для**

эффективной, плодотворной работы пользователя.

Бесспорным лидером на мировом рынке рабочих станций является американская фирма Sun Microsystems, она контролирует 40 % этого рынка (H.P. - 20 %, IBM - 7 %, DEC - 11 %). Архитектура SPARC, разработанная фирмой Sun и использующаяся в её машинах, стала фактически стандартом де-факто. Компьютеры этой архитектуры составляют более 75 % среди всех RISC-машин. Уже несколько десятков фирм в Америке, Европе, Японии производят SPARC-совместимые машины.

Традиционно доминирующей ОС на рынке WS была система Unix и ей подобные системы (Solaris и др.). В частности в 1991 г. 92,6 % рабочих станций продавалось именно с этой ОС. В последнее время появились два фактора, которые могут поколебать положение Unix на рынке WS. С появлением новых WS фирмы DEC на базе процессоров Alpha ожидается некоторый рост использования операционной системы VAX VMS.

Однако главным конкурентом является ОС Windows NT. Фирма HP в 1997 г. представила новую серию недорогих высокопроизводительных рабочих станций Kayak, построенных на основе процессора Pentium II корпорации Intel и работающих под управлением ОС Windows NT. Корпорация Intel проявляет повышенный интерес к этому типу платформ, направляет усилия на разработку комплектов микросхем, памяти, графики и других продуктов *VAX VMS.*

Такое развитие событий не сулит ничего хорошего рабочим станциям Unix-стандарта, которые уступают рынок NT-системам пядь за пядью на протяжении вот уже более пяти лет.

2. ФУНКЦИОНАЛЬНАЯ И СТРУКТУРНАЯ ОРГАНИЗАЦИЯ ЭВМ

2.1. Связь между функциональной и структурной организацией ЭВМ

Существуют два взгляда на построение и функционирование ЭВМ. Первый - взгляд пользователя, не интересующегося технической реализацией ЭВМ и озабоченного только получением некоторого набора функций и услуг, обеспечивающих эффективное решение его задач; второй - разработчика ЭВМ, усилия которого направлены на рациональную техническую реализацию необходимых пользователю функций. С учетом этого обстоятельства и вводятся понятия "функциональная и структурная организация компьютера.

Действительно, с точки зрения пользователя решение любой задачи на ЭВМ требует поэтапного выполнения некоторой последовательности действий: кодирования, программирования, ввода, обработки, документирования. На каждом из этих этапов учет запросов пользователя может потребовать расширения реализуемых ЭВМ функцией услуг, что решается при проектировании ЭВМ и входит в понятие **функциональной организации ЭВМ.**

В результате создается абстрактная модель ЭВМ, описывающая функциональные возможности машины и предоставляемые ею услуги. Функциональная организация ЭВМ в значительной степени определяется предъявляемыми к ней требованиями, уровнем подготовки потенциальных

пользователей, типом решаемых **ими** задач, потребностями в развитии компьютера (по емкости ЗУ, разрядности, составу периферийных устройств и др.).

Предусматриваемые абстрактной моделью функции ЭВМ реализуются на основе реальных, физических средств (устройств, блоков, узлов, элементов) в рамках определенной структуры. В общем случае под **структурной организацией ЭВМ** понимается некоторая физическая модель, устанавливающая состав, порядок и принципы взаимодействия основных функциональных частей машины (без излишних деталей их технической реализации).

По степени детальности различают структурные схемы, составленные на уровне устройств, блоков, узлов и элементов.

Блок — функциональный компонент ЭВМ, состоящий из элементов, узлов и выполняющий операции над машинными словами или управляющий такими операциями (сумматор, блок регистров).

Устройство — наиболее крупная функциональная часть ЭВМ, состоящая из элементов, узлов, блоков и выполняющая глобальные операции над кодированными данными (запоминание, обработку, преобразование).

Блоки и устройства часто изготавливаются в виде самостоятельных конструктивов-модулей.

Функциональная организация ЭВМ играет ведущую роль и в значительной степени определяет структурную организацию машины, хотя и не дает жестких ограничений на конечную техническую реализацию структурных элементов. Вместе с тем функции и структура любого элемента находятся в диалектической взаимосвязи и взаимозависимости. С одной стороны, функциональным назначением устройства (блока, узла) ЭВМ определяется необходимый состав материальных объектов (реальных аппаратных и программных средств) и характер их взаимодействия. С другой стороны, одна и та же функция может быть реализована на совершенно разных технических средствах, а изменение состава или связей между элементами, изменение пропорций между аппаратными и программными средствами может сохранить неизменной функцию системы, сообщив ей новые свойства.

2.2. Обобщенная структура ЭВМ и пути её развития

Развитие архитектуры неизбежно ведет к развитию структуры ЭВМ. Реализация принципов интеллектуализации, которые все больше определяют развитие архитектуры ЭВМ, возможна при совершенствовании структурной организации, обеспечивающей повышение эффективности вычислительного процесса и, как следствие этого, рост производительности ЭВМ. В конечном счете, условием и критерием развития структуры является рост производительности ЭВМ.

Основной тенденцией в развитии структуры ЭВМ является разделение функций системы и максимальная специализация подсистем для выполнения этих функций.

Обобщенная структура ЭВМ приведена на рис.2.1. Она состоит из следующих составных частей:

- обрабатывающей подсистемы;
- подсистемы памяти;
- подсистемы ввода-вывода;
- подсистемы управления и обслуживания.

Для каждой подсистемы выделены основные направления их развития.

Обрабатывающая подсистема

Развитие обрабатывающей подсистемы в большей степени, чем всех остальных подсистем, идет по пути разделения функций и повышения специализации составляющих ее устройств. Создаются специальные средства, которые осуществляют функции управления системой, освобождая от этих функций средства обработки. Такое распределение функций сокращает эффективное время обработки информации и повышает производительность ЭВМ. В то же время средства управления, как и средства обработки, становятся более специализированными. Устройство управления памятью реализует эффективные методы передачи данных между средствами обработки и подсистемой памяти. Меняются функции центрального устройства управления. С одной стороны, ряд функций передается в другие подсистемы (например, функции ввода-вывода), с другой - развиваются средства организации параллельной обработки нескольких команд (суперскалярная обработка) с одновременным повышением темпа исполнения последовательности команд. Для повышения темпа выполнения последовательности команд применяются методы конвейерной обработки наряду с совершенствованием алгоритмов диспетчеризации и исполнения команд. Бурно развивается управление межпроцессорным обменом как эффективное средство передачи информации между несколькими центральными процессорами, входящими в состав вычислительной системы или комплекса.

Операционные устройства (АЛУ) обрабатывающей подсистемы, кроме традиционных средств скалярной (суперскалярной) и логической обработки, все шире стали включать специальные средства векторной обработки. При этом время выполнения операций можно резко сократить как за счет использования арифметического конвейера (одного или нескольких), так и за счет сокращения такта работы конвейера. Возможности задач к распараллеливанию алгоритма счета снимают принципиальные ограничения к организации существенно параллельной обработки информации и использованию структур с глубокой конвейеризацией. В устройствах скалярной обработки все шире появляются специальные операционные блоки, оптимизированные на эффективное выполнение отдельных операций.

Рис.2.1.Обобщенная структура ЭВМ и основные направления ее развития

Подсистема памяти

Подсистема памяти современных компьютеров имеет иерархическую структуру, состоящую из нескольких уровней:

- сверхоперативный уровень (локальная память процессора, кэш-память первого и второго уровня);
- оперативный уровень (оперативная память, дисковый кэш);
- внешний уровень (внешние ЗУ на дисках, лентах и т.д.). Эффективными методами повышения производительности ЭВМ являются увеличение количества регистров общего назначения процессора, использование многоуровневой кэш-памяти, увеличение объема и пропускной способности оперативной памяти, буферизация передачи информации между ОП и внешней памятью. Увеличение пропускной способности оперативной памяти достигается за счет увеличения их расслоения и секционирования.

Подсистема ввода-вывода

В состав подсистемы ввода-вывода входит набор специализированных устройств, между которыми распределены функции ввода-вывода, что позволяет свести к минимуму потери производительности системы при операциях ввода-вывода. Эти устройства можно условно разделить на критичные и некритичные по быстродействию. К критичным по быстродействию устройствам относятся обработчики команд ввода-вывода и контроллеры интерфейсов. Эти устройства определяют пропускную способность подсистемы ввода-вывода. Некритичные по быстродействию устройства управляют распределением линий в подсистеме ввода-вывода.

Основными направлениями развития подсистем ввода-вывода являются канальная технология ввода-вывода, матричная топология коммутации периферийных устройств (ПУ), увеличение количества и пропускной способности каналов.

Подсистема управления и обслуживания

Подсистема управления и обслуживания — это совокупность аппаратно-программных средств, предназначенных для обеспечения максимальной производительности, заданной надежности, ремонтопригодности, удобства настройки и эксплуатации. Она обеспечивает проблемную ориентацию и заданное время наработки на отказ, подготовку и накопление статистических сведений о загрузке и прохождении вычислительного процесса, выполняет функции "интеллектуального" интерфейса с различными категориями обслуживающего персонала, осуществляет инициализацию, тестирование и отладку. Подсистема управления и обслуживания позволяет поднять на качественно новый уровень эксплуатацию современных ЭВМ.

При разработке структуры ЭВМ все подсистемы должны быть сбалансированы между собой. Только оптимальное согласование быстродействия обрабатывающей подсистемы с объемами и скоростью передачи информации подсистемы памяти, с пропускной способностью подсистемы ввода-вывода позволяет добиться максимальной эффективности использования ЭВМ.

Важнейшими факторами, определяющими функциональную и структурную организацию ЭВМ, являются выбор системы и форматов команд, типов данных и способов адресации.

2.3. Структура и форматы команд ЭВМ

Все возможные преобразования, дискретной информации могут быть сведены к четырем основным видам:

- передача информации в пространстве (из одного блока ЭВМ в другой);
- передача информации во времени (хранение);
- логические (поразрядные) операции;
- арифметические операции.

ЭВМ, являющаяся универсальным преобразователем дискретной информации, выполняет указанные виды преобразований.

Обработка информации (решение задач) в ЭВМ осуществляется автоматически путем программного управления. Программа представляет собой алгоритм обработки информации (решение задачи), записанный в виде последовательности команд, которые должны быть выполнены машиной для получения результата.

Команда представляет собой код, определяющий операцию и данные, участвующие в операции. Команда содержит также в явной или не явной форме информацию об адресе, по которому помещается результат операции, и об адресе следующей команды.

По характеру выполняемых операций различают следующие основные группы команд:

а) команды арифметических операций над числами с фиксированной

и плавающей точками;

б) команды десятичной арифметики;

в) команды логических операций;

г) команды передачи кодов;

д) команды операций ввода-вывода;

е) команды передачи управления;

ж) команды задания режима работы машины и др.

В команде, как правило, содержатся не сами операнды, а информация об адресах ячеек памяти или регистрах, в которых они находятся.

Команда в общем случае состоит из операционной и адресной частей (рис.2.2,а).

В свою очередь, эти части, что особенно характерно для адресной части, могут состоять из нескольких полей.

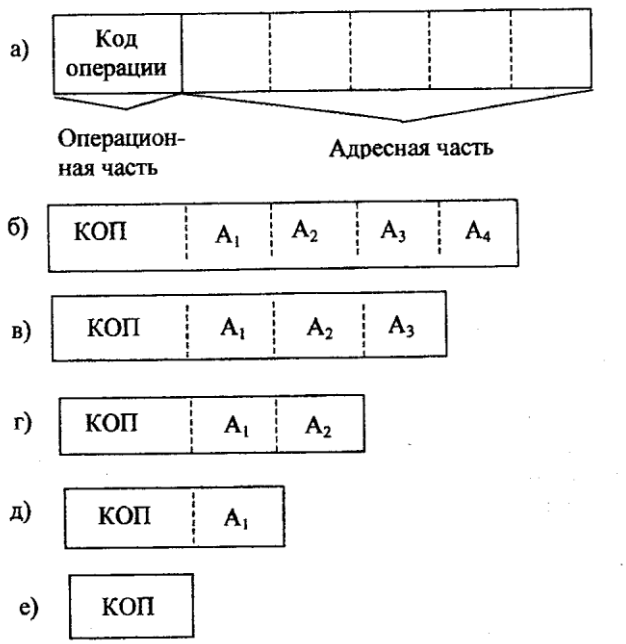


Рис. 2.2. Структуры команд: а) обобщенная; б) четырех-; в) трех-; г) двух -; д) одно -; е) безадресная

Операционная часть содержит код операции (КОП), который задает вид операции (сложение, умножение и др.). Адресная часть содержит информацию об адресах операндов и результате операции, а в некоторых случаях -информацию об адресе следующей команды.

Структура команды определяется составом, назначением и расположением полей в команде.

Форматом команды называют ее структуру с разметкой номеров разрядов (бит), определяющих границы отдельных полей команды, или с указанием числа бит в определенных полях.

Важной и сложной проблемой при проектировании ЭВМ является выбор структуры и форматов команды, т.е. ее длины, назначения и размерности отдельных ее полей. Естественно стремление разместить в команде в возможно более полной форме информацию о

предписываемой командой операции. Однако в условиях, когда в современных ЭВМ значительно возросло число выполняемых различных операций и соответственно команд (в компьютерах с CISC-архитектурой более 200 команд) и значительно увеличилась емкость адресуемой основной памяти (32, 64 Мб), это приводит к недопустимо большой длине формата команды.

Действительно, число двоичных разрядов, отводимых под код операции, должно быть таким, чтобы можно было представить все выполняемые машинные операции. Если ЭВМ выполняет M различных операций, то число разрядов в коде операции

$n_{ком} > \log_2 M$, например, при M=200, $n_{коп} = 8$.

Если основная память содержит S адресуемых ячеек (байт), то для явного представления только одного адреса необходимо в команде иметь адресное поле для одного операнда с числом разрядов

$n_A > \log_2 S$, например, при S = 64 Мб, $n_A = 26$.

Вместе с тем для упрощения аппаратуры и повышения быстродействия ЭВМ длина формата команды должна быть согласована с выбираемой, исходя из требований к точности вычислений, длиной обрабатываемых машиной слов (операндов), составляющей для большинства применений 32 бита с тем, чтобы для операндов и команд можно было эффективно использовать одни и те же память и аппаратные средства обработки информации. Формат команды должен быть по возможности короче, укладываться в машинное слово или полуслово, а для ЭВМ с коротким словом (8-16 бит) должен быть малократным машинному слову. Решение проблемы выбора формата команды значительно усложняется в микропроцессорах, работающих с коротким словом.

Отмечавшиеся ранее характерные для процесса развития ЭВМ расширение системы (наборы) команд и увеличение емкости основной памяти, а особенно создание микроЭВМ с коротким словом, потребовали разработки методов сокращения длины команды. При решении этой проблемы существенно видоизменилась структура команды, получили развитие различные способы адресации информации.

Проследим изменения классических структур команд.

Чтобы команда содержала в явном виде всю необходимую информацию о задаваемой операции, она должна, как это показано на рис. 2.2,6, содержать следующую информацию:

A_1 , A_2; - адреса операндов, A_3 - адрес результата, A_4 _ адрес следующей команды (принудительная адресация команд).

Такая структура приводит к большей длине команды (например, при S = 200, S = 32 Мб длина команды - 108 бит) и неприемлема для прямой адресации операндов основной памяти. В компьютерах с RISC-архитектурой четырехадресные команды используются для адресации операндов, хранящихся в регистровой памяти процессора.

Можно установить, как это принято для большинства машин, что после выполнения данной команды, расположенной по адресу K (и занимающей L ячеек), выполняется команда из (K+L)-й ячейки. Такой порядок выборки команды называется естественным. Он нарушается

только специальными командами (передачи управления). В таком случае отпадает необходимость указывать в команде в явном виде адрес следующей команды.

В трехадресной команде (рис. 2.2,в) первый и второй адреса указывают ячейки памяти, в которых расположены операнды, а третий определяет ячейку, в которую помещается результат операции.

Можно условиться, что результат операции всегда помещается на место одного из операндов, например первого. Получим двухадресную команду (рис. 2.2,г), т.е. для результата используется подразумеваемый адрес.

В одноадресной команде (рис. 2.2д) подразумеваемые адреса имеют уже и результат операции и один из операндов. Один из операндов указывается адресом в команде, в качестве второго используется содержимое регистра процессора, называемого в этом случае регистром результата или аккумулятором (A_{kk}). Результат операции записывается в тот же регистр:

$A_{kk} := A_{kk} * ОП[A_1]$.

Наконец, в некоторых случаях возможно использование безадресных команд (рис. 2.2,е), когда подразумеваются адреса обоих операндов и результата операции, например, при работе со стековой памятью.

С точки зрения программиста, наиболее естественны и удобны трехадресные команды. Однако из-за необходимости иметь большее число разрядов для представления адресов основной памяти и кода операции длина трехадресной команды становится недопустимо большой, и ее не удается разместить в машинном слове. Следует отметить, что очень часто в качестве операндов используются результаты предыдущих операций, хранимые в регистрах машины. По указанным причинам в современных ЭВМ применяют трехадресные команды для адресации регистров.

Способ расширения кодов операции

В машинах с коротким словом практически невозможно в одном формате команды, т.е. при фиксированном назначении ее полей, кодировать большое число различных операций и одновременно иметь гибкую форму адресации операндов. Это противоречие в машинах с коротким словом преодолевается расширением кодов операций в команде. Для задания небольшой группы основных операций (арифметических и др.) используется короткий код операции, а получаемая при этом сравнительно большая адресная часть команды позволяет реализовать гибкую, например двухадресную с многими модификациями, адресацию. Для задания других операций используются более длинные (расширяемые) коды операций, при этом сокращаемая адресная часть оставляет возможность лишь для более простой, например одноадресной адресации операндов. В пределе расширяемый код операции занимает весь формат команды (безадресная команда).

Обычно в ЭВМ используется несколько структур и форматов команд разной длины.

Приведенные на рис. 2.2. структуры команд достаточно схематичны.

В действительности адресные поля команд большей частью содержат не сами адреса, а только информацию, позволяющую определить действительные (исполнительные) адреса операндов в соответствии с используемыми в командах способами адресации.

2.4. Способы адресации информации в ЭВМ

Существует два различных принципа поиска операндов в памяти: **ассоциативный и адресный.**

Ассоциативный поиск операнда (поиск по содержанию ячейки) предполагает просмотр содержимого всех ячеек памяти для выявления кодов, содержащих заданный командой ассоциативный признак. Эти коды и выбираются из памяти в качестве искомых операндов.

Адресный поиск предполагает, что искомый операнд извлекается из ячейки, номер которой формируется на основе информации в адресном поле команды.

Ниже мы будем рассматривать только реализацию адресного принципа поиска операнда.

Следует различать понятия "адресный код" в команде Ак и "исполнительный (физический) адрес" $А_И$.

Адресный код — это информация об адресе операнда, содержащаяся в команде.

Исполнительный адрес - это номер ячейки памяти, к которой производится фактическое обращение. В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом.

Таким образом, способ адресации можно определить как способ формирования исполнительного адреса операнда $А_и$ по адресному коду команды Ак.

Способов адресации существует много. Параметры процесса обработки информации существенно зависят от выбранного способа адресации. Одни способы адресации позволяют увеличить объём адресуемой памяти без удлинения команды, но снижают скорость выполнения операции, другие ускоряют операции над массивами данных, третьи - упрощают работу с подпрограммами и т. д.

В системах команд современных ЭВМ часто предусматривается возможность использования нескольких способов адресации операндов для одной и той же операции. Для указания способа адресации в некоторых системах команд выделяется специальное поле в команде - «метод» (указатель адресации УА), (рис. 2.3,а). В этом случае любая операция может выполняться с любым способом адресации, что значительно упрощает программирование.

Рис.2.3. Общая структура команды: а) с указателем метода адресации; б) без указателя метода адресации

Если только небольшая часть операций должна работать с

различными способами адресации, то в команде поле УА не выделяется, а способ адресации определяется по коду операции, длина которого при этом возрастает (рис. 2.3,6).

Способ адресации операнда определяется многими характеристиками. Многообразие способов адресации обусловлено сочетанием различных значений этих характеристик. Укажем некоторые из этих характеристик и рассмотрим наиболее употребляемые способы адресации.

Вспомним некоторые важные для этой темы понятия и введем необходимые обозначения.

Адресуемые в командах операнды хранятся в основной памяти (ОП) и регистровой памяти (РП), (рис. 2.4).

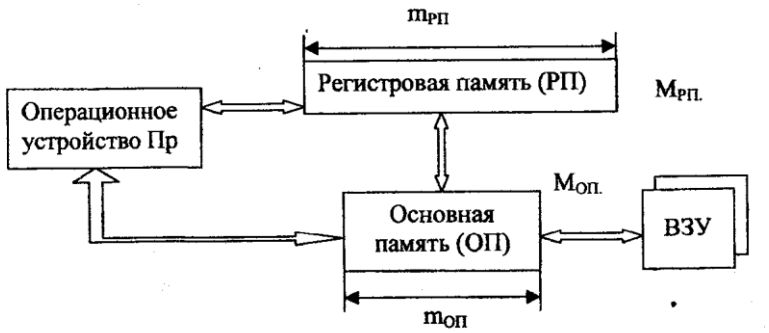


Рис.2.4. Памяти для хранения адресуемых операндов

Каждая память (РП и ОП) имеет самостоятельную нумерацию ячеек (регистров), самостоятельные средства адресования. Пусть:

m - длина многоразрядного двоичного кода, хранимого одной ячейкой
(регистром);

n_A - длина двоичного кода адреса ячейки ($A_и$);

M - емкость памяти, количество ячеек в адресуемом пространстве памяти. Обычно $M = 2^{Na}$ ячеек.

Регистровую память и ОП можно описать следующими параметрами:

Мрп - кол-во регистров в РП;

$m_{рп}$ - разрядность регистра;

АиРП - исполнительный адрес в РП;

АиОП — исполнительный адрес в ОП.

Классификация способов адресации по наличию адресной информации в команде

Явная и неявная адресация

При явной адресации операнда в команде есть поле адреса этого операнда, в котором задается адресный код Ак. Большинство методов адресации являются явными.

При неявной адресации адресное поле в команде отсутствует, адрес операнда подразумевается кодом операции.

Метод неявной адресации операндов используется во всех

процессорах. Основное его назначение — уменьшение длины команды за счет исключения части адресов. При этом методе код операции точно задает адрес операнда. Например, из команды исключается адрес приемника результата. При этом подразумевается, что результат в этой команде помещается на место второго операнда.

Классификация способов адресации по кратности обращения в память

Широко используются следующие методы адресации операнда с различной кратностью обращения (г) в память:

1. Непосредственная (г = *0).*
2. **Прямая** (г = 1).
3. Косвенная (г > 2).

Непосредственная адресация операнда

При этом способе операнд располагается в адресном поле команды. Обращение к РП или ОП не производится. Таким образом, уменьшается время выполнения операции, сокращается объем памяти. Непосредственная адресация удобна для задания констант, длина которых меньше или равна длине адресного поля команды.

Прямая адресация операндов

При этом способе (рис. 2.5) адресации обращение за операндом в РП или ОП производится по адресному коду в поле команды, т.е. исполнительный адрес операнда совпадает с адресным кодом команды (Аи = Ак).

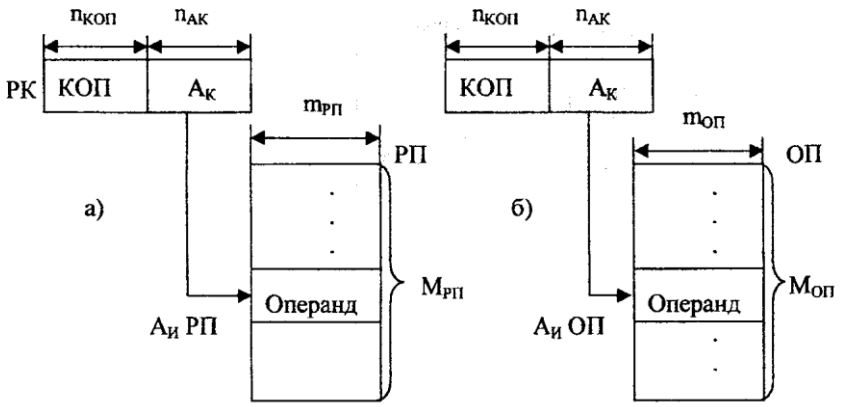


Рис.2.5. Схема прямой адресации: а) к регистровой памяти; б) к основной памяти

Обеспечивая простоту программирования, этот метод имеет существенные недостатки, так как для адресации к ячейкам памяти большой емкости (число адресов М велико) требуется «длинное» адресное поле в команде. Прямая адресация используется широко в сочетании с другими способами адресации. В частности, вся адресация к «малой» регистровой памяти ведется только с помощью прямой адресации.

Косвенная адресация операндов

При этом способе адресный код команды указывает адрес ячейки памяти, в которой находится не сам операнд, а лишь адрес операнда, называемый указателем операнда. Адресация к операнду через цепочку указателей (косвенных адресов) называется косвенной.

Адрес указателя, задаваемый программой, остается неизменным, а косвенный адрес может изменяться в процессе выполнения программы. Косвенная адресация, таким образом, обеспечивает переадресацию данных, т.е. упрощает обработку массивов и списковых структур данных, упрощает передачу параметров подпрограммам, но не обеспечивает перемещаемость программ в памяти (рис. 2.6,а).

Косвенная адресация так же широко используется в ЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата. В этом случае первый указатель должен располагаться в РП (рис. 2.6,6).

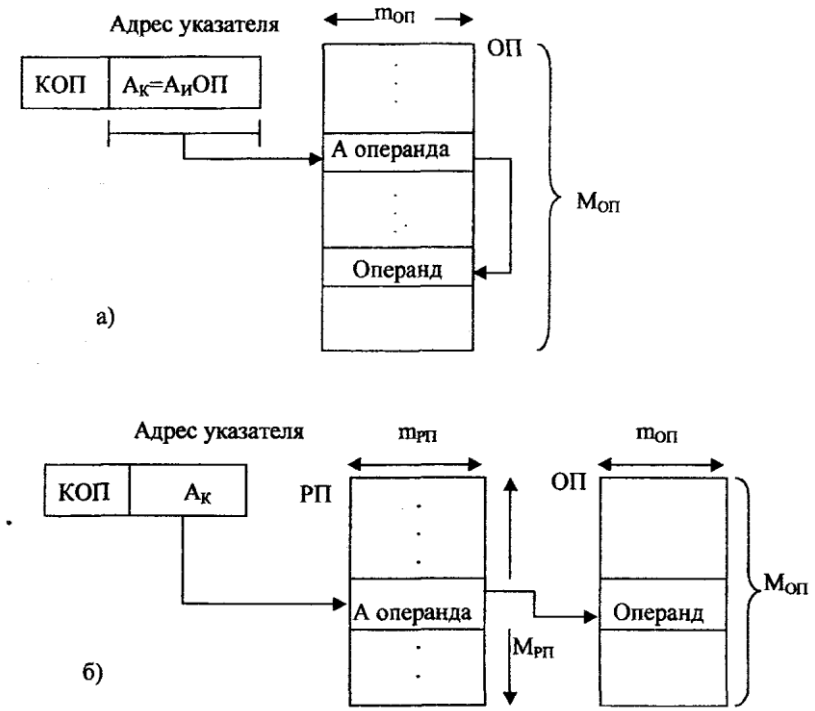


Рис.2.6. Схема косвенной адресации: а) указатель операнда и операнд расположены в одном адресном пространстве ОП; 6) указатель операнда расположен в РП, а операнд-в ОП

Классификация способов формирования исполнительных адресов ячеек памяти

Способы формирования адресов ячеек памяти (Аи) можно разделить на **абсолютные и относительные.**

Абсолютные способы формирования предполагают, что двоичный код адреса ячейки памяти — Аи может быть извлечен целиком либо из адресного поля команды (в случае прямой адресации), или из какой-либо другой ячейки (в случае косвенной адресации), никаких преобразований кода адреса не производится.

Относительные способы формирования Ац предполагают, что двоичный код адреса ячейки памяти образуется из нескольких составляющих: Б — код базы, И - код индекса, С - код смещения, используемых в сочетаниях (Б и С), (И и С). (Б,ИиС).

При относительной адресации применяются два способа вычисления Зд-реса Аи:

- суммирование кодов составляющих адреса;
- совмещение (конкатенация) кодов составляющих адреса.

Суммирование кодов составляющих производится для случаев:
Аи= Б + С; Аи = И + С; Аи = Б + И + С.

Относительная адресация ячейки ОП Базирование способом суммирования

В команде адресный код Ак разделяется на две составляющие: Аб - адрес регистра в регистровой памяти , в котором хранится база Б

(базовый адрес); С - код смещения относительно базового адреса (рис. 2.7).

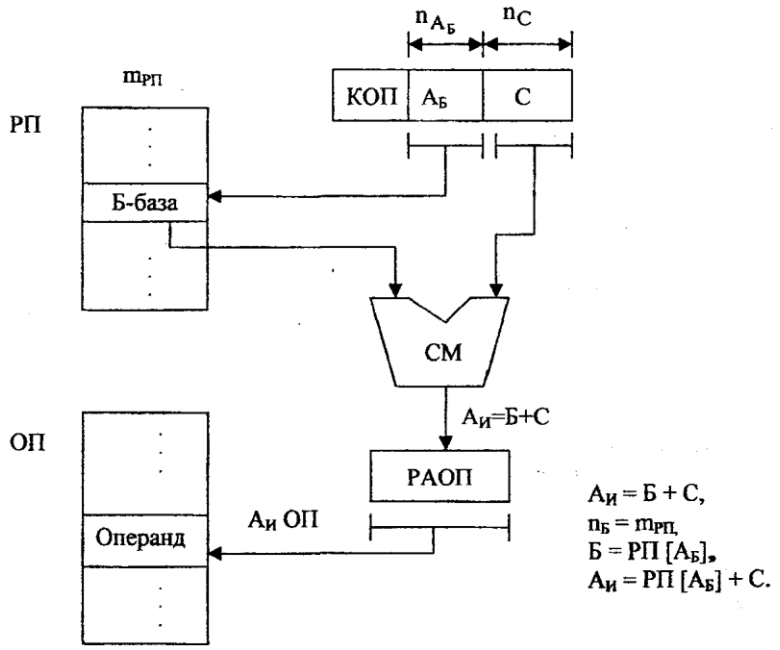


Рис. 2.7. Схема формирования относительного адреса способом суммирования кодов базы и смещения: СМ - сумматор; РАОП - регистр адреса ОП;

Б - база (базовый адрес); С - смещение; *Аб*- адрес регистра базы;

$n_б$ - длина кода базы; $n_с$ - длина поля смешения

Для определения максимальной емкости ОП, адресуемой с помощью базирования, способом суммирования, определим длину кода исполнительного адреса

$$n_{A_И} = n_{A_И ОП} = \max\{ n_Б; n_C \}.$$

Так как $n_б = m_{рп}$ и обычно больше, чем Пс, то справедливо следующее выражение:

$$M_{ОП} = 2^{n_Б} = 2^{m_{РП}},$$

т. е. максимальная адресуемая емкость ОП определяется разрядностью РП. Длина $Na_Б$ поля кода команды, задающего адрес регистра базы Аб, определяется через емкость РП $M_{рп}$, по формуле

$$n_{A_Б} \geq \log_2 M_{РП}$$

Таким образом, можно определить количество $Na_Б$ двоичных разрядов в адресном поле команды, необходимое для формирования Аи с размещением базы в РП:

$$n_{A_К} = n_{A_Б} + n_C = \log_2 M_{РП} + n_C.$$

Приведенные выражения позволяют определить числовые значения параметров относительной адресации (базирование способов суммирования). Для того, чтобы увеличить Моп, необходимо выполнить условие:

$$n_{A_И} \geq n_{A_К}, \text{ т.е } m_{РП} \geq \log_2 M_{РП} + n_C.$$

С помощью метода относительной адресации удается получить так

называемый перемещаемый программный модуль, который одинаково выполняется процессором независимо от адресов, в которых он расположен. Начальный адрес программного модуля (база) загружается, при входе в модуль, в базовый регистр. Все остальные адреса программного модуля формируются через смещение относительно начального адреса (базы) модуля. Таким образом, одна и та же программа может работать с данными, расположенными в любой области памяти, без перемещения данных и без изменения текста программы только за счет изменения содержания всего одного базового регистра. Однако время выполнения каждой операции при этом возрастает.

Относительная адресация с совмещением составляющих Ад

Для увеличения емкости адресной ОП (Моп) без увеличения длины адресного поля команды Пд можно использовать для формирования исполнительного адреса совмещение (конкатенацию) кодов базы и смещения (рис. 2.8).

При совмещении кодов базы и смещения

$$n_{A_И} = n_Б + n_C.$$

Таким образом,

$$M_{ОП} = 2^{n_{A_И}} = 2^{n_Б + n_C}.$$

Следует отметить, что емкость ОП (Моп) может быть увеличена в 2^{nc} раз за счет использования способа совмещения. Однако в данном случае начальные адреса массивов не могут быть реализованы произвольно, а должны иметь в младших разрядах n_c нулей.

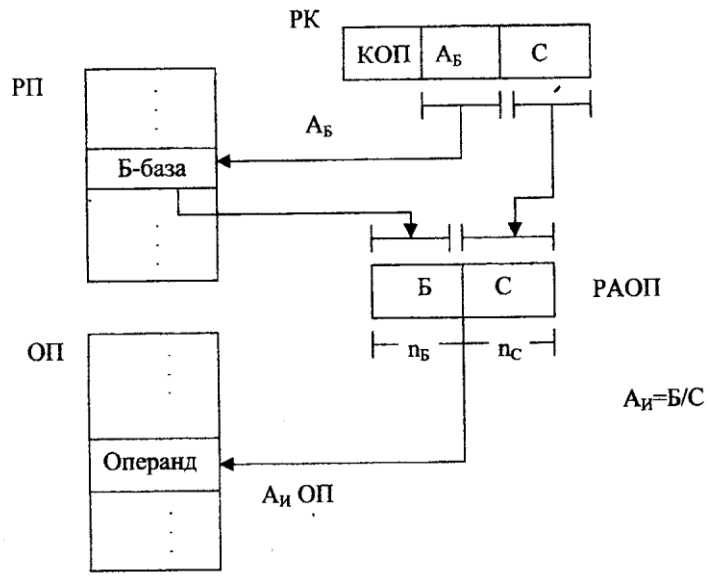


Рис. 2.8. Схема формирования относительного адреса способом совмещения кодов базы и смещения

Индексная адресация

Для работы программ с массивами, требующими однотипных операций над элементами массива, удобно использовать индексную

адресацию. Схема индексной адресации аналогична базированию путем суммирования (см. рис. 2.7). В этом случае адрес i-го операнда в массиве определяется как сумма начального адреса массива (задаваемого полем смещения С) и индекса И, записанного в одном из регистров РП, называемом теперь индексным регистром. Адрес индексного регистра задается в команде полем адреса индекса — Аин (аналогично Аб).

В каждом i-м цикле содержимое индексного регистра изменяется на величину постоянную (часто равную 1). Использование индексной адресации значительно упрощает программирование циклических алгоритмов.

Для эффективной работы при относительной адресации применяется комбинированная индексация с базированием, при которой адрес операнда вычисляется как сумма трех величин (рис. 2.9):

АцОП=Б+И+С.

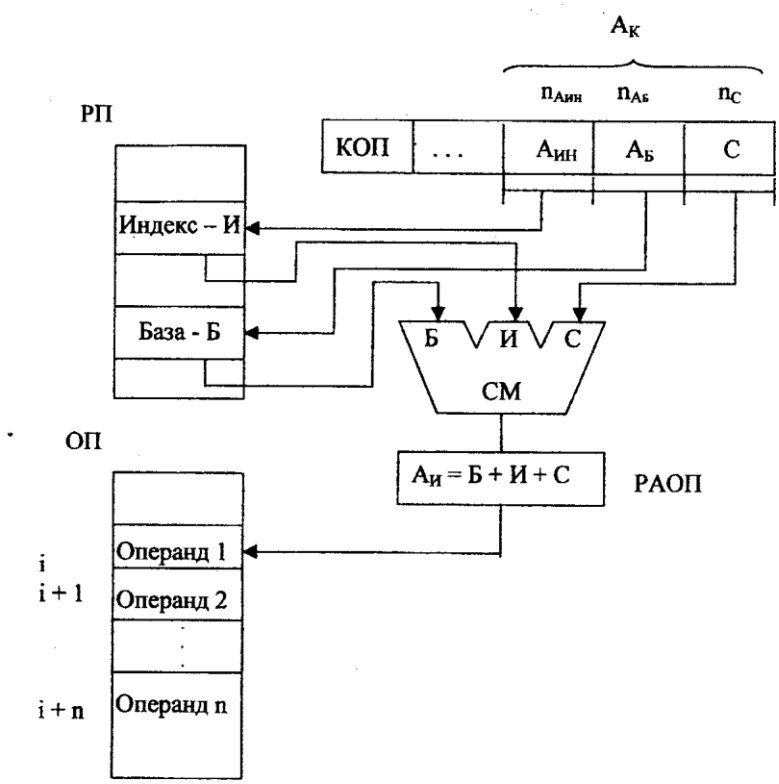


Рис. 2.9. Схема формирования исполнительного адреса при индексной адресации и базировании: Анн - адрес индексного регистра; пдии - длина адреса индексного регистра

Стековая адресация

Стековая память (стек) является эффективным элементом современных ЭВМ, реализует неявное задание адреса операнда. Хотя адрес обращения в стек отсутствует в команде, он формируется схемой управления автоматически по специальному правилу.

2.5. Примеры форматов команд и способов адресации

2.5.1. Форматы команд и способы адресации в CISC-процессорах

В качестве примера рассмотрим набор команд и способы адресации, используемые в процессорах интеловской архитектуры. Для этих процессоров в табл. 2.1 приведены данные о развитии их системы команд.

Таблица 2.1

Развитие системы команд процессоров архитектуры Intel

Год п команд	Тип процесс реализован в	Общее	Смысл расширения
1979	i8086	170	Исходный набор кома
1985	i386	220	50 новых команд, перехода к 32-разрядн
1997	Pentium/MM2	277	57 MMX-команд обработки видео- и ау
1999	Katmai (Penti	347	70 команд: SIMD-FP управление потоковы

В базовый набор команд 8086 входили операции с плавающей запятой (FP), но до i386 включительно они выполнялись отдельным сопроцессором, которого могло в компьютере и не быть. Блок FP-функций был включен в состав основного процессора в i486, однако в варианте 486SX обращение к этому блоку было заблокировано. Эти команды стали обязательной частью процессора, только начиная с Pentium.

Базовый набор команд 32-разрядного интеловского процессора обеспечивает выполнение операций над операндами, которые находятся в регистре, памяти или непосредственно в команде. В набор входят безадресные, одно-, двух- и трехадресные команды. Процессор реализует следующие шесть типов двухадресных команд:

- регистр—регистр;
- память — регистр;
- непосредственный операнд — регистр;
- регистр — память;
- память — память;
- непосредственный операнд — память.

Операнды могут содержать 8, 16 или 32 разряда. Для реализации различных типов команд определены форматы, задающие порядок размещения информации о выполняемой операции и способах выбора операндов.

Общий формат команд

Обобщенный вид формата команды показан на рис. 2.10. Он допускает наличие следующих полей: кода операции (1 или 2 байта); байтов адресации (О, 1 или 2 байта); байтов смещения (0, 1,2 или 4 байта); байтов непосредственных данных — операндов (0,1,2 или 4 байта).

КОП	Байты адресации		Смещение	Операнд
	MOD R/M	SIB		
1 или 2 ба	0 или 1 ба	0 или 1 ба	0,1,2 или 4	0,1,2 или 4

Рис. 2.10. Общий формат команд

Команды содержат от 1 до 11 байт. Проведенные оценки показывают, что в среднем длина команды составляет 4 — 5 байт.

. Рассмотрим назначение основных полей кода команды (рис.2.10). Код операции (КОП) определяет тип выполняемой операции, а также в некоторых командах в первом байте может содержаться бит W, задающий разрядность операндов:

W=0 — операция с байтами;

W = 1 — операция со словами (16 или 32 разряда).

В ряде команд первый байт КОП содержит поля reg или sreg, определяющие адрес используемых регистров. Трехбитовое поле reg задает выбираемый регистр в соответствии с разрядностью обрабатываемых операндов. Поле sreg (двух или трехбитовое) определяет адрес сегментных регистров.

Байт адресации MOD R/M содержит три поля (рис. 2.11). Поля: MOD и R/M задают адрес одного из операндов, который может храниться в регистре или ячейке памяти. Кодировка этих полей определяет выбираемый способ адресации.

Рис. 2.11. Форматы байтов MOD R/M и SIB

В одноадресных командах поле REG/КОП содержит дополнительные биты кода операции. В двухадресных командах поле REG содержит адрес регистра, в котором хранится второй из операндов. Тип команды (одно- или двухадресная) определяется первым битом КОП. Поле MOD указывает, какой разрядности смещение используется для формирования адреса. Если оно имеет значение 00 (при некоторых значениях R/M) или 01, 10, то используется 8-, 16- или 32-разрядное смещение. Это смещение задается соответствующими байтами в коде команды, которые располагаются после байтов адресации.

Для реализации некоторых способов относительной адресации используется байт SIB. Он содержит 3-битовые поля INDEX и BASE, определяющие выбор регистров, используемых в качестве индексного и базового регистров, и поле SS, задающее масштабный коэффициент для модификации значения индекса.

При выполнении операций с непосредственной адресацией один из операндов задается в последних байтах команды (рис. 2.10). В этом случае КОП ряда команд содержит бит S, определяющий способ использования непосредственно задаваемых данных.

Способы адресации

Интеловский 32-разрядный процессор реализует сегментную организацию оперативной памяти, при которой физический адрес ячейки

памяти формируется путем сложения базового адреса сегмента и относительного адреса ячейки внутри сегмента.

Базовый адрес определяется содержимым 16-разрядного сегментного регистра и зависит от режима работы процессора. Если он работает в режиме обработки 16-разрядных данных (режим реальных адресов), то 20-разрядный базовый адрес формируется путем сдвига содержимого сегментного регистра на 4 разряда влево. Если процессор работает в режиме обработки 32-разрядных данных (защищенный режим), то 32-разрядный базовый адрес содержится в дескрипторе, выбор которого из таблицы дескрипторов осуществляется с помощью селектора — содержимого соответствующего сегментного регистра.

В качестве относительного адреса используется содержимое регистров общего назначения или эффективный адрес (EA), который формируется в соответствии с заданным способом адресации. EA является 16- или 32- разрядным и формируется в зависимости от значения полей MOD и R/M и содержимого байта SIB (для 32-разрядных адресов). В общем случае EA образуется путем арифметического сложения трех компонентов:

- содержимого базового регистра;
- содержимого индексного регистра;
- 8, 16, 32-разрядного смещения, заданного в одном, двух или четырех байтах команды.

В зависимости от значений полей MOD и R/M для формирования EA используются все или часть этих слагаемых.

В процессоре осуществляются следующие способы адресации операндов:

- непосредственная адресация;
- регистровая адресация;
- косвенно-регистровая адресация;
- прямая адресация;
- базовая адресация;
- индексная адресация;
- базово-индексная адресация;
- базово-индексная адресация со смещением.

2.5.2. Форматы команд и способы адресации в RISC-процессорах

Рассмотрим архитектурные особенности на примере процессора R3000. Этот процессор строится на основе СБИС 32-разрядного центрального процессора, арифметического сопроцессора и буфера записи.

Система команд включает 74 команды, которые можно разделить на 6 групп: загрузки/запоминания, операционные, переходов, работы с сопроцессором, управления системой и специальные.

Все команды имеют длину 32 бита и могут быть трех форматов:

I - команды с непосредственным операндом и обращения к памяти

КОП(6);Rs(5);Rt(5);I(16),

где Kg, R, - номера регистров, I - непосредственный операнд или

смещение;

J - для команд переходов;

КОП (6); АДРЕС ПЕРЕХОДА (26);

R - для операционных команд;

КОП (6); Rs(5); Rt(5); Rd(5); R_l,(5); FUNC (6).

Операционные команды служат для выполнения арифметических, логических операций и сдвигов. Операционные команды используют как R-формат (команды типа регистр-регистр), так и 1-формат (команды регистр - непосредственный операнд). Предусмотрены 8 команд целочисленного умножения и деления.

Команды загрузки/запоминания обеспечивают обмен данными между регистрами общего назначения и памятью. Адреса памяти формируются с использованием базового регистра и 16-разрядного смещения (1-формат).

Безусловные переходы выполняются либо по косвенному адресу (R-формат), либо по прямому адресу (J-формат). В последнем случае старшие биты адреса переходов добавляются из счетчика команд.

Команды управления системой обеспечивают работу с виртуальной памятью.

Команды работы с сопроцессором являются дополнительными, их форматы и состав зависят or типа используемого сопроцессора.

2.6. Типы данных

Основными типами данных в компьютерах являются байты, слова, двойные слова и квадрослова (рис. 2.12).

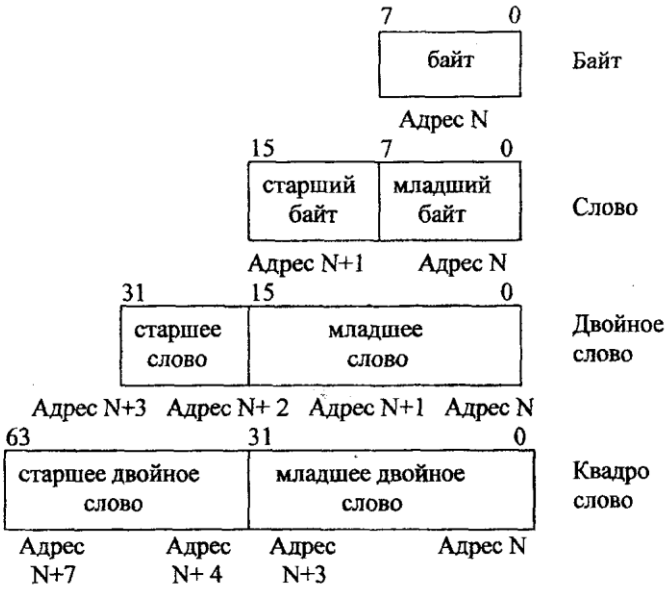


Рис. 2.12. Основные типы данных

Каждый из представленных на рис. 2.12 типов данных может начинаться с любого адреса: это означает, что слово не обязано начинаться с четного адреса; двойное слово - с адреса, кратного 4 и т.д. Таким образом достигается максимальная гибкость структур данных и эффективность использования памяти.

Однако обмен данными между процессором и памятью осуществляется в Pentium через 64-битовую ШД (i486 - 32 р.) и для достижения максимальной производительности этого обмена желательно выравнивать слова по чётным адресам, двойные слова - по адресам, кратным 4 и т.д.

На базе основных типов данных строятся все остальные тишя, распознаваемые командами процессора.

Данные со знаком

На рис. 2.13 приведены 4 формата данных со знаком с фиксированной точкой.

Представление таких данных и выполнение операций производится в дополнительном коде.

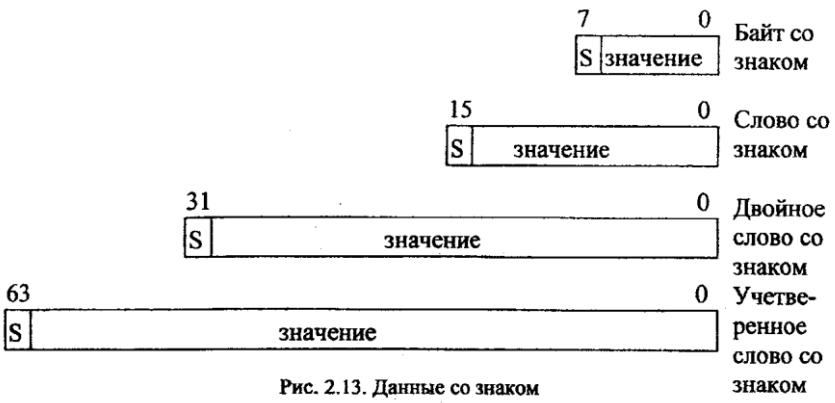


Рис. 2.13. Данные со знаком

Данные без знака

На рис. 2.14 показаны три формата данных без знака-

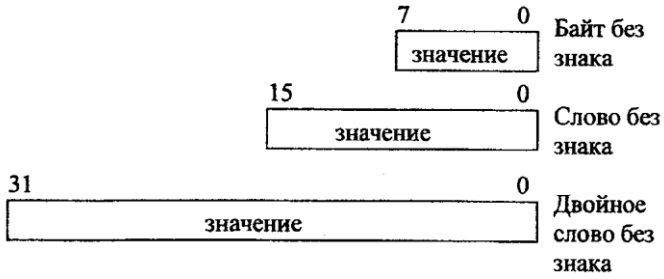


Рис. 2.14. Данные без знака

Данные в формате с плавающей точкой

Формат включает три поля: знака, мантиссы и порядка (рис. 2.15). Поле мантиссы содержит значащие биты числа, а поле порядка содержит степень 2 и определяет масштабирующий множитель для мантиссы. Поддерживаются блоком FPU.

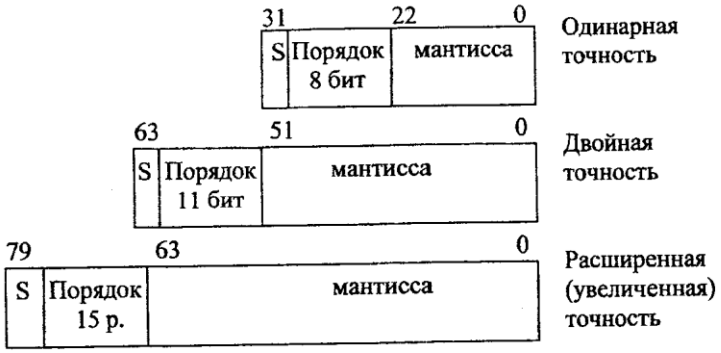


Рис.2.15. Форматы данных с плавающей точкой

Двоично-десятичные данные (BCD)

На рис. 2.16 приведены форматы двоично-десятичных данных.

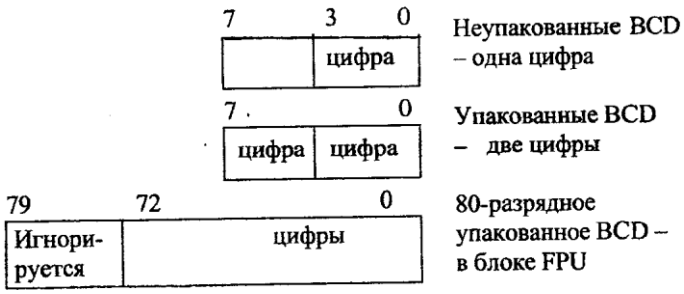


Рис.2.16. Форматы двоично-десятичных данных

Данные типа строка

Строка представляет собой непрерывную последовательность бит, байт, слов или двойных слов (рис. 2.17). Строка бит может быть длиной до 1

Гби-та, а длина остальных строк может составлять от 1 байта до 4 Гбайтов. Поддерживается АЛУ.

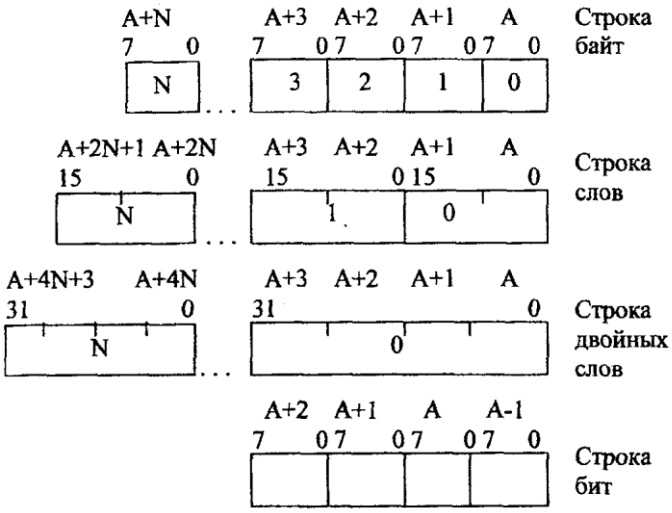


Рис.2.17. Данные типа строка

Символьные данные

Поддерживаются строки символов в коде ASCII и арифметические операции (сложение, умножение) над ними (рис. 2.18). Поддержка осуществляется блоком АЛУ.

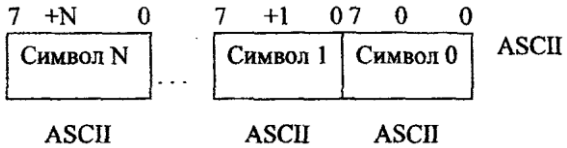


Рис.2.18. Символьные данные

Данные типа указатель

Указатель содержит величину, которая определяет адрес фрагмента данных. Поддерживается два типа указателей, приведенных на рис. 2.19.

Диапазон представления целых чисел лежит в интервале от -2^{64} до 2^{64}. Диапазон нормализованных чисел с двойной точностью - от $\pm 2{,}23 \times 10^{-308}$ до $\pm 1{,}79 \times 10^{-308}$, а с расширенной точностью - от $\pm 3{,}37 \times 10^{-4932}$ *до* $\pm\ 1{,}18 \times 10^{4932}$.

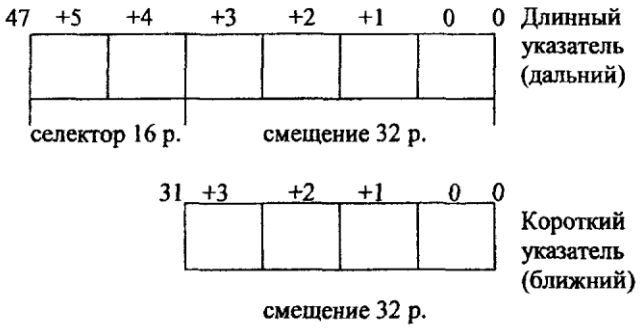


Рис.2.19. Данные типа указатель

2.7. Теги и дескрипторы. Самоопределяемые данные

Одним из эффективных средств совершенствования архитектуры современных ЭВМ является теговая организация памяти, при которой каждое хранящееся в памяти (или регистре) слово снабжается указателем - **тегом** (рис. 2.20,а). Последний определяет тип данных - целое двоичное число, число с плавающей точкой, десятичное число, адрес, строка символов, дескриптор и т.д.

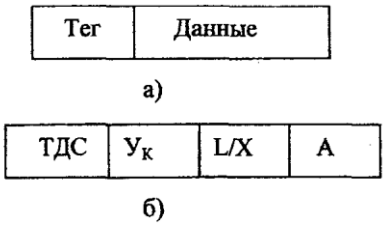


Рис.2.20. Структура описания данных: а) с теговой организацией памяти; б) дескриптор данных *

В поле тега обычно указывается не только тип, но и длина (формат) и некоторые другие его параметры. Теги формируются компилятором.

Наличие тегов придает хранящимся в машине данным свойство самоопределяемости, вносящее принципиальные особенности в архитектуру и функционирование ЭВМ.

Отметим, что ЭВМ с теговой памятью (самоопределяемыми данными) выходит за рамки модели вычислительной машины фон Неймана, где тип (характер) данного определяется командой, использующей данное в качестве операнда. В обычных ЭВМ, соответствующих классической модели фон Неймана, тип данных — операндов и **их** формат задаются кодом операции команды, а в раде случаев размер (формат) определяется следующими полями команды.

Теговая организация памяти позволяет достигнуть инвариантности команд относительно типов и форматов операндов, что приводит к значительному сокращению набора команды машины. Это упрощает и делает более регулярной структуру процессора, облегчает работу программиста, в том числе при отладке программ, упрощает компиляторы и сокращает затраты времени на компиляцию (отпадает необходимость выбора типа команды в зависимости от типа данных), облегчает обнаружение ошибок, связанных с некорректным заданием типов данных (например, при попытке сложить адрес с числом с плавающей точкой).

Теговая организация памяти способствует реализации принципа независимости программ от данных.

И, наконец, нечто неожиданное. Использование тегов приводит к экономии памяти, так как в программах обычных машин имеется большая информационная избыточность на задание типов и размеров операндов при их использовании несколькими командами.

В качестве недостатка теговой организации памяти можно указать на некоторое замедление работы процессора из-за того, что установление соответствия типа команды типу данных, в обычных ЭВМ выполняемое на этапе компиляции, при использовании тегов переносится на этап выполнения программы.

В архитектуре некоторых ЭВМ используются **дескрипторы** — служебные слова, содержащие описание массивов данных и команд, причем дескрипторы могут употребляться как в машинах с теговой организацией памяти, так и без тегов.

Дескриптор содержит сведения о размере массива данных, его местоположении (в ОП или внешней памяти), адресе начала массива, типе данных, режиме защиты данных (например, запрет записи в ячейки массива) и некоторых других параметрах данных. Отметим, что задание в дескрипторе размера массива позволяет контролировать выход за границу массива при индексации его элементов. На рис. 2.20,6 в качестве примера представлен один из видов дескрипторов - дескриптор данных.

Дескриптор содержит специфический тег — ТДС, указывающий, что данное слово является дескриптором определенного вида; Ук — группа указателей; А — адрес начала массива данных; L — длина массива; Х — индекс.

Использование в архитектуре ЭВМ дескрипторов подразумевает, что обращение к информации в памяти производится через дескрипторы, которые при этом можно рассматривать как дальнейшее развитие аппарата косвенной адресации.

Адресация информации в памяти может осуществляться с помощью цепочки дескрипторов, при этом реализуется многоступенчатая косвенная адресация. Более того, сложные многомерные массивы данных (таблицы и т. п.) эффективно описываются древовидными структурами дескрипторов.

3. ФУНКЦИОНАЛЬНАЯ И СТРУКТУРНАЯ ОРГАНИЗАЦИЯ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА ЭВМ

В области вычислительной техники различают процессоры центральные, специализированные, ввода-вывода, передачи данных и коммуникационные.

3.1. Назначение и структура центрального процессора

Центральный процессор — основное устройство ЭВМ, которое наряду с обработкой данных выполняет функции управления системой: инициирование ввода-вывода, обработку системных событий, управление доступом к сновной памяти и т.п.

Организация центрального процессора (ЦП) определяется архитектурой и принципами работы ЭВМ (состав и форматы команд, представление чисел, способы адресации, общая организация машины и её основные элементы), а также технико-экономическими показателями.

Логическую структуру ЦП представляет ряд функциональных средств (рис. 3.1): средства обработки, средства управления системой и программой, локальная память, средства управления вводом-выводом и памятью, системные средства.

Средства обработки обеспечивают выполнение операций с фиксированной и плавающей точкой, операций с десятичными данными и полями переменной длины. Локальная память состоит из регистров общего назначения и с плавающей точкой, а также управляющих регистров. К средствам управле-

ния памятью относятся средства управления доступом к ОП и предвыборкой команд, буферная память, средства защиты памяти. Средства управления вводом-выводом обеспечивают приоритетный доступ программ через контроллеры (каналы) к периферийному оборудованию. К системным средствам относятся средства службы времени: часы астрономического времени, таймер, коммутатор и т.д.

Существует обязательный (стандартный) минимальный набор функциональных средств для каждого типа центрального процессора. Он включает в себя: регистры общего назначения, средства выполнения стандартного набора операций и средства управления вычислительным процессом. Конкретная реализация ЦП может различаться составом средств, способом их реализации, техническими параметрами.

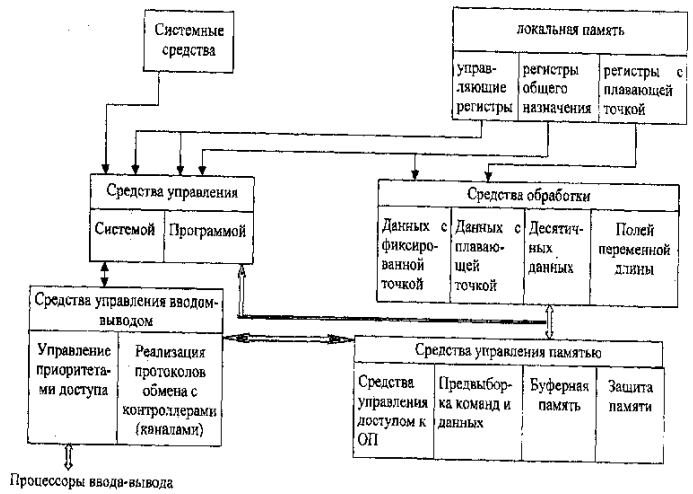


Рис.3.1. Логическая организация ЦП

Структурно все функциональные средства разбиваются на следующие устройства (рис. 3.2): центральное устройство управления (ЦУУ), арифметико-логическое устройство (АЛУ), устройство управления памятью (УУП), сверхоперативное запоминающее устройство (СОЗУ), устройство предвыборки команд и данных (УП) и интерфейс магистрали (ИМ).

Центральное устройство управления включает дешифратор команд, блок управления и блок прерываний. Дешифратор команд дешифрирует команды, которые поступают из блока предварительной выборки (очереди команд). Блок управления (БУ) формирует последовательности управляющих сигналов, которые поступают на все блоки процессора, обеспечивающие выполнение очередной команды и переход к следующей. Блок прерываний проводит анализ запросов на прерывания, формирует сигнал прерывания работы процессора и код (вектор) запроса с наивысшим приоритетом.

Арифметико-логическое устройство выполняет все арифметические и логические операции набора команд ЭВМ. В состав устройства входят традиционные арифметико-логические блоки, специализированные аппаратные средства (блок ускоренного умножения), буферные и рабочие регистры, иногда собственный блок управления. Во многих случаях выполнение операций с плавающей точкой осуществляется в отдельном блоке (процессоре), который имеет собственные регистры данных, управления и работает параллельно с центральным процессором.

Сверхоперативное ЗУ (регистровый файл) содержит регистры общего назначения (РОН), в которых хранятся данные и адреса.

Устройство управления памятью (диспетчер памяти) предназначено для сопряжения центрального процессора и подсистемы ввода-вывода с оперативной памятью. Оно состоит из блока сегментации и блока страничной адресации, осуществляющих двухступенчатое формирование физического адреса ячейки памяти: сначала в пределах сегмента, а затем в пределах страницы. Наличие блоков сегментации и страничной адресации, их одновременное функционирование обеспечивают максимальную гибкость проектируемой системы. Сегментация полезна для организации в памяти локальных модулей и является инструментом программиста, в то время как страницы нужны системному программисту для эффективного использования физической памяти системы.

Устройство предвыборки команд и данных включает блок предвыборки команд и внутреннюю кэш-память (кэш-память первого уровня). Первый осуществляет заполнение очереди команд, причем выборка из памяти производится в промежутках между магистральными циклами команд. Внутренняя кэш-память позволяет существенно повысить производительность процессора за счет буферизации в ней часто используемых команд и данных, сокращения числа обращений к оперативной памяти.

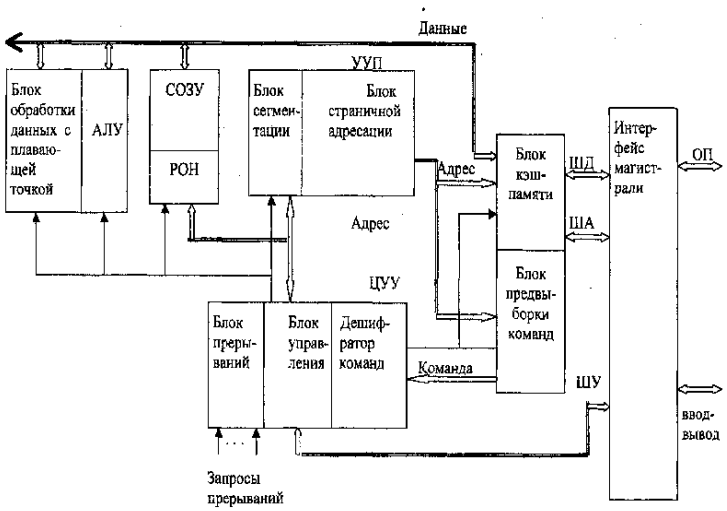


Рис.3.2.Структурная схема процессора

Интерфейс магистрали реализует протоколы обмена центрального процессора с памятью, контроллерами (каналами) ввода-вывода, другими активными устройствами системы. Обмен осуществляется с помощью шин данных, адреса и управления. Состав линий управления, тактовая сетка, магистральные циклы обмена существенно отличаются у различных типов процессоров.

В современных суперскалярных процессорах используется целый ряд параллельно функционирующих исполнительных устройств (от 2 до 6 устройств). В их состав могут входить:

- несколько целочисленных устройств;

- устройство плавающей точки;
- устройство выполнения переходов;
- устройство загрузки/записи.

Устройство выполнения переходов обрабатывает команды условных переходов. Если условия перехода доступны, то решение о направлений перехода принимается немедленно, в противном случае выполнение последующих команд продолжается по предположению (спекулятивно).

Пересылки данных между кэш-памятью данных, с одной стороны, и регистрами общего назначения и регистрами плавающей точки, с другой стороны, обрабатываются устройством загрузки/записи.

3.2. Регистровые структуры центрального процессора

Набор регистров и **их** структуры рассмотрим на примере процессоров с интеловской архитектурой. Можно выделить следующие группы регистров:

1. Основные функциональные регистры:
- регистры общего назначения (РОНы);
- указатель команд;
- регистр флагов;
- регистры сегментов.

2. Регистры процессора обработки чисел с плавающей точкой (FPU):
- регистры данных;
- регистр тегов;
- регистр состояния;
- регистр указателей команд и данных FPU;
- регистр управления FPU.

3. Системные регистры:
- регистры управления микропроцессора;
- регистры системных адресов.

4. Регистры отладки и тестирования.

Все 16-разрядные регистры микропроцессоров 8086, 80186, 80286 входят в состав набора 32-разрядных регистров. Регистры первых двух групп используются при выполнении прикладных программ, третьей группы _ системных, четвертой — при отладке и тестировании.

3.2.1. Основные функциональные регистры

Содержимое этих регистров определяется текущей задачей, т.е. в эти регистры автоматически загружается новое значение при переключении задач.

Регистры общего назначения

Восемь 32-разрядных регистров предназначены для хранения данных и адресов. Они поддерживают работу с данными разрядностью 1, 8, 16,32 и 64 бита, битовыми полями длиной от 1 до 32 бит и адресами размером 16 и 32 бита. Младшие 16 разрядов этих регистров (рис. 3.3) доступны отдельно при использовании соответствующего имени, например регистр EAX (имя AX для 16 разрядов).

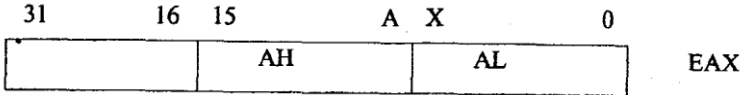


Рис. 3.3. Структура регистра общего назначения EAX

При операциях с байтами можно отдельно обращаться к младшему байту (разряды 0 - 7) и старшему байту (8-15) по именам AL и AH. Доступ к отдельным байтам обеспечивает дополнительную гибкость при операциях с данными.

Регистры сегментов и дескрипторы сегментов

Шесть 16-разрядных сегментных регистров (CS, SS, DS, ES, FS, GS) содержат значения селекторов сегментов, указывающих на текущие адресуемые сегменты памяти. С каждым из них связан программно-недоступный регистр дескриптора сегмента (рис. 3.4).

В защищенном режиме каждый сегмент может иметь размер от 1 байта до 4 Гбайт, в режиме реальных адресов максимальный размер сегмента составляет 64 Кбайта.

Селектор в CS обеспечивает обращение к текущему сегменту команд, селектор в SS — к текущему сегменту стека, селекторы в DS, ES, FS, GS — к текущим сегментам данных. Каждый регистр дескриптора содержит 32-разрядный размер сегмента и другие необходимые атрибуты.

Когда в регистр сегмента загружается новое значение селектора, содержимое соответствующего регистра дескриптора автоматически корректируется. В реальном режиме базовый адрес сегмента получается путем сдвига значения селектора на 4 разряда влево (20 разрядов), максимальный размер и атрибуты сегмента в реальном режиме имеют фиксированные значения.

Рис. 3.4. Регистры сегментов и соответствующие регистры дескрипторов

Указатель команд

Указатель команд (рис. 3.5) представляет собой 32-разрядный регистр с именем EIP, содержимое которого используется в качестве смещения при определении адреса следующей выполняемой команды. Смещение задается относительно базового адреса сегмента команд CS. Младшие 16 бит (0 — 15) содержат 16-разрядный указатель команд с именем IP, который используется при 16-разрядной адресации.

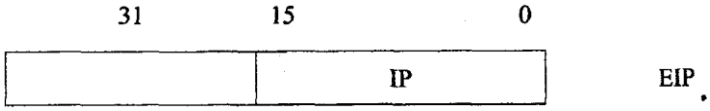


Рис. 3.5. Структура регистра указателя команд

Указатель команд непосредственно программисту недоступен. Его содержимое изменяется при выполнении команд передачи управления и прерываний.

Регистр флагов

Регистр флагов является 32-разрядным, имеет имя EFLAGS. Его разряды содержат признаки результата выполнения команды, управляют обработкой прерываний, последовательностью вызываемых задач, вводом/выводом и рядом ЛПУГИХ nnouenvo.

3.2.2. Регистры процессора обработки чисел с плавающей точкой

Набор регистров, входящих в блок (FPU), изображен на рис. 3.6.

Рис. 3.6. Регистры блока FPU

Регистр тегов FPU. Он содержит 16-разрядное слово, включающее восемь двухбитовых тегов. Каждый тег (признак) характеризует содержимое одного из регистров данных.

Тег определяет, является ли регистр пустым (незаполненным) - код 11 или в него введено конечное число — 00 (достоверное значение), или нуль -01, неопределенное значение (бесконечность) — 10 (нет числа и неподдерживаемый формат). Слово тегов позволяет оптимизировать функционирование FPU посредством идентификации пустых и непустых регистров данных, проверить содержимое регистра без сложного декодирования хранящихся в нем данных.

3.2.3. Системные регистры

Системные регистры управляют функционированием микропроцессора вцелом и режимами работы отдельных внутренних блоков: процессора с плавающей точкой, кэш-памятью, диспетчера памяти.

Эти регистры доступны только в защищенном режиме для программ.

Набор системных регистров включает три регистра управления (CRO, CR2, CR3) и четыре регистра системных адресов и сегментов.

Регистры управления 32-разрядные, служат для фиксации общего состояния процессора. Эти регистры вместе с регистрами системных адресов хранят информацию о состоянии процессора, которое затрагивает все задачи.

3.2.4. Регистры отладки и тестирования

Микропроцессор i486, например, имеет одиннадцать регистров отладки и тестирования (все они 32-разрядные). Из них б программно-

доступных регистров (DRO — DR3, DR6, DR7) поддерживают процесс отладки программ. Пять программно-доступных регистров (TR3 — TR7) поддерживают тестирование внутренних блоков: TR3 — TR5 используются для проверки кэшпамяти; TR6, TR7 — для тестирования механизма быстрого формирования адресов страниц.

3.3. Назначение, классификация и организация ЦУУ

Центральное устройство управления — это комплекс средств автоматического управления процессом передачи и обработки информации. ЦУУ вырабатывает управляющие сигналы (УС), необходимые для выполнения всех операций, предусмотренных системой команд, а также координирует работу всех узлов и блоков ЭВМ. В связи с этим можно считать ЦУУ преобразователем первичной командной информации, представленной программой решения задачи, во вторичную командную информацию, представляемую управляющими сигналами.

В общем случае ЦУУ формирует управляющие сигналы для реализации следующих функций:

- выборки из памяти кода очередной команды;
- расшифровки кода операции и признаков выбранной команды;
- выборки операндов и выполнения машинной операции;
- обеспечения прерываний при выполнении команд;
- формирования адреса следующей команды;
- учета состояний других устройств машины;
- инициализации работы контроллеров (каналов) ввода-вывода;
- организации контроля работоспособности ЭВМ.

Для дальнейшего рассмотрения характеристик и способов организации ЦУУ введем ряд определений.

Элементарное машинное действие, выполняемое по одному УС, называют **микрооперацией.** Набор микроопераций, выполняемых параллельно в одном машинном такте, называют **микрокомандой** Последовательность микрокоманд, обеспечивающих выполнение некоторой операции, предписанной командой, называют **микропрограммой.**

К основным характеристикам ЦУУ следует отнести:

- принцип формирования и развертывания временной последовательности УС;
- способ построения цикла работы ЦУУ и ЭВМ в целом;
- общая организация управления ЭВМ;
- способ синхронизации узлов и блоков ЭВМ. **По принципу формирования и развертывания временной последовательности УС различают ЦУУ:**
- аппаратного (схемного) типа, выполненным в виде управляющего автомата с жесткой логикой, в котором функции переходов и выходов реализуются набором логических элементов, а требуемое количество состояний автомата задается множеством запоминающих элементов;
- микропрограммного типа, в которых блок управления реализован

как блок микропрограммного управления (БМУ).

По способу построения рабочего цикла различают ЦУУ:

- с прямым циклом, когда на первом этапе производится выборка из памяти команды, а затем следуют этапы выполнения машинной операции;
- с обращенным циклом, когда сначала выдаются УС микроопераций для выполнения машинной операции по коду команды, поступившей в ЦУУ на предыдущем цикле (предвыборка команд), а затем из памяти выбирается код команды, которая будет исполняться в следующем цикле;
- с совмещением во времени циклов выполнения нескольких команд (конвейером команд).

По общей организации управление может быть центральным и смешанным. В первом случае в БУ вырабатываются все УС микроопераций для всех команд, выполняемых процессором ЭВМ. Во втором случае, кроме БУ центрального устройства управления, операционные и другие устройства процессора имеют собственные блоки местного управления. В последнем случае БУ вырабатывает сигналы для запуска в работу блоков местного управления.

По способу синхронизации работы ЭВМ в зависимости от числа тактов в цикле выполнения команды, различают ЦУУ с постоянным или переменным числом тактов. В микропрограмме рабочего цикла выделяют общую и специальную части. К общей относятся микрокоманды, исполняемые в цикле любой команды: выборки команды, анализа запросов на прерывание работы процессора. Они выполняются за постоянное число тактов. К специальной части относятся микрокоманды, по которым вырабатываются УС в зависимости от содержания операционной части исполняемой команды. В этом случае количество тактов будет переменным для различных команд. В современных ЭВМ с различной структурой используемых команд число тактов в рабочем цикле зависит от формата выбираемой команды, структуры ее адресной части и длины операндов.

По принципу организации циклов различают ЦУУ:

- синхронного типа, в которых время цикла может быть постоянным или переменным;
- асинхронного типа, в которых продолжительность цикла определяется фактическими затратами времени на выполнение каждой операции. В этом случае необходимо вырабатывать сигналы об окончании операции;
- смешанного типа, где частично реализуются оба предыдущих принципа организации циклов.

3.3.1. Центральное устройство управления микропрограммного типа

Микропрограммный принцип управления обеспечивает реализацию одной машинной команды путем выполнения определенной микропрограммы, интерпретирующей алгоритм выполнения данной операции. Совокупность микропрограмм, необходимая для реализации

системы команд ЭВМ, хранится в специальной памяти микропрограмм. Каждая микропрограмма состоит из определенной последовательности микрокоманд, которые после выборки из памяти преобразуются в набор управляющих сигналов.

Микрокоманда (МК) имеет операционно-адресную структуру. В операционной части МК размещается информация о микрооперациях (МО), одновременно выполняемых в блоках ЭВМ под управлением данной МК. В адресной части МК находится информация, необходимая для формирования адреса следующей микрокоманды.

Существуют различные способы организации операционной части МК:

- горизонтальное микропрограммирование;
- вертикальное микропрограммирование;
- смешанное микропрограммирование.

В первом случае операционная часть МК содержит столько разрядов, сколько различных МО выполняется в ЭВМ (число управляющих точек). Каждому разряду ставится в соответствие определенный УС, под действием которого выполняется соответствующая микрооперация. Таким образом, нет необходимости в преобразовании операционной части МК в управляющие сигналы. За счет этого сокращаются затраты времени на формирование УС. Недостатком данного метода является большая длина операционной части МК, что ведет к значительным затратам памяти микропрограмм.

По второму способу из всего множества M микроопераций выделяются подмножества, содержащие не более H совместно выполняемых в каждом такте МО. Номера МО кодируются двоичным кодом, разрядность которого определяется по формуле $m \geq \log_2 M$. Операционная часть МК должна содержать H полей, каждое из которых имеет разрядность m и определяет код номера микрооперации. В результате использования данного метода уменьшается длина МК, сокращаются затраты микропрограммной памяти, но возникает необходимость в дешифрировании полей операционной части МК, что приводит к увеличению затрат времени на выработку УС.

В настоящее время наибольшее распространение получил третий способ — смешанное микропрограммирование, в котором сочетаются первые два способа. В этом случае операционная часть МК содержит как коды номеров микроопераций, так и сами УС, соответствующие отдельным МО.

Адресная часть МК используется для определения адреса следующей МК.

Существуют два способа адресации микрокоманд:

- принудительная адресация;
- естественная адресация.

Принудительная адресация МК заключается в том, что в каждой МК указывается адрес следующей МК. Адрес следующей МК может задаваться безусловно, независимо от значений признаков (осведомительных сигналов) или выбираться по условию, определяемому текущими значениями осведомительных сигналов,

которые, в свою очередь, отображают текущее состояние операционных блоков процессора. Для этого в адресную часть МК кроме адресных полей включаются поля для задания условий (осведомительных сигналов).

При естественной адресации адрес следующей МК принимается равным увеличенному на единицу адресу предыдущей МК. В этом случае отпадает необходимость во введении адресной части в каждую МК. Если микрокоманды идут в естественном порядке, то процесс адресации реализуется счетчиком адреса МК. Для организации безусловных или условных переходов в микропрограмму включаются дополнительные управляющие МК.

Обобщенная структура блока микропрограммного управления (ЕМУ) представлена на рис. 3.7. Узел ФАМ предназначен для формирования адреса очередной МК с учетом значений адресной части (АЧ) предыдущей МК и множества *{x}* осведомительных сигналов. Микропрограммная память (МПП) хранит микропрограммы операций и по сформированному адресу в каждом такте выдает значение очередной МК, которое записывается в регистр микрокоманд (РМК). Поля операционной части (ОЧ), выбранной МК, при необходимости дешифрируются для выработки управляющих сигналов *{y}*. Первоначальное обращение к какой-либо микропрограмме осуществляется по начальному адресу (НА), который соответствует коду операции выполняемой команды.

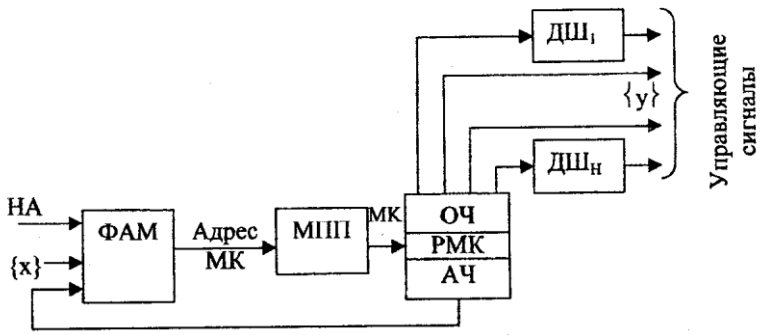


Рис.3.7. Обобщенная структура БМУ

С точки зрения физической реализации управления МПП делится на два вида: память с постоянно записанной информацией и память, допускающая перезапись информации. Память с постоянно записанной информацией (ПЗУ) работает только на чтение информации и, как правило, является более быстродействующей и простой по управлению, нежели память с перезаписью. В то же время память, допускающая перезапись, предоставляет больше дополнительных возможностей для повышения эффективности работы процессора за счет постоянного совершенствования алгоритмов выполнения операций.

Таким образом, использование в составе центрального устройства управления БМУ приводит к двухуровневому принципу управления процессом обработки данных. Первый уровень — это

система команд ЭВМ (программное управление), второй — микропрограммное управление. Возникает задача организации перехода от одного уровня к другому. На рис. 3.8 приведена упрощенная структура процессора, в котором решается эта задача. По содержимому счетчика адреса команд (СЧАК) из памяти программ (кэшпамяти) выбирается команда и записывается в регистр команд (РК). Код операции из РКОП подается на дешифратор начального адреса (ДШНА), который на выходе формирует адрес первой микрокоманды микропрограммы, соответствующей данному коду операции. ДШНА реализуется на ПЗУ или ПЛМ (программируемой логической матрице). Под управлением микрокоманд выполняются все последующие действия. Адрес операнда из РА передается в память данных, осуществляется выборка операнда и занесение его в регистр общего назначения (СОЗУ) или в АЛУ. В АЛУ выполняется определенная микропрограммой операция, результат записывается в РОН или память данных.

Анализ аппаратурной (схемной) и микропрограммной реализации устройства управления указывает на зависимость стоимости управления от величины набора команд и их сложности. Для сокращенного набора простых команд выгоднее использовать схемное управление, что и реализуется в RISC-процессорах. При расширенном составе сложных команд (как в CISC-процессорах) наиболее эффективно, с точки зрения затрат оборудования, микропрограммное управление. Однако оно приводит к увеличению затрат времени на выработку управляющих воздействий. Основным же преимуществом микропрограммного управления является его гибкость, которая позволяет повышать эффективность серийно выпускаемых и эксплуатируемых машин за счет введения новых средств математического обеспечения, использующих дополнительный набор команд и новые функции процессора. Модернизация алгоритмов или реализация дополнительных команд легко осуществляется путем изменения содержимого микропрограммной памяти. Наглядным примером использования данной возможности является технология MMX, разработанная фирмой Intel. В серийно выпускаемый процессор Pentium были добавлены 57 новых команд для параллельной обработки видео- и аудиоинформации. Аппаратурные средства процессора остались прежними, изменению подверглась лишь микропрограммная память.

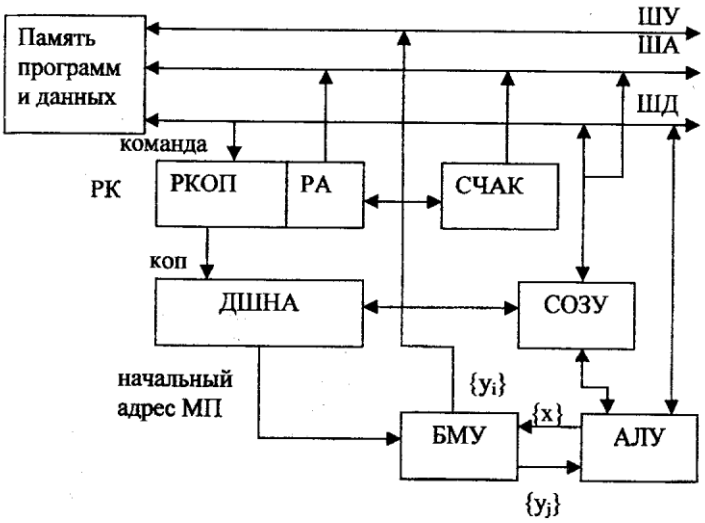


Рис.3.8. Процессор с микропрограммным управлением

33.2. Процедура выполнения команд

Стандартные фазы работы ЦП включают в себя выборку команды, вычисление адреса и выборку операндов, выполнение команды и запись результатов, обработку прерывания, изменение состояния процессора и системы в целом.

Выборка команд (ВК) — передача содержимого счетчика команд в регистр адреса памяти, считывание команды из основной памяти в регистр команды, модификация содержимого счетчика команд для выборки следующей
команды.

Выборка операнда (ВО) — вычисление адреса и обращение в основную память или к регистру локальной памяти. Операнд считывается и принимается в регистр АЛУ.

Арифметическая операция (АО) — инициализация (кодом операции) цикла работы устройства управления, которое, в свою очередь, управляет работой АЛУ, регистров и схем сопряжения. Результат выполнения передается в локальную или основную память и процессор переходит к выборке и выполнению следующей команды.

На рис. 3.9 показаны временные диаграммы обработки команды с разбиением на этапы (фазы) выполнения (а): последовательная обработка команд (б); обработка команд в режиме совмещения — конвейер команд (в).

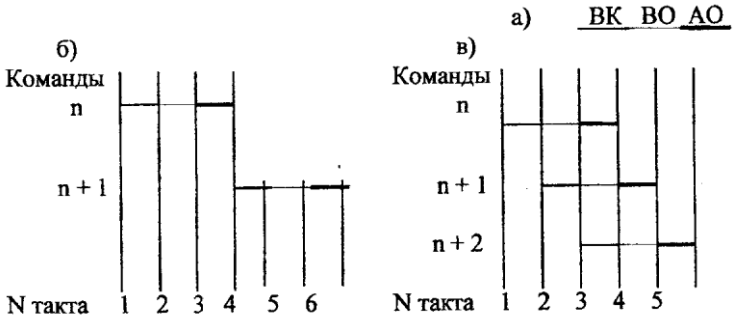


Рис. 3.9. Временные диаграммы обработки команд в процессоре:
а) этапы выполнения команды;

б) последовательное выполнение команд;

в) совмещенное выполнение команд (конвейеризация) »

Совмещенные принципы обработки (конвейер команд) существенно увеличивают пропускную способность процессора, однако эффективность их использования зависит от управления (синхронизации), числа уровней обработки.

Приостановка работы конвейера вызывает любая команда условного перехода в программе или взаимозависимость команд, т. е. использование следующей командой результатов предыдущей команды.

Следует учитывать, что совмещение обработки увеличивает объем оборудования и усложняет схемы управления тем сильнее, чем больше число уровней совмещения.

Все эти обстоятельства приходится учитывать при выборе числа уровней совмещения в каждом конкретном случае для получения заданных параметров и прежде всего удельных затрат (отношение производительности к стоимости). Опыт разработки ЭВМ общего назначения и проведенные исследования показывают, что технически и экономически целесообразной является совмещенная обработка 5-6 команд.

Для обеспечения непрерывности вычислительного процесса и сглаживания влияния логической зависимости команд в структуре ЦП используется блок прогнозирования ветвлений или устройство выполнения переходов.

В большинстве современных компьютеров используется конвейер команд.

3.3.3. Принципы организации системы прерывания программ

Во время выполнения ЭВМ текущей программы внутри машины и в связанной с ней внешней среде (технологический процесс, управляемый ЭВМ) "могут возникать события, требующие немедленной реакции на них со стороны машины.

Реакция состоит в том, что машина прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершению этой программы ЭВМ возвращается к выполнению прерванной программы.

Рассматриваемый процесс, называемый прерыванием программ, поясняется на рис. 3.10.

Принципиально важным является то, что моменты возникновения событий, требующих прерывания программ, заранее не известны и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, который называют **запросом прерывания.**

Программу, затребованную запросом прерывания, называют **прерывающей программой,** противопоставляя ее **прерываемой программе,** выполнявшейся в ЭВМ до появления запроса.

Запросы на прерывания могут возникать внутри самой ЭВМ и в ее внешней среде. К первым относятся, например, запросы при

возникновении в ЭВМ таких событий, как появление ошибки в работе ее аппаратуры, переполнение разрядной сетки, попытка деления на 0, выход из установленной для данной программы области памяти, затребование периферийным устройством операции ввода-вывода, завершение операции ввода-вывода периферийным устройством или возникновение при этой операции особой ситуации и др. Хотя некоторые из указанных событий порождаются самой программой, моменты их появления, как правило, невозможно предусмотреть. Запросы во внешней среде могут возникать от других ЭВМ, от аварийных и некоторых других датчиков технологического процесса и т.п.

Таким образом, запросы прерывания генерируются несколькими развивающимися параллельно во времени процессами, которые в некоторые моменты требуют вмешательства процессора.

К этим процессам, в частности, относится процесс выполнения самой программы, процесс контроля правильности работы ЭВМ, операции ввода -вывода, технологический процесс в управляемом машиной объекте и др.

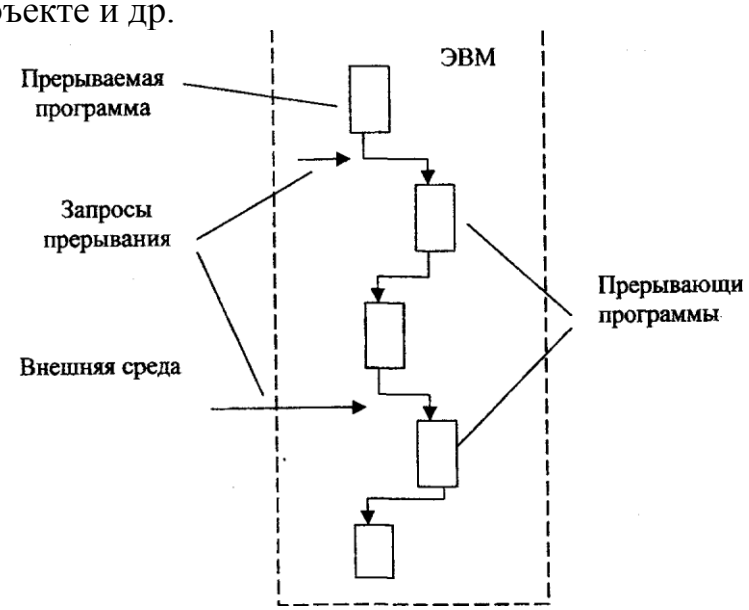


Рис.3.10. Процесс прерывания программы

Возможность прерывания программ - важное архитектурное свойство ЭВМ, позволяющее эффективно использовать производительность процессора при наличии нескольких, протекающих параллельно во времени, процессов, требующих в произвольные моменты времени управления и обслуживания со стороны процессора. В первую очередь это относится к организации параллельной во времени работы процессора и периферийных устройств машины, а также к использованию ЭВМ для управления в реальном времени технологическими процессами,

В некоторых машинах, наряду или вместо прерывания с переключением управления на другую программу, используется примитивное прерывание -так называемая приостановка, когда по соответствующему запросу приостанавливается выполнение программы и выполняется аппаратурными средствами некоторая процедура без

изменения содержания счетчика команд, а по ее окончании продолжается выполнение приостановленной программы.

Чтобы ЭВМ могла, не требуя больших усилий от программиста, реализовывать с высоким быстродействием прерывания программ, машине необходимо придать соответствующие аппаратурные и программные средства, совокупность которых получила название **системы прерывания программ.** В качестве аппаратных средств используется **контроллер прерывания** (блок прерывания).

Основными функциями системы прерывания являются:

- запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе;
- восстановление состояния прерванной программы и возврат к ней.

При наличии нескольких источников запросов прерывания между ними должны быть установлены **приоритетные соотношения,** определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающие: имеет право или нет данный запрос (прерывающая программа) прерывать ту или иную программу.

Характеристики системы прерывания

Для оценки эффективности систем прерывания могут быть использованы следующие характеристики.

1. Общее число запросов прерывания (входов в систему прерывания).

2. Время реакции — время между появлением запроса прерывания и моментом прерывания текущей программы. На рис. 3.11 приведена упрощенная временная диаграмма процесса прерывания.

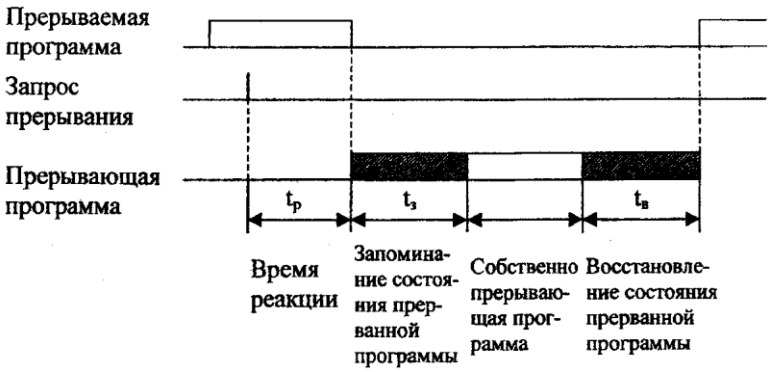


Рис.3.11. Упрощенная временная диаграмма процесса прерывания

Для одного и того же запроса задержки в исполнении прерывающей программы зависят от того, сколько программ со старшим приоритетом ждут обслуживания, поэтому время реакции определяют для запроса с наивысшим приоритетом ($t_р$).

Время реакции зависит от того, в какой момент допустимо прерывание. Большей частью прерывание допускается после окончания текущей команды. В этом случае время реакции определяется в основном длительностью выполнения команды.

Это время реакции может оказаться недопустимо большим для ЭВМ,

предназначенных для работы в реальном масштабе времени. В таких машинах часто допускается прерывание после любого такта выполнения команды (микрокоманды). Однако при этом возрастает количество информации, подлежащей запоминанию и восстановлению при переключении программ, так как в этом случае необходимо сохранять также и состояние в момент прерывания счетчика тактов, регистра кода операции и некоторых других узлов, поэтому такая организация прерывания возможна только в машинах с быстродействующей сверхоперативной памятью.

Имеются ситуации, в которых желательно немедленное прерывание. Если аппаратура контроля обнаружила ошибку, то целесообразно сразу же прервать операцию, пока ошибка не оказала влияние на следующие такты работы программы.

3. Затраты времени на переключение программ (издержки прерывания) равны суммарному расходу времени на запоминание и восстановление состояния программы (рис. 3.11):

$$t_{изд} = t_з + t_в.$$

4. Глубина прерывания - максимальное число программ, которые могут прерывать друг друга. Если после перехода к прерывающей программе и вплоть до ее окончания прием запросов прекращается, то говорят, что система имеет глубину прерывания, равную 1. Глубина равна n, если допускается последовательное прерывание до n программ. Глубина прерывания обычно совпадает с числом уровней приоритета в системе прерывания. На рис. 3.12 показаны процессы прерывания в системах с различной глубиной прерывания (предполагается, что приоритет каждого последующего запроса выше предыдущего). Система с большим значением глубины прерывания обеспечивает более быструю реакцию на срочные запросы.

Если запрос окажется не обслуженным к моменту прихода нового запроса от того же источника, то возникает так называемое **насыщение системы прерывания.**

В этом случае предыдущий запрос от данного источника будет машинально утерян, что недопустимо.

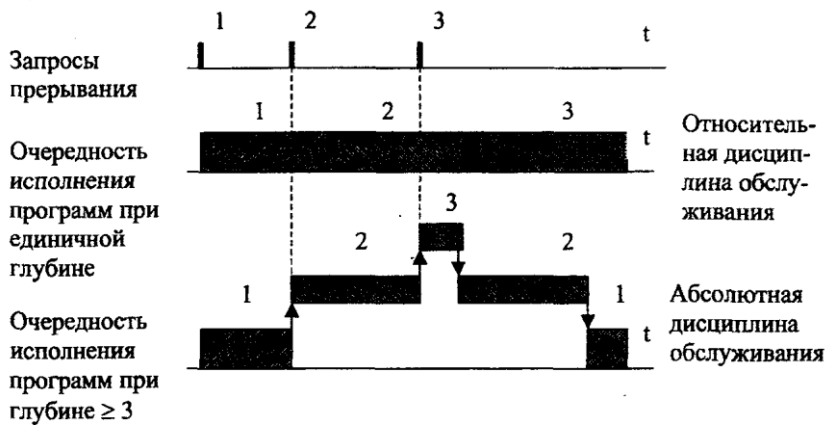


Рис.3.12. Процессы прерывания с различной глубиной прерывания и дисциплиной обслуживания

5. Число классов (уровней) прерывания. В ЭВМ число различных запросов (причин) прерывания может достигать нескольких десятков или сотен. В таких случаях часть запросов разделяют на отдельные классы или уровни.

Совокупность запросов, инициирующих одну и ту же прерывающую программу, образует класс или уровень прерывания (рис. 3.13).

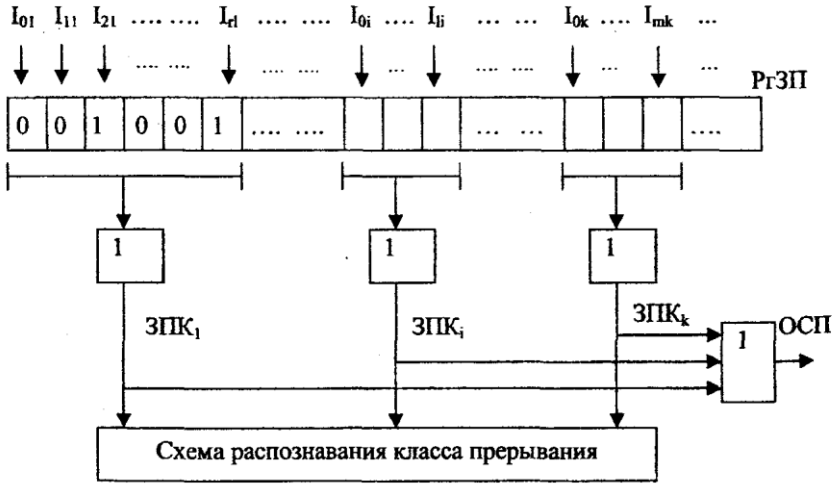


Рис.3.13. Разделение запросов на классы прерывания

Запросы всех источников прерывания поступают на регистр запросов прерывания РгЗП, устанавливая соответствующие его разряды в состояние 1, указывающее на наличие запроса прерывания определенного источника. Запросы классов прерывания $ЗПК_1$-$ЗПК_k$ формируются элементами ИЛИ, объединяющими разряды РгЗП, относящиеся к соответствующим классам (уровням). Еще одна схема ИЛИ формирует общий сигнал прерывания ОСП, поступающий в устройство управления процессора.

Информация о действительной причине прерывания, породившей запрос данного класса, содержится в коде прерывания, который отражает состояние разрядов РгЗП, относящихся к данному классу прерывания.

После принятия запроса прерывания на исполнение и передачу управления прерывающей программе соответствующий триггер РгЗП сбрасывается. Объединение запросов в классы прерывания позволяет уменьшить объем аппаратуры. но приводит к замедлению работы системы прерывания.

Программно-управляемый приоритет прерывающих программ

Относительная степень важности программ, их частота повторения, относительная степень срочности в ходе вычислительного процесса могут меняться, требуя установления новых приоритетных отношений. Поэтому во многих случаях приоритет между прерывающими программами не может быть зафиксирован раз и навсегда. Необходимо

иметь возможность изменять по мере необходимости приоритетные соотношения программным путем. Приоритет между прерывающими программами должен быть динамичным, т.е. программно управляемым.

В ЭВМ широко применяются два способа программно-управляемого приоритета прерывающих программ:

- использование порога прерывания;
- использование маски прерывания.

Порог прерывания. Этот способ позволяет в ходе вычислительного процесса программным путем изменить уровень приоритета процессора (а следовательно, и обрабатываемой в данный момент на процессоре программы) относительно приоритетов запросов источников прерывания (периферийных устройств), другими словами, задавать порог прерывания, т. е. минимальный уровень приоритета запросов, которым разрешается прерывать программу, идущую на процессоре.

Порог прерывания задается командой программы, устанавливающей в регистре порога прерывания код порога прерывания. Специальная схема выделяет наиболее приоритетный запрос, сравнивает его приоритет с порогом прерывания и, если он оказывается выше порога, вырабатывает общий сигнал прерывания, и начинается процедура прерывания.

Маска прерывания представляет собой двоичный код, разряды которого поставлены в соответствие запросам или классам (уровням) прерываний. Маска загружается командой программы в регистр маски (рис. 3.14).

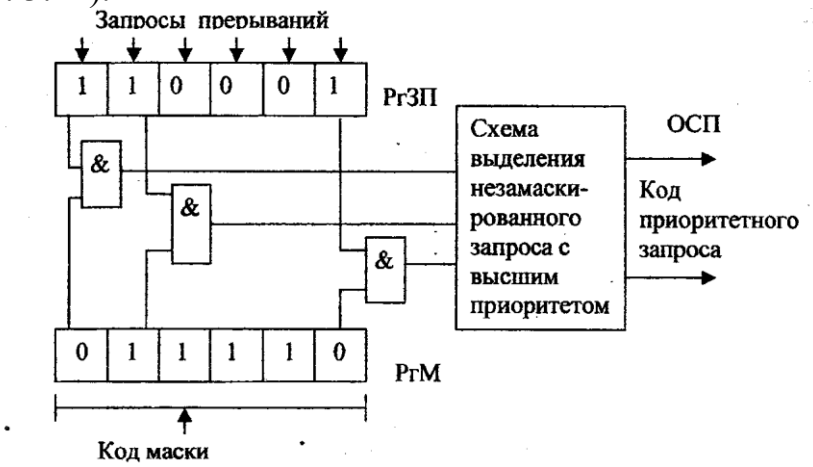


Рис.3.14. Маскирование прерываний

Состояние 1 в данном разряде регистра маски (РгМ) разрешает, а состояние 0 запрещает (маскирует) прерывание текущей программы от соответствующего запроса. Таким образом, программа, изменяя маску в регистре маски, может устанавливать произвольные приоритетные соотношения между программами без перекоммутации линий, по которым поступают запросы прерывания. Каждая прерывающая программа может установить свою маску. При формировании маски 1 устанавливаются в разряды, соответствующие запросам (прерывающим программам) с более высоким, чем у данной программы, приоритетом. Схемы И выделяют поступившие незамаскированные запросы

прерывания, из которых специальная схема выделяет наиболее приоритетный и формирует код его номера.

С замаскированным запросом, в зависимости от причины прерывания, поступают двояким образом: или он игнорируется, или запоминается, с тем чтобы осуществить затребованные действия, когда запрет будет снят. Например, если прерывание вызвано окончанием операции в ПУ, то его следует, как правило, запомнить, так как иначе ЭВМ останется неосведомленной о том, что ПУ освободилось.

Прерывание, вызванное переполнением разрядной сетки при арифметической операции, следует при его маскировании игнорировать, так как запоминание этого запроса может оказать действие на часть программы или другую программу, к которым это переполнение не относится.

Организация перехода к прерывающей программе

%,

Вектор начального состояния прерывающей программы называют **вектором прерывания.** Он содержит всю необходимую информацию для перехода к прерывающей программе, в том числе ее начальный адрес. Каждому запросу (уровню) прерывания соответствует свой вектор прерывания, способный инициировать выполнение соответствующей прерывающей программы. Векторы прерывания обычно находятся в специально выделенных фиксированных ячейках памяти (стеке).

Главное место в процедуре перехода к прерывающей программе занимает передача из соответствующего регистра (регистров) процессора в память (стек) на сохранение текущего вектора состояния прерываемой программы (чтобы можно было вернуться к ее исполнению) и загрузка в регистр (регистры) процессора вектора прерывания прерывающей программы, к которой при этом переходит управление процессором.

Наиболее гибким и динамичным является **векторное прерывание,** при котором источник прерывания, выставляя запрос прерывания, посылает в процессор (выставляет на шины интерфейса) код адреса в памяти своего вектора прерывания.

При векторном прерывании каждому запросу прерывания или, другими словами, устройству — источнику прерывания, соответствует переход к начальному адресу соответствующей прерывающей программы, задаваемому вектором прерывания.

3.4. Назначение, классификация и организация АЛУ

Арифметико-логическое устройство (АЛУ) является одной из основных функциональных частей процессора, осуществляющей непосредственное

преобразование информации.

Все операции, выполняемые в АЛУ, можно разделить на следующие группы:

- операции двоичной арифметики для чисел с фиксированной запятой;

- операции двоичной (или шестнадцатеричной) арифметики для чисел с плавающей запятой;

- операции десятичной арифметики (над числами, представленными в

двоично-десятичном коде);

- операции адресной арифметики (при модификации адресов команд);

- операции специальной арифметики;

- логические операции;

- операции над алфавитно-цифровыми полями.

Современные универсальные ЭВМ обычно реализуют операции всех приведенных выше групп, а специализированные ЭВМ часто не имеют аппаратуры для обработки чисел с плавающей запятой, десятичных чисел и операций над алфавитно-цифровыми полями. В этом случае эти операции выполняются специальными подпрограммами.

Основными являются арифметические и логические операции. К арифметическим операциям относятся сложение, вычитание, вычитание модулей ("короткие операции"), умножение и деление ("длинные операции"). Группу логических операций составляют операции дизъюнкции (логическое ИЛИ) и конъюнкции (логическое И) над многоразрядными двоичными словами, сравнение кодов на равенство. Специальные арифметические операции включают в себя нормализацию, арифметический сдвиг (сдвигаются только цифровые разряды, знаковый разряд остается на месте), логический сдвиг (знаковый разряд сдвигается вместе с цифровыми разрядами). Обширна группа операций редактирования алфавитно-цифровой информации.

Для выполнения перечисленных операций в АЛУ включаются следующие функциональные узлы:

- сумматор для выполнения суммирования и других действий над кодами операндов;

- регистры для хранения кодов операндов на время выполнения действий над ними;

- сдвигатели для сдвига кода на один или несколько разрядов вправо или влево;

- преобразователи для преобразования прямого кода числа в обратный или дополнительный код;

- комбинационные схемы для реализации логических операций, мультиплексирования данных, управляемой передачи информации, формирования признаков результата и т.д.

Регистры и в некоторых случаях сумматоры имеют цепи управления приемом, выдачей и сбросом кодов операндов. Логические операции, операции сдвига и преобразования кодов могут выполняться не только специальными устройствами, но и с помощью дополнительных связей регистров и сумматора. В зависимости от типов используемых для суммирования базовых элементов различают комбинационные и

накапливающие сумматоры.

Классификация АЛУ

По способу представления чисел различают АЛУ:

- для чисел с фиксированной запятой;
- для чисел с плавающей запятой;
- для десятичных чисел.

По способу действия над операндами АЛУ делятся на последовательные и параллельные. В параллельных АЛУ операнды представляются параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов. В последовательных АЛУ операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами. Такие АЛУ, как правило, используют конвейерный метод обработки, при котором совмещаются во времени фазы выполнения операции для различных разрядов операндов.

По выполняемым функциям АЛУ делятся на многофункциональные и функциональные (блочные). В блочном АЛУ операции над числами с фиксированной и плавающей запятой, десятичными и алфавитно-цифровыми полями, операции типа "умножение" выполняются в отдельных блоках. Такой подход позволяет увеличить скорость работы АЛУ за счет использования быстродействующих блоков, а также за счет организации параллельной работы этих блоков. Однако в этом случае значительно возрастают затраты оборудования.

В многофункциональных АЛУ всевозможные операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

По структурной организации АЛУ можно разделить на устройства, имеющие:

- регистровую структуру с непосредственными связями и закрепленной

логикой;

- магистральную структуру с сосредоточенной памятью и логикой. Арифметико-логические устройства первого типа базируются на принципе закрепления логических схем, используемых для выполнения микроопераций, за каждым из регистров. Так, на рис. 3.15 регистры Р1 и Р2 выполняют функции приема, хранения и выдачи операндов, поступающих из регистров общего назначения (РОН) процессора или КЭШ-памяти данных. С регистром Р1 непосредственно связан преобразователь кода ПК1. Комбинационный сумматор КСМ объединен с регистром РЗ по схеме накапливающего сумматора, с которым непосредственно связаны ПК2 и комбинационная схема КС для мультиплексирования входных данных. На регистре РЗ выполняются микрооперации сдвига вправо или влево и сброс. Регистр Р4 выполняет микрооперации сдвига и непосредственно связан с преобразователем кода ПКЗ.

Таким образом, в данной структуре функции хранения и

преобразования
информации выполняются одним и тем же операционным блоком.

Магистральная структура АЛУ отличается тем, что в ней регистры и схемы для преобразования информации выделены в отдельные блоки, связанные между собой по входам и выходам. В этом случае блок регистров (БР) выполняет функции приема, хранения, выдачи операндов и результатов, а операционный блок (ОБ) выполняет весь необходимый набор микроопераций над словами, хранимыми в БР. В данной структуре блок регистров может быть реализован двумя способами: либо как совокупность отдельных регистров с индивидуальными схемами управления, либо как сверхоперативное адресное запоминающее устройство.

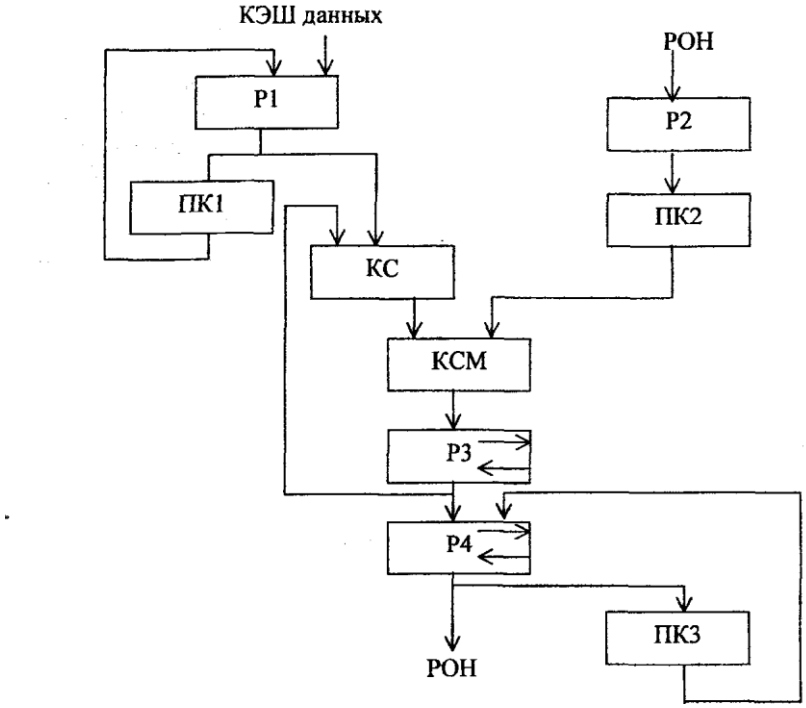


Рис.3.15. Регистровая структура с закрепленной логикой

Структура операционного блока имеет следующие модификации:

- последовательное соединение операционных узлов;
- параллельное соединение операционных узлов.

В первом случае (рис. 3.16) преобразователь кода ПК, комбинационный сумматор КСМ и сдвигатель СДВ соединены последовательно, причем входы ПК и КСМ связаны с выходными шинами блока регистров, а выход СДВ - с входной шиной БР. Такая организация операционного блока дает возможность выполнять с высокой скоростью последовательности микроопераций, обеспечивающей вычисление одного слова-

Во втором случае (рис. 3.17) операционные узлы (СМ, СДВ, ПК, КС) подсоединяются к входным и выходным шинам БР параллельно, что позволяет выполнять несколько микроопераций одновременно.

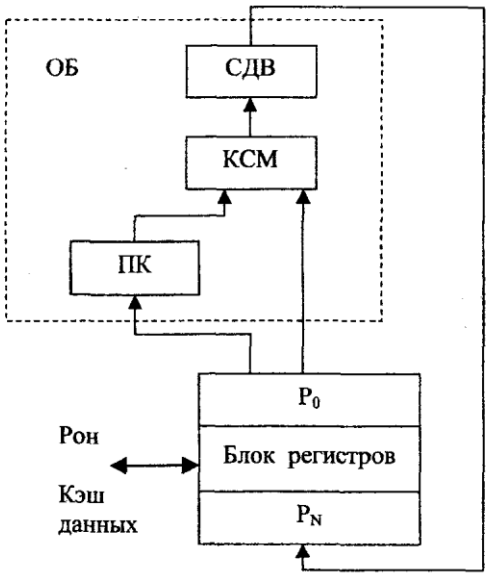
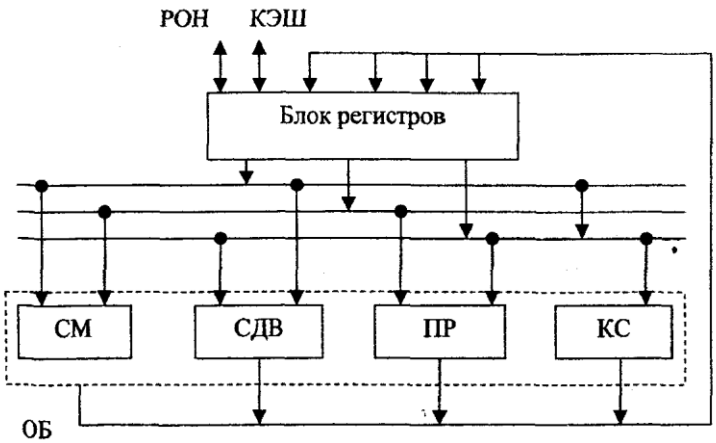


Рис. 3.16. Магистральная структура с последовательным соединением операционных узлов



Рнс.3.17. Магистральная структура с параллельным соединением операционных узлов

Обобщенная структурная схема АЛУ

Обобщенная структурная схема АЛУ (рис. 3.18) включает:

- блок регистров для приема и размещения операндов и результатов;
- операционный блок, в котором осуществляется преобразование операндов в соответствии с реализуемыми алгоритмами;
- схемы контроля, обеспечивающие непрерывный оперативный контроль и диагностирование ошибок;
- блок управления (БУ), в котором после приема кода операции (КОП) из центрального устройства управления формируются управляющие сигналы (УС), координирующие взаимодействие всех узлов АЛУ между собой и с другими блоками процессора.

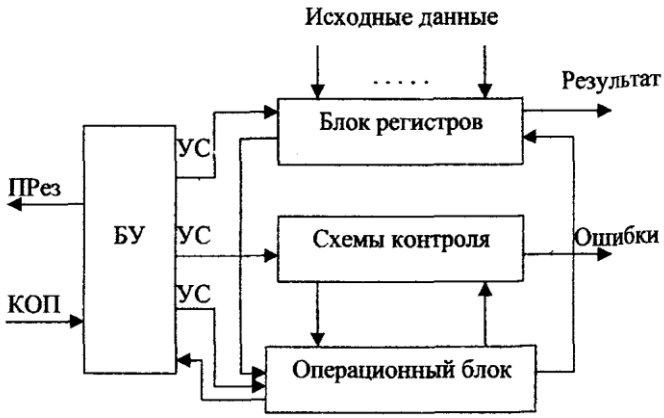


Рис.3.18. Обобщенная схема АЛУ

Блок регистров связан с ГОН центрального процессора и кэш-памятью данных.

Иногда АЛУ не содержит своего БР, в этом случае операционный блок непосредственно работает с регистрами общего назначения процессора. Для оперативного управления выполнением операции в ОБ на разных этапах анализируется преобразуемая информация и формируются сигналы признаков (флаги), которые используются в БУ для выработки и посылки в процессор сигнала признака результата (ПРез).

Для оценки АЛУ используются следующие характеристики: множество выполняемых операций, разрядность, время выполнения операций, надежностные и энергетические характеристики.

Методы повышения быстродействия АЛУ

Одним из эффективных и широко используемых методов повышения быстродействия АЛУ является реализация **принципа локального параллелизма.** Суть этого принципа заключается в распараллеливании во времени алгоритма выполнения отдельной команды на ряд независимых этапов и их реализации на различных операционных узлах (СМ, СДВ, ПК и т.д.) АЛУ.

Другой хорошо известный метод увеличения быстродействия - **конвейерная обработка.** Операционный блок разбивается на несколько частей -ступеней (уровней) конвейера. На каждой ступени выполняется определенная стадия операции. Совмещение стадий выполнения нескольких операций на различных ступенях конвейера приводит к тому, что реализация следующей операции начинается в нём до окончания предыдущей. Это значительно увеличивает быстродействие операционного блока. Такую организацию ОБ стали называть **арифметическим конвейером.**

На рис. 3.19 показана схема конвейерного сумматора с плавающей запятой. Конвейер содержит четыре ступени. Результат выполнения каждой стадии операции фиксируется на регистрах. Когда конвейер полностью заполнен, то стадия 4 (нормирование) выполняется, например, для первой пары операндов, стадия 3 (предварительной обработки) - для второй пары операндов, стадия 2 (сложение мантисс) - для третьей пары,

стадия 1 (сравнение порядков) - для четвертой пары операндов. В каждый последующий такт времени на выходе конвейера будет формироваться результат выполнения операции для каждой пары операндов.

Еще один способ сокращения длительности выполнения многотактных операций типа умножения является разработка и использование **эффективных алгоритмов.** Ускорение выполнения операции умножения достигается одновременным анализом нескольких разрядов множителя, использованием быстрых сумматоров с сохранением переносов и реализацией конвейерного метода обработки. Такой подход широко используется при создании функционально независимых блоков ускоренного умножения (умножителей).

Развитием системы команд универсальных ЭВМ, в том числе и персональных компьютеров, стало **введение векторных операций** — операций над упорядоченными массивами данных (у супер ЭВМ векторные операции появились давно).

В связи с этим в структуре процессора наблюдается специализация устройств по типам операндов: скалярные и векторные. В составе процессора появляются регистровая память и средства обработки двух типов: скалярные и векторные.

К векторным средствам обработки относятся:

- один или несколько арифметических'конвейеров для обработки элементов векторов;

- векторные регистры для хранения векторной информации. Векторные средства обработки информации позволяют увеличить производительность ЭВМ в несколько раз.

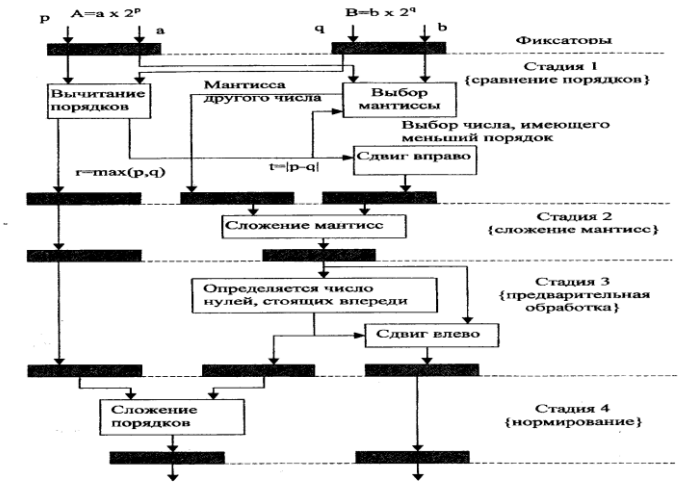


Рис.3.19. Конвейерный сумматор с плавающей запятой

4. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПОДСИСТЕМЫ ПАМЯТИ ЭВМ И ВС 4.1. Иерархическая структура памяти ЭВМ

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации.

Основными характеристиками отдельных устройств памяти (запоминающих устройств) являются емкость памяти, быстродействие и стоимость хранения единицы информации (бита).

Емкость памяти определяется максимальным количеством данных, которые могут в ней храниться. Емкость измеряется в двоичных единицах (битах), машинных словах, но большей частью в байтах (1 байт = 8 бит). Часто емкость памяти выражают через число К = 1024, например, Кбит — килобит, Кбайт — килобайт, 1024 Кбайт = 1 Мбайт (Мегабайт), 1024 Мбайт = 1 Гбайт
(гигабайт), 1024 Гбайт = 1 Тбайт (терабайт).

Быстродействие (задержка) памяти определяется временем доступа и длительностью цикла памяти. Время доступа представляет собой промежуток времени между выдачей запроса на чтение и моментом поступления запрошенного слова из памяти. Длительность цикла памяти определяется минимальным временем между двумя последовательными обращениями к памяти.

Требования к увеличению емкости и быстродействия памяти, а также к снижению ее стоимости являются **противоречивыми.** Чем больше быстродействие, тем технически труднее достигается и дороже обходится увеличение емкости памяти. Стоимость памяти составляет значительную часть общей стоимости ЭВМ. Исходя из этого, память ЭВМ организуется в виде иерархической структуры (рис. 4.1) запоминающих устройств, обладающих различным быстродействием и емкостью. Чем выше уровень, тем выше быстродействие соответствующей памяти, но меньше её емкость. К верхнему (сверхоперативному) уровню относятся регистры операционных и управляющих блоков процессора, сверхоперативная память, управляющая память, буферная память. На втором уровне находится основная или оперативная память. На последующих уровнях размещается внешняя и архивная память. Система управления памятью обеспечивает обмен информационными блоками между уровнями, причем обычно первое обращение к блоку информации вызывает его перемещение с низкого медленного уровня на более высокий. Это позволяет при последующих обращениях к данному блоку осуществлять его выборку с более быстродействующего уровня памяти.

Сравнительно небольшая емкость оперативной памяти (8 — 64 Мбайта) компенсируется практически неограниченной емкостью внешних запоминающих устройств. Однако эти устройства сравнительно медленные — время обращения за данными для магнитных дисков составляет десятки микросекунд. Для сравнения: цикл обращения к оперативной памяти (ОП) составляет 50 не. Исходя из этого, вычислительный процесс должен протекать с возможно

меньшим числом обращений к внешней памяти.

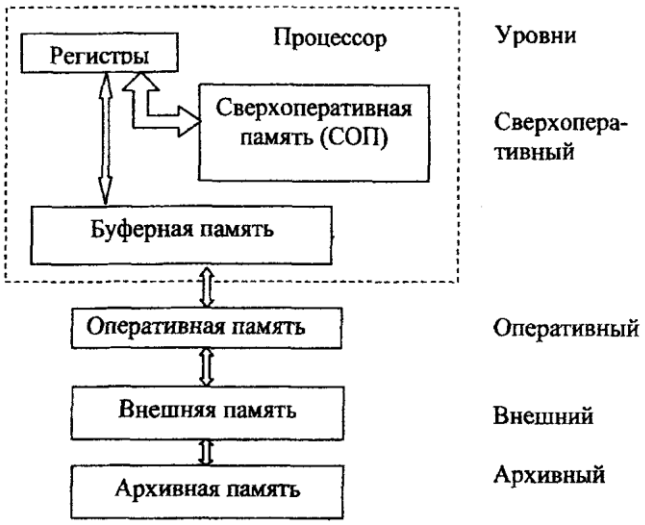


Рис.4.1. Иерархическая струюура памяти

Непрерывный рост производительности ЭВМ проявляется, в первую очередь, в повышении скорости работы процессора. Быстродействие ОП также растет, но все время отстает от быстродействия аппаратных средств процессора в значительной степени потому, что одновременно происходит опережающий рост её емкости, что делает более трудным уменьшение времени цикла работы памяти. Вследствие этого быстродействие ОП часто оказывается недостаточным для обеспечения требуемой производительности ЭВМ. Это проявляется в несоответствии пропускных способностей процессора и ОП. Возникающая проблема выравнивания их пропускных способностей решается путем использования сверхоперативной буферной памяти небольшой емкости и повышенного быстродействия, хранящей команды и данные, относящиеся к обрабатываемому участку программы.

При обращении к блоку данных, находящемуся на оперативном уровне, его копия пересылается в сверхоперативную буферную память (СБП). Последующие обращения производятся к копии блока данных, находящейся в СБП. Поскольку время выборки из сверхоперативной буферной памяти t^y (5 ис) много меньше времени выборки из оперативной памяти ton, введение в структуру памяти СБП приводит к уменьшению эквивалентного времени обращения <э по сравнению с ton:

$$t_э = t_{сбп} + \alpha t_{оп},$$

где а = (1 — q) и q — вероятность нахождения блока в СВП в момент обращения к нему, т.е. вероятность «попадания». Очевидно, что при высокой вероятности попадания эквивалентное время обращения приближается к времени обращения к СБП.

В основе такой организации взаимодействия ОП и СБП лежит принцип локальности обращений, согласно которому при выполнении какой-либо программы (практически для всех классов задач) большая

часть обращений в пределах некоторого интервала времени приходится на ограниченную область адресного пространства ОП, причем обращения к командам и элементам данных этой области производятся многократно. Это позволяет копии наиболее часто используемых участков программ и некоторых данных загрузить в СБП и таким образом обеспечить высокую вероятность попадания q. Высокая эффективность применения СБП достигается при $q \geq 0{,}9$.

Буферная память не является программно доступной. Это значит, что она влияет только на производительность ЭВМ, но не должна оказывать влияния на программирование прикладных задач. Поэтому она получила название кэш-памяти (в переводе с английского - тайник). В структуре одних ЭВМ используется объединенная кэш-память команд и данных, в других ЭВМ — раздельные кэш-памяти для команд и для данных. Кэш-память, входящую в состав процессора, называют кэш-памятью первого уровня. В современных компьютерах применяют кэш-память второго уровня, которая находится между процессором и ОП и еще больше повышает производительность ЭВМ.

4.2. Организация внутренней памяти процессора

Регистровая структура процессора была рассмотрена в разделе 3. Обращает на себя внимание ставший стандартным в современной архитектуре ЭВМ прием организации регистров общего назначения в виде **сверхоперативного ЗУ с прямой адресацией** (короткие адреса регистров размещаются в команде). В машинах с коротким словом, вынуждающим прибегать к одноадресным командам, один из общих регистров выделяется в качестве аккумулятора — регистра, в котором находится один из операндов и в который помещается результат операции. Регистр аккумулятор в явном виде в команде не адресуется, используется подразумеваемая адресация. В этом случае СОЗУ с прямой адресацией (рис. 4.2, а) состоит из совокупности регистров P[0"],..., P[M], связанных с входной X и выходной Z шинами, по которым передаются n — разрядные слова. Адрес регистра, к которому производится обращение с целью записи или чтения (управляющий сигнал ЗП/ЧТ) информации, поступает по шине A. Дешифратор адреса ДША формирует управляющие сигналы 0, 1,..., M, подключающие регистр с заданным адресом к шинам СОЗУ.

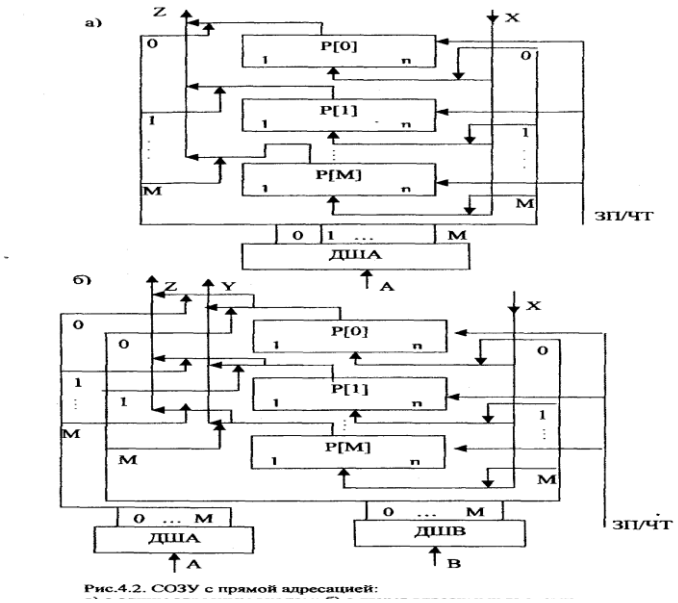


Рис.4.2. СОЗУ с прямой адресацией:

При использовании двухадресных команд типа «регистр-регистр», в которых указываются адреса двух операндов, расположенных в регистрах, и результат операции помещается по одному из этих адресов, данная организация СОЗУ становится неэффективной. Это связано с тем, что за один машинный такт может быть выбрано содержимое только одного регистра. Для реализации таких команд за один такт СОЗУ строится по схеме, приведенной на рис. 4.2, б.

СОЗУ состоит из совокупности регистров, связанных с входной шиной X и двумя выходными шинами Y, Z. Адреса регистров, к которым производится обращение с целью чтения информации, поступают по шинам A и B. Адрес регистра для записи информации поступает по входу B. Дешифраторы адресов ДША и ДШВ формируют управляющие сигналы 0, 1,... , M, подключающие два регистра к выходным шинам при чтении и один регистр к входной шине при записи.

Стековая память, реализующая безадресное задание операндов, является эффективным элементом архитектуры ЭВМ. Стек представляет собой группу последовательно пронумерованных регистров (аппаратный стек) или ячеек памяти, снабженных указателем стека (обычно регистром), в котором автоматически при записи и считывании устанавливается номер (адрес) первой свободной ячейки стека (вершина стека). При операции записи заносимое в стек слово помещается в свободную ячейку стека, а при считывании из стека извлекается последнее поступившее в него слово. Таким образом, в стеке реализуется правило «последний пришел - первый ушел» - магазинная адресация.

Механизм стековой адресации поясняется на рис. 4.3. Для реализации магазинной адресации используется счетчик адреса СЧА, который перед началом работы устанавливается в состояние ноль, и память (стек) считается пустой. Состояние СЧА определяет адрес первой свободной ячейки. Слово загружается в стек с входной шины X в момент поступления сигнала записи

ЗП.

По сигналу ЗП слово X записывается в регистр P[СЧА], номер которого

определяется текущим состоянием счетчика адреса, после чего с задержкой D, достаточной для выполнения микрооперации записи P[СЧА]:=X, состояние счетчика увеличивается на единицу. Таким образом, при последовательной загрузке слова A, B и C размещаются в регистрах с адресами P[S], P[S + 1] и P[S + 2], где S — состояние счетчика на момент начала загрузки. Операция чтения слова из ЗУ инициируется сигналом ЧТ, при поступлении которого состояние счетчика уменьшается на единицу, после чего на выходную шину Y поступает слово, записанное в стек последним. Если слова загружались в стек в порядке A. B, C, то они могут быть прочитаны только в обратном

порядке C, B, A.

В современных архитектурах процессоров и микропроцессоров стек и стековая адресация широко используется при организации переходов к подпрограммам и возврата из них, а также в системах прерывания.

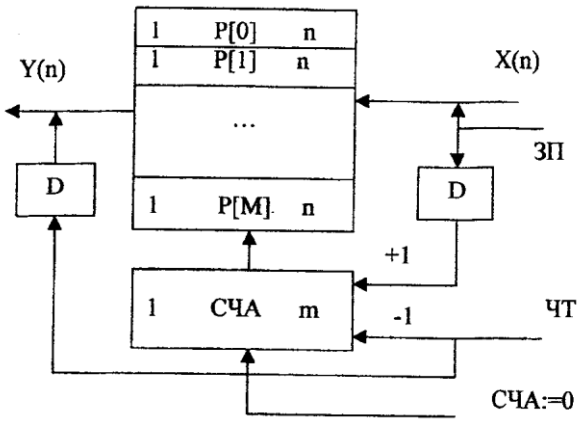


Рис.4.3. Стек с магазинной адресацией

4.3. Способы организация кэш-памяти

43.1. Общие сведения

В функциональном отношении кэш-память рассматривается как буферное ЗУ, размещённое между основной (оперативной) памятью и процессором. Основное назначение кэш-памяти — кратковременное хранение и выдача активной информации процессору, что сокращает число обращений к основной памяти, скорость работы которой меньше, чем кэш-памяти.

За единицу информации при обмене между основной памятью и кэш-памятью принята строка, причём под строкой понимается набор слов, выбираемый из оперативной памяти при одном к ней обращении. Хранимая в оперативной памяти информация представляется, таким образом, совокупностью строк с последовательными адресами. В любой момент времени строки в кэш-памяти представляют собой копии строк из некоторого их набора в ОП, однако расположены они необязательно в такой же последовательности, как вОП.

Построение кэш-памяти может осуществляться по различным принципам, которые будут рассмотрены ниже. При использовании принципа ассоциативной памяти каждая строка в кэш-памяти связана с так называемым **адресным тегом.** Адресный тег — это расширенный адрес, который объединяет адреса всех слов, принадлежащих данной строке. Он указывает, какую строку в ОП представляет данная строка в кэш-памяти.

Типовая структура кэш-памяти

Рассмотрим типовую структуру кэш-памяти (рис. 4.4), включающую основные блоки, которые обеспечивают её взаимодействие с ОП и центральным процессором.

Строки, составленные из информационных слов, и связанные с ними адресные теги хранятся в накопителе, который является основой кэш-памяти. Адрес требуемого слова, поступающий от центрального процессора (ЦП), вводится в блок обработки адресов, в котором реализуются принятые в данной кэш-памяти принципы использования адресов при организации их сравнения с адресными тегами. Само

сравнение производится в блоке сравнения адресов (БСА), который конструктивно совмещается с накопителем, если кэш-память строится по схеме ассоциативной памяти. Назначение БСА состоит в выявлении попадания или промаха при обработке запросов от центрального процессора. Если имеет место кэш-попадание (т.е. искомое слово хранится в кэш-памяти, о чём свидетельствует совпадение кодов адреса, поступающего от центрального процессора, и одного из адресов некоторого адресного тега), то соответствующая строка из кэш-памяти переписывается в регистр строк. С помощью селектора-демультиплексора из неё выделяется искомое слово, которое и направляется в центральный процессор. В случае промаха с помощью блока формирования запросов осуществляется инициализация выборки из ОП необходимой строки. Адресация ОП при этом производится в соответствии с информацией, поступившей от центрального процессора. Выбираемая из памяти строка вместе со своим адресным тегом помещается в накопитель и регистр строк, а затем искомое слово передается в центральный процессор.

Для высвобождения места в кэш-памяти с целью записи выбираемой из ОП строки одна из строк удаляется. Определение удаляемой строки производится посредством блока замены строк, в котором хранится информация, необходимая для реализации принятой стратегии обновления находящихся в накопителе строк.

4.3.2. Способы размещения данных в кэш-памяти

Существует четыре способа размещения данных в кэш-памяти: прямое распределение, полностью ассоциативное распределение, частично ассоциативное распределение и распределение секторов. Ниже подробно описан каждый способ размещения и его механизм преобразования адресов. Для того, чтобы конкретизировать описание, положим, что кэш-память может содержать 128 строк, размер строки - 16 слов, а основная память может содержать 16384 строк. Для адресации основной памяти используется 18 бит.

Из них старшие 14 показывают адрес строки, а младшие 4 бит — адрес слова внутри этой строки. При одном обращении к памяти выбирается одна строка. 128 строк кэш-памяти указываются 7-разрядными адресами.

Рис.4.4. Типовая структура кэш-памяти

Прямое распределение

При прямом распределении место хранения строк в кэш-памяти однозначно определяется по адресу строки.

Структура кэш-памяти показана **на** рис. 4.5.

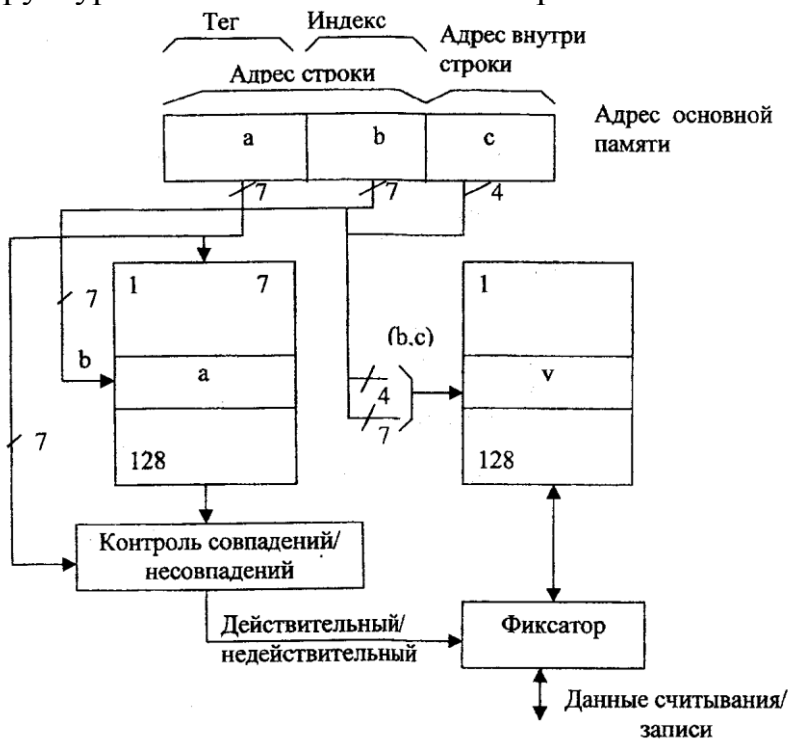


Рис.4.5. Структура кэш-памяти с прямым распределением

Адрес основной памяти состоит из 14-разрядного адреса строки и 4-разрядного адреса слова внутри этой строки. Адрес строки подразделяется на тег (старшие 7 бит) и индекс (младшие 7 бит).

Для того, чтобы поместить в кэш-память строку из основной памяти с адресом bn, выбирается область внутри кэш-памяти с адресом be, который равен 7 младшим битам адреса строки bn. Преобразование из bn в Ьс сводится только к выборке младших 7 бит адреса строки. По адресу be в кэш-памяти может быть помещена любая из 128 строк основной памяти, имеющих адрес, 7 младших битов которого равны адресу be. Для того чтобы определить, какая именно строка хранится в данное время в кэш-памяти, используется память ёмкостью 7 бит х 128 слов, в которую помещается по соответствующему адресу в качестве тега 7 старших битов адреса строки, хранящейся в данное время по адресу be кэш-памяти. Это специальная память, называемая теговой памятью. Память, в которой хранятся строки, помещенные в кэш, называются памятью данных. В качестве адреса теговой памяти используются младшие 7 битов адреса строки.

Из теговой памяти считывается тег. Параллельно этому осуществляется доступ к памяти данных с помощью 11 младших битов адреса основной памяти (используется 7 разрядов индекса и 4 разряда адреса слова внутри строки). Если считанный из теговой памяти тег и старшие 7 бит адреса основной памяти совпадают, то это означает, что данная строка существует в памяти данных, т.е. осуществляется кэш-попадание.

Если тег отличается от старших 7 бит (кэш-промах), то из основной памяти считывается соответствующая строка, а из кэш-памяти удаляется строка, определяемая 7 младшими разрядами адреса строки, и на его место помещается строка, считанная из основной памяти. Осуществляется также обновление соответствующего тега в теговой памяти.

Способ прямого распределения реализовать довольно просто, однако из-за того, что место хранения строки в кэш-памяти однозначно определяется по адресу строки, вероятность сосредоточения областей хранения строк в некоторой части кэш-памяти высока, т.е. замены строк будут происходить довольно часто. В такой ситуации эффективность кэш-памяти заметно снижается.

Полностью ассоциативное распределение

При полностью ассоциативном распределении допускается размещение каждой строки основной памяти на месте любой строки кэш-памяти. Структура кэш-памяти с полностью ассоциативным распределением представлена на рис. 4.6.

Адрес основной памяти состоит из 14-разрядного адреса строки (тега) и 4-разрядного адреса внутри строки.

При полностью ассоциативном распределении механизм преобразования адресов должен быстро дать ответ, существует ли копия строки с произвольно указанным адресом в кэш-памяти, и если существует, то по какому адресу. Для этого необходимо, чтобы теговая память являлась ассоциативной памятью. Входной информацией для ассоциативной памяти тегов является тег -14-разрядный адрес строки, а выходной информацией - адрес строки внутри кэш-памяти. Каждое слово теговой памяти состоит из 14-разрядного тега и 7-разрядного адреса строки внутри кэш-памяти. Ключом для поиска адреса строки внутри кэш-памяти является тег (старшие 14 разрядов адреса основной памяти).

При совпадении ключа с одним из тегов теговой памяти (кэш-попадание) происходит выборка соответствующего данному тегу адреса и обращение к памяти данных. Входной информацией для памяти данных является 11-разрядное слово (7 бит адреса строки и 4 бит адреса слова в данной.

При несовпадении ключа ни с одним из тегов теговой памяти (кэш-Промах) осуществляется обращение к основной памяти и чтение необходимой строки. По этому способу при замене строк кандидатом на удаление могут быть все строки в кэш-памяти.

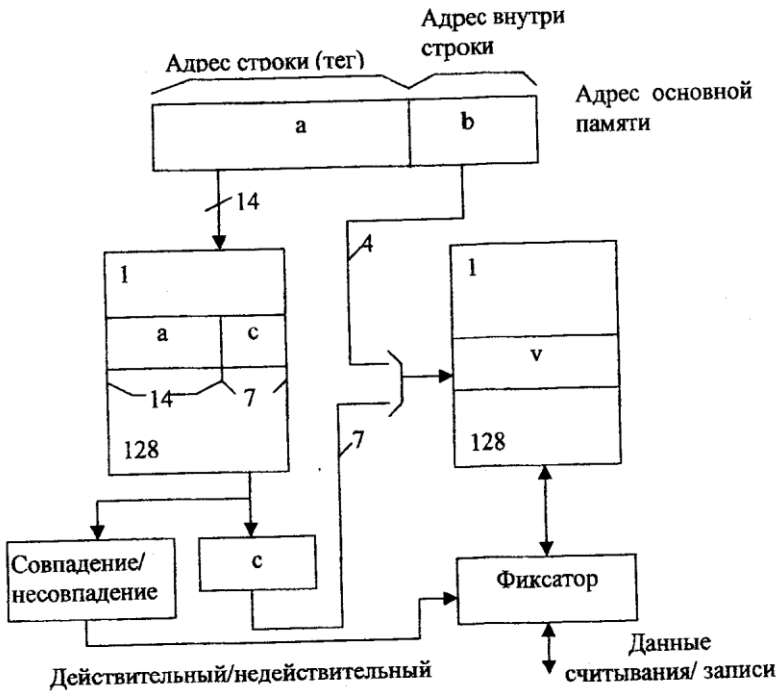


Рис.4.6. Структура кэш-памяти с полностью ассоциативным распределением

Частично ассоциативное распределение

При данном способе несколько соседних строк (фиксированное число, не менее двух) из 128 строк кэш-памяти образуют структуру, называемую

Структура кэш-памяти, основанная на использовании частично ассоциативного распределения, показана на рис. 4.7. В данном случае в одну группу входят 4 строки.

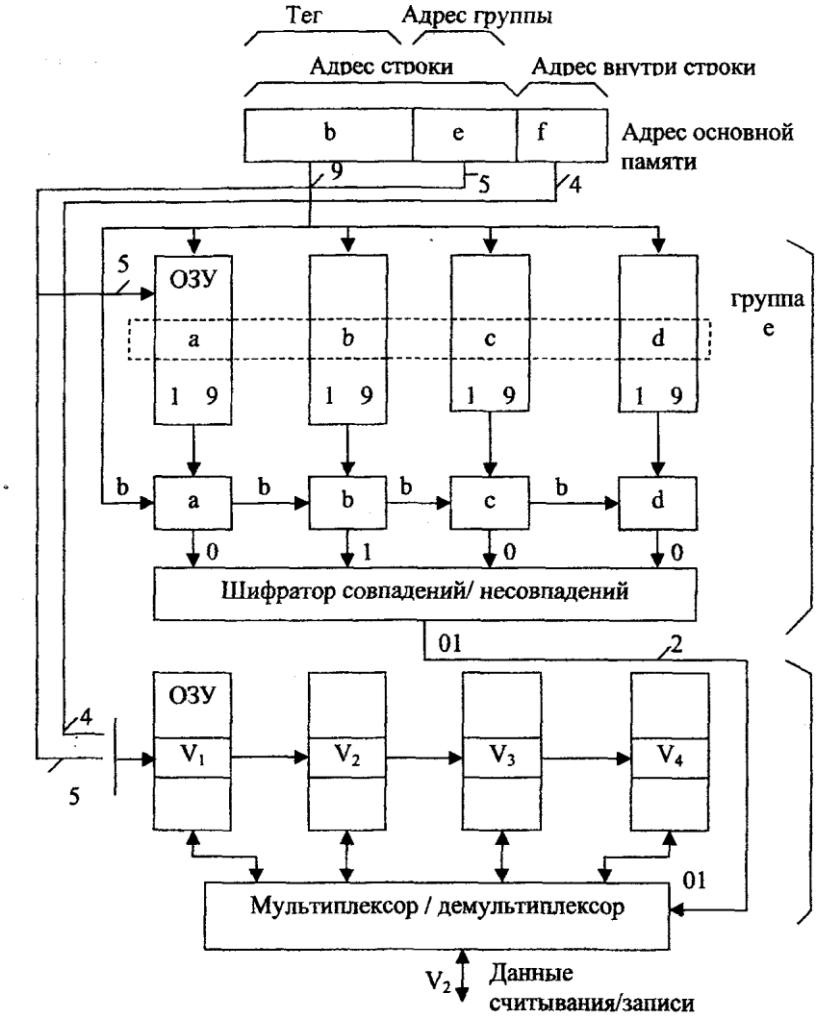


Рис.4.7. Структура кэш-памяти с частично ассоциативным распределением

Адрес строки основной памяти (14 бит) разделяется на две части: b - тег (старшие 9 бит) и e - адрес группы (младшие 5 бит). Адрес строки внутри кэш-памяти, состоящий из 7 бит, разделяется на адрес группы (5 бит) и адрес строки внутри группы (2 бит).

Массивы тегов и данных состоят из четырех банков данных, доступ к каждому из которых осуществляется параллельно одинаковыми адресами. Каждый банк массива тегов имеет длину слова 9 бит для помещения значения тега, а число слов равно числу групп, т.е. 32. Каждый банк массива данных имеет длину слова такую же, как и у основной памяти, а ёмкость его определяется числом слов в одной строке, умноженных на число групп в кэшпамяти.

Для помещения в кэш-память строки, хранимой в ОП по адресу b, необходимо выбрать группу с адресом e. При этом не имеет значения, какая из четырех строк в группе может быть выбрана. Для выбора группы используется метод прямого распределения, а для выбора строки в группе используется метод полностью ассоциативного распределения.

Когда центральный процессор запрашивает доступ по i-му адресу, то осуществляется обращение к массиву тегов по адресу e, выбирается группа

из четырёх тегов (a, b, c, d), каждый из которых сравнивается со старшими 9 битами (b) адреса строки. На выходе четырех схем сравнения формируется унитарный код совпадения (0100), который на шифраторе преобразуется в двухразрядный позиционный код, служащий адресом для выбора банка данных^!).

Одновременно осуществляется обращение к массиву данных по адресу e.f(9 бит) и считывание из банка V;, требуемой строки иди слова.

При пересылке новой строки в кэш-память удаляемая из нее строка выбирается из четырех строк соответствующего набора (группы).

Распределение секторов

По данному методу основная память разбивается на секторы, состоящие из фиксированного числа строк, кэш-память также разбивается на секторы, состоящие из такого же числа строк. Рассмотрим случай, когда длина строки равна 16 словам, а сектор состоит из 16 строк. Структура кэш-памяти с распределением секторов представлена на рис. 4.8. В адресе основной памяти старшие 10 бит показывают номер сектора, следующие 4 бит — номер строки внутри сектора, а младшие 4 бит — адрес слова в строке.

По данному методу распределение секторов в основной памяти, и секторов в кэш-памяти осуществляется полностью ассоциативно. Другими словами, каждый сектор в основной памяти может соответствовать любому сектору в кэш-памяти. Распределение строк в секторе одинаково для основной памяти и кэш-памяти. К каждой строке, хранимой в кэш-памяти, добавляется один бит информации, называемый битом достоверности (действительности); он показывает, совпадает или нет содержимое этой строки с содержимым строки в основной памяти, которая в данный момент анализируется на соответствие строки кэш-памяти.

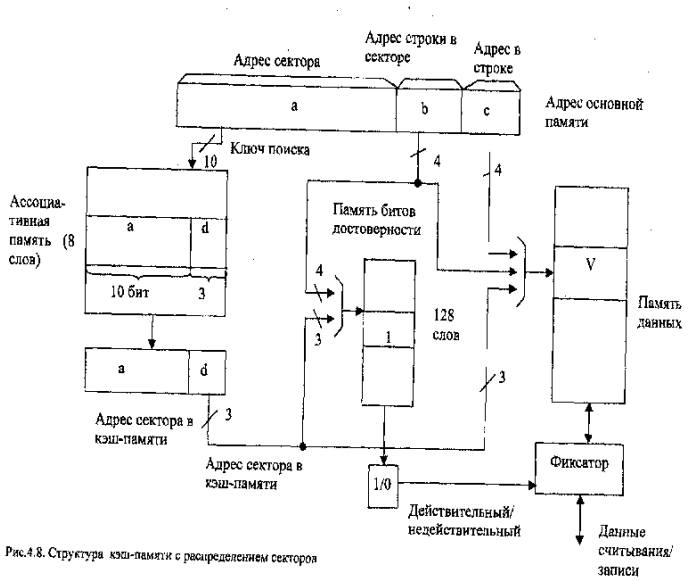


Рис.4.8. Структура кэш-памяти с расщеплением секторов

Если слова, запрашиваемого центральным процессором при доступе, не существует в кэш-памяти, то сначала центральный процессор проверяет, был ли сектор, содержащий это слово, ранее помещен в кэш-память. Если

он отсутствует, то один из секторов кэш-памяти заменяется на этот сектор. Если все сектора кэш-памяти используются, то выбирается один какой-нибудь сектор, и при необходимости только некоторые строки из этого сектора возвращаются в основную память, а этот сектор можно использовать дальше. Когда осуществляется доступ к этому сектору в кэш-памяти и строка, содержащая нужное слово, пересылается из основной памяти, то бит достоверности устанавливается до пересылки строки. Все биты достоверности других строк этого сектора сбрасываются.

Если сектор, содержащий слово, доступ к которому запрашивается, уже находится в кэш-памяти, то в том случае, когда бит достоверности строки, содержащий это слово, равен 0, этот бит устанавливается и строка пересылается из основной памяти в данную область кэш-памяти. В том случае, когда бит достоверности уже равен 1, данное слово можно считать из кэш-памяти.

43.3. Методы обновления строк основной памяти

В табл. 4.1 приведены условия сохранения и обновления информации в ячейках кэш-памяти и основной памяти.

Таблица 4.1

Условия сохранения и обновления информации

Режим рабо	Наличие ко кэш-памяти	Информация	
		В ячейке кэш	В ячейке осн
Чтение	Копия есть К	Не изменя (создается ко	Не изменяетс
Сквозная за	Копия есть К	Обновляется	Обновляется
Обратная за	Копия есть К	Обновляется Обновляется	Не изменяетс

Если процессор намерен получить информацию из некоторой ячейки основной памяти, а копия содержимого этой ячейки уже имеется в кэш-памяти (первая строка табл. 4.1.), то вместо оригинала считывается копия. Информация в кэш-памяти и основной памяти не изменяется. Если копии нет, то производится обращение к основной памяти. Полученная информация пересылается в процессор и попутно запоминается в кэш-памяти. Чтение информации в отсутствии копии отражено во второй строке таблицы. Информация в основной памяти не изменяется.

При записи существует несколько методов обновления старой информации. Эти методы называются **стратегией обновления срок основной памяти.** Если результат обновления строк кэш-памяти не возвращается в основную память, то содержимое основной памяти становится неадекватным вычислительному процессу. Чтобы избежать этого, предусмотрены методы обновления основной памяти, которые можно разделить на две большие группы: метод сквозной записи и метод обратной записи.

Сквозная запись

По методу сквозной записи обычно обновляется слово, хранящееся в основной памяти. Если в кэш-памяти существует копия этого слова, то она

также обновляется. Если же в кэш-памяти отсутствует копия этого слова, то либо из основной памяти в кэш-память пересылается строка, содержащая это слово (метод WTWA — сквозная запись с распределением), либо этого не допускается (метод WTNWA — сквозная запись без распределения). Когда по методу сквозной записи область (строка) в кэш-памяти назначается для хранения другой строки, то в основную память можно не возвращать удаляемый блок, так как копия там есть. Однако в этом случае эффект от использования кэш-памяти отсутствует.

Обратная запись

По методу обратной записи, если адрес объектов, по которым есть запрос обновления, существует в кэш-памяти, то обновляется только кэш-память, а основная память не обновляется. Если адреса объекта обновления нет в кэш-памяти, то в неё из основной памяти пересылается строка, содержащая этот адрес, после чего обновляется только кэш-память. По методу обратной записи в случае замены строк удаляемую строку необходимо также пересылать в основную память. У этого метода существуют две разновидности: метод SWB (простая обратная запись), по которому удаляемая строка возвращается в основную память, и метод FWB (флатовая обратная запись), по которому в основную память записывается только обновлённая строка кэш-памяти. В последнем случае каждая область строки в кэш-памяти снабжается однобитовым флагом, который показывает, было или нет обновление строки, хранящейся в кэш-памяти.

Метод FWB обладает достаточной эффективностью, однако более эффективным считается метод FPWB (флатовая регистровая обратная запись), в котором благодаря размещению буфера между кэш-памятью и основной памятью предотвращается конфликт между удалением и выборкой строк.

Таким образом, теоретически более предпочтительным алгоритмом записи для кэша является метод обратной записи. Кэш с обратной записью будет хранить новую информацию до тех пор, пока у него не появится необходимость избавиться от неё. Тем самым процессор может более оперативно управлять системой. В связи с тем, что кэш со сквозной записью сразу же передаёт вновь записанную информацию в память следующего уровня, кэш со сквозной записью может вызывать дополнительные потери в быстродействии по сравнению с кэшем с обратной записью. В случае кэша с обратной записью допускается выполнение длинных последовательностей быстрых операций записи из процессора, поскольку нет необходимости немедленно направлять эти данные в основную память.

4.3.4. Методы замещения строк кэш-памяти

Способ определения строки, удаляемой из кэш-памяти, называется **стратегией замещения.** Для замещения строк кэш-памяти существует несколько методов: метод замещения наиболее давнего по использованию объекта — строки, метод LRU (замещение наименее используемой информации);

метод FIFO (первым пришёл — первым вышел) и метод произвольного за-

мещения.. В первом случае среди строк, являющихся объектами замещения, выбирается строка, к которой наиболее длительное время не было обращений. По методу FIFO среди всех строк, являющихся объектами замещения, выбирается та, которая самой первой была переслана в кэш-память. И наконец, по последнему методу строка выбирается произвольно. Реализация этих методов упрощается в указанной последовательности, но наибольшим эффектом обладает метод замещения наиболее давнего по использованию объекта (строки).

Для реализации этого метода необходимо манипулировать строками, которые являются объектами замещения, с помощью LRU-стека. При каждой загрузке в этот стек помещается строка, в результате чего при замене используется строка, хранящаяся в наиболее глубокой позиции стека, и эта строка удаляется из стека. При доступе к строке, которая уже содержится в LRU-стеке, эта строка удаляется из стека и заново загружается в него. Стек типа LRU устроен таким образом, что, чем дольше к строке не было доступа, тем в более глубокой позиции она располагается. Реализация стека типа LRU, позволяющего с высокой скоростью выполнять такую операцию, усложняется
по мере увеличения числа строк.

По методу частично ассоциативного распределения число строк в каждом стеке LRU равно числу строк в одной группе, и так как это число мало (порядка 2 - 4), то для каждой группы необходимо использовать свой стек. Если число групп сравнительно велико, то оснащение каждой из них стековым механизмом приводит к повышению стоимости.

4.4. Принципы организации оперативной памяти

4.4.1. Общие положения

Оперативная (основная) память представляет собой следующий уровень иерархии памяти. Оперативная память удовлетворяет запросы кэш-памяти и служит в качестве интерфейса ввода/вывода, поскольку является местом назначения для ввода и источником для вывода. Для оценки производительности (быстродействия) основной памяти используются два основных параметра: задержка и полоса пропускания. Традиционно задержка оперативной памяти имеет отношение к кэш-памяти, а полоса пропускания или пропускная способность относится к вводу/выводу. В связи с ростом популярности кэшпамяти второго уровня и увеличением размеров блоков у такой кэш-памяти полоса пропускания основной памяти становится важной также и для кэшпамяти.

Задержка памяти традиционно оценивается двумя параметрами: временем доступа и длительностью цикла памяти.

Оперативная память современных компьютеров реализуется на микросхемах статических и динамических запоминающих устройств с произвольной выборкой. Микросхемы статических ЗУ (СЗУ) имеют меньшее время доступа и не требуют циклов регенерации (восстановления) информации. Микросхемы динамических ЗУ (ДЗУ) характеризуются большей емкостью и меньшей стоимостью, но требуют схем регенерации и имеют значительно большее время доступа. У статических ЗУ время доступа совпадает с длительностью цикла.

Для микросхем, использующих примерно одну и ту же технологию, емкость ДЗУ по грубым оценкам в 4 — 8 раз превышает емкость СЗУ, но последние имеют в 8 — 16 раз меньшую длительность цикла и большую стоимость. По этим причинам в основной памяти практически любого компьютера, проданного после 1975 года, использовались полупроводниковые микросхемы ДЗУ (для построения кэш-памяти при этом применялись СЗУ). Естественно, были и исключения, например, в оперативной памяти суперкомпьютеров компании Cray Research использовались микросхемы СЗУ.

Для обеспечения сбалансированности системы с ростом скорости процессоров должна линейно расти и емкость ОП. В последнее время емкость микросхем динамической памяти учетверялась каждые три года, увеличиваясь примерно на 60 % в год. К сожалению скорость этих схем за этот же период росла гораздо меньшими темпами (примерно на 7 % в год). В то же время производительность процессоров, начиная с 1987 года, практически увеличивалась на 50 % в год.

Таким образом, согласование производительности современных процессоров со скоростью ОП вычислительных машин и систем остается на сегодняшний день одной из важнейших проблем. Методы повышения производительности за счет увеличения размеров кэш-памяти и введения многоуровневой организации кэш-памяти могут оказаться недостаточно эффективными с точки зрения стоимости систем. Поэтому важным направлением современных разработок являются методы повышения пропускной способности памяти за счет ее организации, включая специальные методы организации ДЗУ.

4.4.2. Методы управления памятью

Оперативная память (ОП) является важнейшим и наиболее дефицитны ресурсом в вычислительных машинах и системах, требующим тщательного эффективного управления. Проблема усложняется при переходе к мульт программным системам, так как в них ОП одновременно используют н сколько вычислительных процессов (программ).

Типы адресов

Для идентификации переменных и команд используются символьш имена (метки), виртуальные адреса и физические адреса (рис. 4.9).

Символьные имена присваивает пользователь при написании программ на алгоритмическом языке или ассемблере.

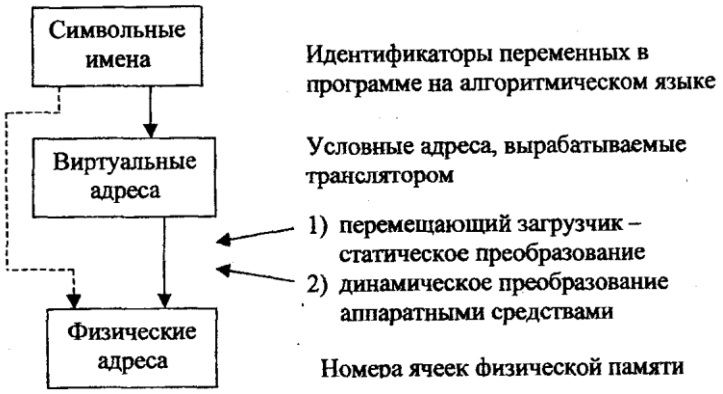


Рис.4.9. Типы адресов

Виртуальные адреса вырабатывает транслятор, переводящий программу на машинный язык. Так как во время трансляции в общем случае неизвестно, в какое место ОП будет загружена программа, то транслятор присваивает переменным и командам виртуальные (условные) адреса, обычно считая по умолчанию, что программа будет размещена, начиная с нулевого адреса. Совокупность виртуальных адресов процесса (программы) называется виртуальным адресным пространством. Каждый процесс имеет собственное виртуальное адресное пространство. Максимальный размер виртуального адресного пространства ограничивается разрядностью адреса, присущей данной архитектуре компьютера и, как правило, не совпадает с объемом физической памяти, имеющимся в компьютере.

Физические адреса соответствуют номерам ячеек ОП, где в действительности расположены или будут расположены переменные и команды. Переход от виртуальных адресов к физическим может осуществляться двумя способами.

В первом случае замену виртуальных адресов на физические делает специальная системная программа — перемещающий загрузчик. Перемещающий загрузчик на основании имеющихся у него исходных данных о начальном адресе физической памяти, в которую предстоит загружать программу, и информации, предоставленной транслятором об адресно-зависимых константах программы, выполняет загрузку программы, совмещая её с заменой виртуальных адресов физическими.

Второй способ заключается в том, что программа загружается в память в неизменном виде в виртуальных адресах, при этом ОС фиксирует смещение действительного расположения программного кода относительно виртуального адресного пространства. Во время выполнения программы при каждом обращении к ОП выполняется преобразование виртуального адреса в физический. Второй способ является более гибким, он допускает перемещение программы во время ее выполнения, в то время как перемещающий загрузчик жестко привязывает программу к первоначально выделенному ей участку. Вместе с тем использование загрузчика уменьшает накладные расходы, так как преобразование каждого виртуального адреса происходит только один раз во время загрузки, а во втором случае — каждый раз при обращении по " данному адресу.

В некоторых случаях (обычно в специализированных системах), когда заранее точно известно, в какой области ОП будет выполняться программа, транслятор выдает исполняемый код сразу в физических адресах.

Все методы управления памятью могут быть разделены на два класса (рис. 4.10):

- методы распределения ОП без использования внешней памяти (дискового пространства);
- методы распределения памяти с использованием дискового пространства;

Рассмотрим вначале первую группу методов.

Рис.4.10. Методы распределения памяти

Распределение памяти фиксированными разделами

Самым простым способом управления оперативной памятью является разделение её на несколько разделов (сегментов) фиксированной величины (статическое распределение). Это может быть выполнено вручную оператором во время старта системы или во время её генерации. Очередная задача, поступающая на выполнение, помещается либо в общую очередь (рис. 4.11,а), либо в очередь к некоторому разделу (рис. 4.11,6). Подсистема управления памятью в этом случае выполняет следующие задачи: сравнивает размер программы, поступившей на выполнение, и свободных разделов памяти; выбирает подходящий раздел; осуществляет загрузку программы и настройку адресов.

При очевидном преимуществе, заключающемся в простоте реализации, данный метод имеет существенный недостаток — жесткость. Так как в каждом разделе может выполняться только одна программа, то уровень мультипрограммирования заранее ограничен числом разделов независимо от того, какой размер имеют программы.

Даже если программа имеет небольшой объем, она будет занимать весь раздел, что приводит к неэффективному использованию памяти. С другой стороны, даже если объем оперативной памяти машины позволяет выполнить некоторую программу, разбиение памяти на разделы не позволяет сделать этого.

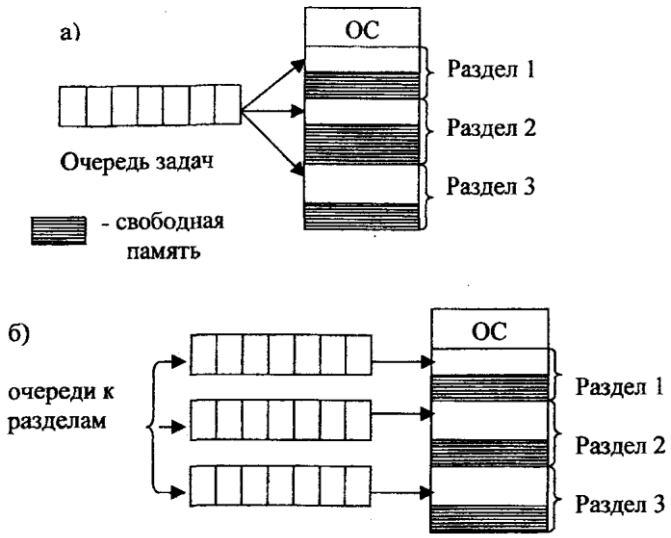


Рис. 4.11. Распределение памяти фиксированными разделами: а) с общей очередью; б) с отдельными очередями

Распределение памяти разделами переменной величины

В этом случае память машины не делится заранее на разделы. Сначала вся память свободна. Каждой вновь поступающей задаче выделяется необходимая ей память. Если достаточный объем памяти отсутствует, то задача не принимается на выполнение и стоит в очереди. После завершения задачи память освобождается, и на это место может быть загружена другая задача. Таким образом, в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера. На рис. 4.12 показано состояние памяти в различные моменты времени при использовании динамического распределения. Так в момент tg в памяти находится только ОС, а к моменту t, память разделена между 5 задачами, причем задача П4, завершая работу, покидает память к моменту t;. На освободившееся место загружается задача П6, поступившая в момент 1з.

Задачами операционной системы при реализации данного метода управления памятью являются: ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти; анализ запроса (при поступлении новой задачи), просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения поступившей задачи; загрузка задачи в выделенный ей раздел и корректировка таблиц свободных и занятых областей; корректировка таблиц свободных и занятых областей (после завершения задачи).

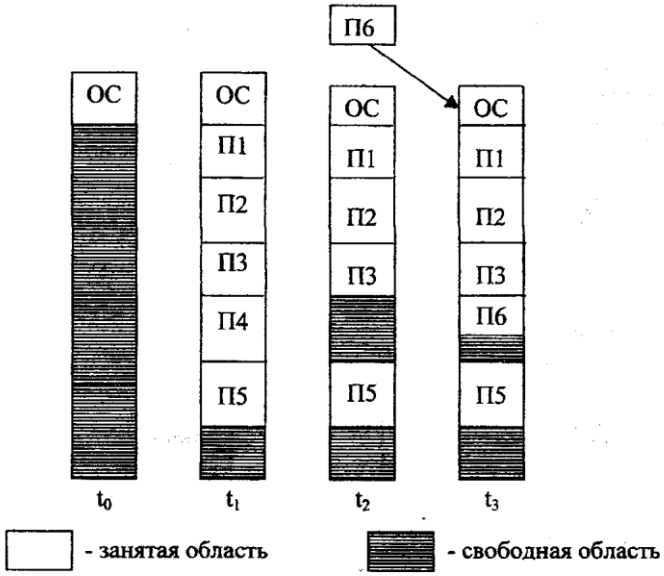


Рис. 4.12. Распределение памяти динамическими разделами

Программный код не перемещается во время выполнения, т.е. может быть проведена единовременная настройка адресов посредством использования перемещающего загрузчика.

Выбор раздела для вновь поступившей задачи может осуществляться по разным правилам: «первый попавшийся раздел достаточного размера»; «раздел, имеющий наименьший достаточный размер»; «раздел, имеющий наибольший достаточный размер». Все эти правила имеют свои преимущества и недостатки.

По сравнению с методом распределения памяти фиксированными разделами данный метод обладает гораздо большей гибкостью, но ему присущ очень серьезный недостаток — **фрагментация памяти.** Фрагментация — это наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов). Настолько маленького, что ни одна из вновь поступающих программ не может поместиться ни в одном из участков, хотя суммарный объем фрагментов может составить значительную величину, намного превышающую требуемый объем памяти[1].

Перемещаемые разделы

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших либо в сторону младших адресов так, чтобы вся свободная память образовывала единую свободную область (рис. 4.13). В дополнение к функциям, которые выполняет ОС при распределении памяти переменными разделами, в данном случае она должна еще время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей.

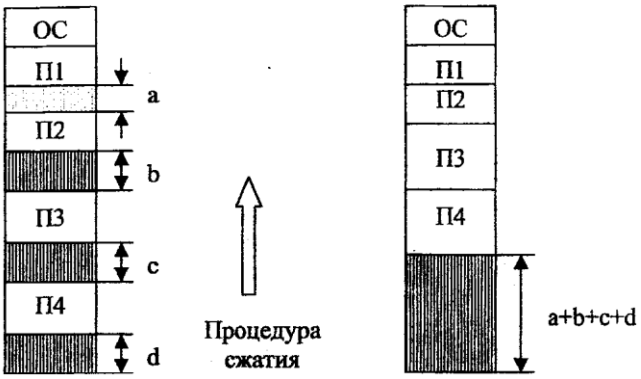


Рис.4.13. Распределение памяти перемещаемыми разделами

Эта процедура называется «сжатием». Сжатие может выполняться либо при каждом завершении задачи, либо только тогда, когда для вновь поступившей задачи нет свободного раздела достаточного размера. В первом случае требуется меньше вычислительной работы при корректировке таблиц, а во втором — реже выполняется процедура сжатия. Так как программы перемещаются по оперативной памяти в ходе своего выполнения, то преобразование адресов из виртуальной формы в физическую должно выполняться динамическим способом.

Хотя процедура сжатия и приводит к более эффективному использованию памяти, она может потребовать значительного времени, что часто перевешивает преимущества данного метода.

4.43. Организация виртуальной памяти

Уже достаточно давно пользователи столкнулись с проблемой размещения в основной памяти программ, размер которых превышал имеющуюся в наличии свободную память. Решением этой проблемы было использование внешней памяти (дискового пространства). Программы разбивались на части (оверлеи), которые хранились в ОП и на диске. Перемещение их между основной памятью и диском осуществлялось средствами ОС. Однако разбиением программы на части должен был заниматься программист. Это приводило к увеличению трудоемкости программирования и к неэффективному использованию памяти.

Развитие методов организации вычислительного процесса в этом направлении привело к появлению метода, известного под названием **виртуальная память.** Виртуальным называется такой ресурс, который для пользователя (пользовательской программы) представляется обладающим свойствами, которыми он в действительности не обладает. Так, например, пользователю может быть предоставлена виртуальная оперативная память, размер которой превосходит всю имеющуюся в системе реальную ОП. Пользователь пишет программы так, как будто в его распоряжении имеется однородная (одноуровневая) оперативная память большого объема, но в действительности все данные, используемые программой, хранятся на нескольких разнородных запоминающих устройствах, обычно в ОП и на дисках, и при необходимости частями перемещаются между ними.

Таким образом, **виртуальная память** — это совокупность

программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся ОП. Для этого виртуальная память (ВП)
решает следующие задачи:

- размещает данные в запоминающих устройствах разного типа, например, часть программы в ОП, а часть на диске;
- перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы
с диска в ОП;
- преобразует виртуальные адреса в физические.

Все эти действия выполняются автоматически, без участия программиста, т.е. механизм ВП является прозрачным по отношению к пользователю.

Наиболее распространенными реализациями виртуальной памяти являются страничное, сегментное и странично-сегментное распределение памяти (рис. 4.10), атаюке свопинг.

Страничное распределение

На рис. 4.14 показана схема страничного распределения памяти. Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые **виртуальными страницами-** В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Вся оперативная память машины также делится на части такого же размера, называемые **физическими страницами** (или блоками).

Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов.

Рис.4.14. Страничное распределение памяти

При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные — на диск. Смежные виртуальные страницы необязательно располагаются в смежных физических страницах. При загрузке операционная система создает для каждого процесса информационную структуру — таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в оперативную память, или делается отметка о том, что виртуальная страница выгружена на диск (ВЗУ). Кроме того, в таблице страниц содержится управляющая информация, такая как признак модификации страницы, признак невыгружаемое™ (выгрузка некоторых страниц может быть запрещена), признак обращения к странице (используется для подсчета числа обращений за определенный период времени) и другие данные, формируемые и используемые механизмом виртуальной памяти.

При активизации очередного процесса в специальный регистр процессора загружается адрес таблицы страниц данного процесса.

При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение. Если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический. Если же нужная виртуальная страница в данный момент выгружена на диск, то происходит так называемое страничное прерывание. Выполняющийся процесс переводится в состояние ожидания и активизируется другой процесс из очереди готовых. Параллельно программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в оперативную память. Если в памяти имеется свободная физическая

страница, то загрузка выполняется немедленно, если же свободных страниц нет, то решается вопрос, какую страницу следует выгрузить из оперативной памяти.

В данной ситуации может быть использовано много разных критериев выбора, наиболее популярные из них следующие:

- дольше всего не использовавшаяся страница;
- первая попавшаяся страница;
- страница, к которой в последнее время было меньше всего обращений.

В некоторых системах используется понятие рабочего множества страниц. Рабочее множество определяется для каждого процесса и представляет собой перечень наиболее часто используемых страниц, которые должны постоянно находиться в оперативной памяти и поэтому не подлежат выгрузке.

После того, как выбрана страница, которая должна покинуть оперативную память, анализируется ее - признак модификации (из таблицы страниц). Если выталкиваемая страница с момента загрузки была модифицирована, то ее новая версия должна быть переписана на диск. Если нет, то она может быть просто уничтожена, т.е. соответствующая физическая страница объявляется свободной.

Рассмотрим механизм преобразования виртуального адреса в физический при страничной организации памяти (рис. 4.15).

Виртуальный адрес при страничном распределении может быть представлен в виде пары (p, s), где p — номер виртуальной страницы процесса (нумерация страниц начинается с 0), s — смещение в пределах виртуальной страницы. Учитывая, что размер страницы равен 2 в степени k, смещение s может быть получено простым отделением k младших разрядов в двоичной записи виртуального адреса. Оставшиеся старшие разряды представляют собой двоичную запись номера страницы p.

При каждом обращении к оперативной памяти аппаратными средствами

выполняются следующие действия:

1. На основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице.
2. Из записи извлекается номер физической страницы.
3. К номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса.

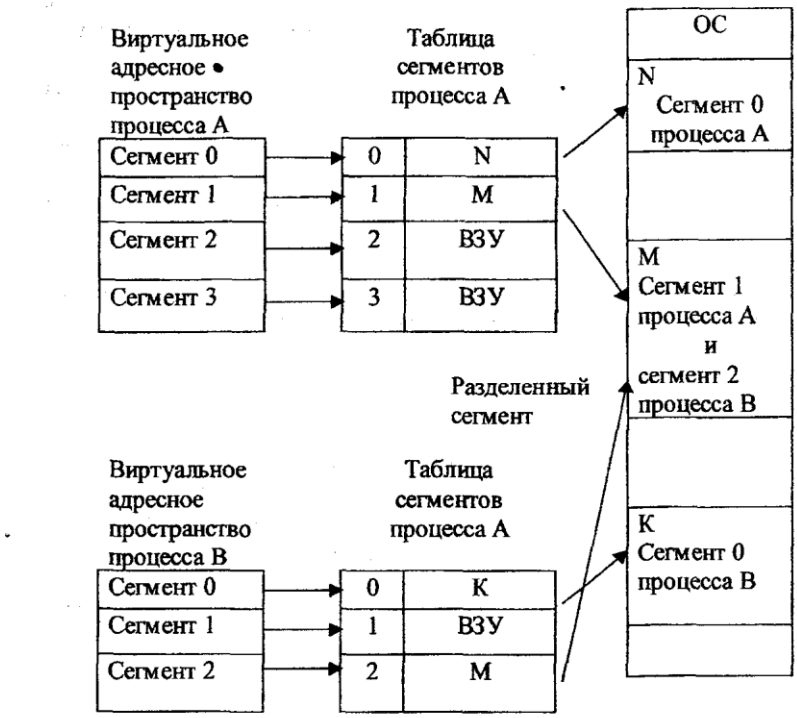


Рис.4.16. Распределение памяти сегментами

Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются, при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g — номер сегмента, as — смещение в сегменте. Физический адрес получается путем сложения начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

Странично-сегментное распределение

Как видно из названия, данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов. Виртуальное пространство процесса делится на сегменты, а каждый сегмент в свою очередь делится на виртуальные страницы, которые нумеруются в пределах сегмента. Оперативная память делится на физические страницы. Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть на диске. Для каждого сегмента создается

своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении.

Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс. На рис. 4.17 показана схема преобразования виртуального адреса в физический для данного метода.

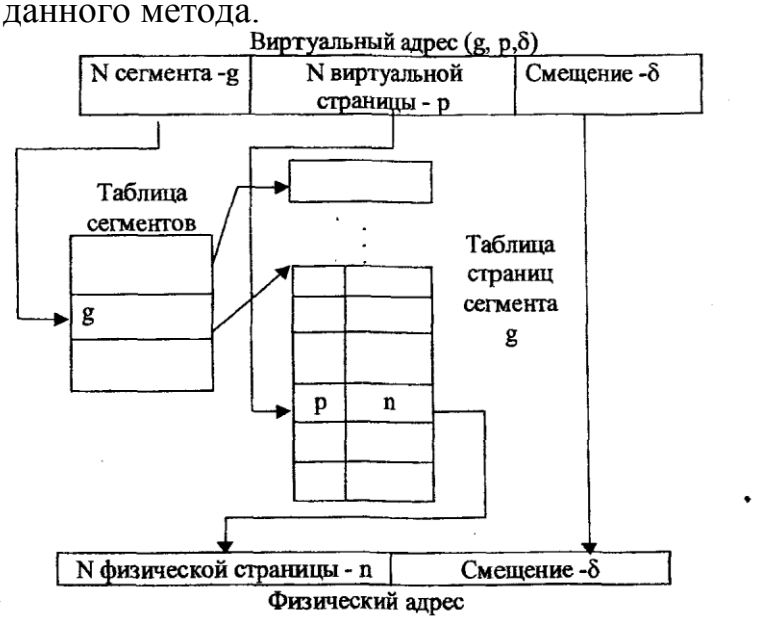


Рис.4.17. Схема преобразования виртуального адреса в физический для сегментно-страничной организации памяти

Процесс преобразования адресов посредством таблиц является достаточно длительным и, естественно, приводит к снижению производительности

системы. С целью ускорения этого процесса преобразование может осуществляться специальными аппаратными средствами, в основе которых лежит использование принципа ассоциативной памяти. Реализуемый при этом механизм получил название механизма динамического преобразования адресов (рис. 4.18).

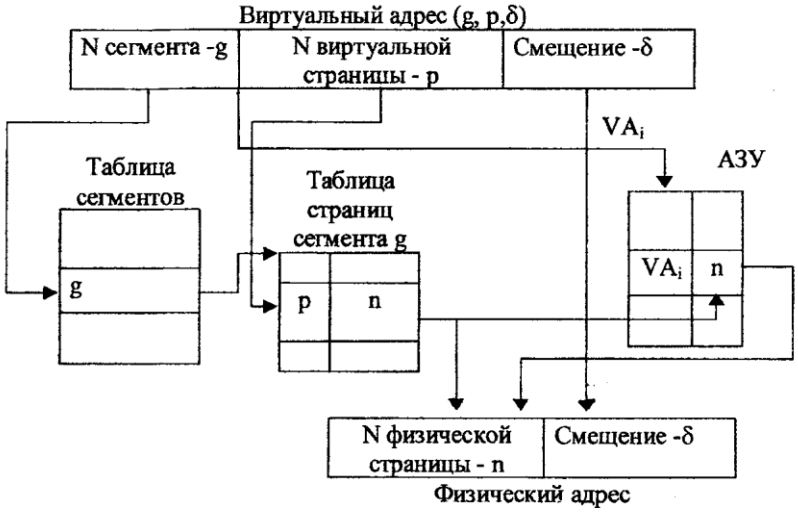


Рис.4.18. Механизм динамического преобразования адресов

Виртуальный адрес страницы VA, составленный из полей g и p, передается в ассоциативную память (буфер быстрой переадресации) в качестве поискового признака — первое поле ассоциативного ЗУ (АЗУ). Вторым полем АЗУ является физический адрес страницы в ОП. При обнаружении совпадения VAi с содержимым памяти из соответствующей ячейки АЗУ выбирается физический адрес страницы n, позволяющий сформировать полный физический адрес элемента данных, находящегося в ОП. Если совпадение не произошло, то трансляция адресов осуществляется обычными методами через таблицы сегментов и страниц. Эффективность механизма динамического преобразования адресов зависит от коэффициента «попадания», т.е. от того, насколько редко приходится обращаться к табличным методам трансляции адресов. Учитывая принцип локальности программ и данных, можно сказать, что при первом обращении к странице, расположенной в ОП, физический адрес определяется с помощью таблиц и загружается в соответствующую ячейку АЗУ. Последующие обращения к странице выполняются с использованием АЗУ.

Свопинг

Разновидностью виртуальной памяти является свопинг. На рис. 4.19 показан график зависимости коэффициента загрузки процессора в зависимости от числа одновременно выполняемых процессов и доли времени, проводимого этими процессами в состоянии ожидания ввода-вывода.

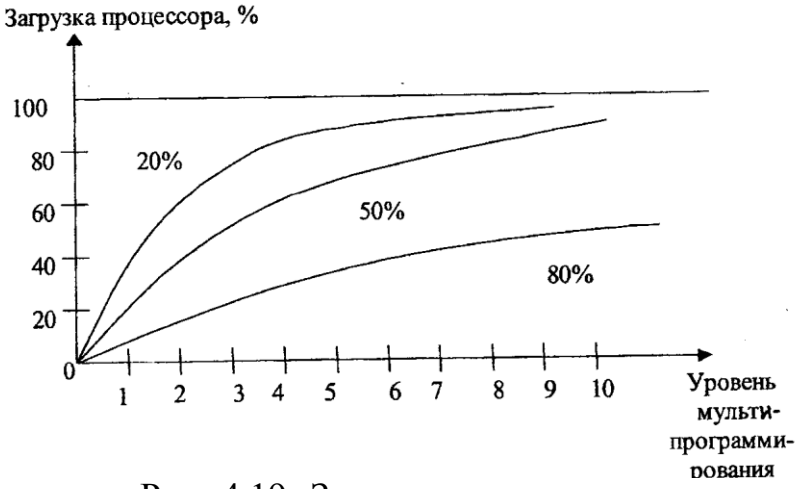


Рис. 4.19. Зависимость загрузки процессора от числа задач и интенсивности ввода-вывода

Из рисунка видно, что для загрузки процессора на 90 % достаточно всего трех счетных задач. Однако для того, чтобы обеспечить такую же загрузку интерактивными задачами, выполняющими интенсивный ввод-вывод, потребуются десятки таких задач. Необходимым условием для выполнения задачи является загрузка её в оперативную память, объем которой ограничен. В этих условиях был предложен метод организации вычислительного процесса, называемый **свопингом.** В соответствии с этим методом некоторые процессы (обычно находящиеся в состоянии ожидания) временно выгружаются на диск. Планировщик операционной системы не исключает их из своего рассмотрения, и при наступлении условий активизации некоторого процесса, находящегося в области свопинга на диске, этот процесс перемещается в оперативную память. Если свободного места в оперативной памяти не хватает, то выгружается другой процесс.

При свопинге, в отличие от рассмотренных ранее методов реализации виртуальной памяти, процесс перемещается между памятью и диском целиком, т.е. в течение некоторого времени процесс может полностью отсутствовать в оперативной памяти. Существуют различные алгоритмы выбора процессов на загрузку и выгрузку, а также различные способы выделения оперативной и дисковой памяти загружаемому процессу.

4.4.4. Методы повышения пропускной способности оперативной памяти

Основными методами увеличения полосы пропускания памяти являются: увеличение разрядности или «ширины» памяти, использование расслоения памяти, использование независимых банков памяти, обеспечение режима бесконфликтного обращения к банкам памяти, использование специальных режимов работы динамических микросхем памяти.

Выборка широким словом

Прямой способ сокращения числа обращений к ОП состоит в организации **выборки широким словом.** Этот способ основывается на свойстве локальности данных и программ. При выборке широким словом за одно обращение к ОП производится одновременная запись или считывание

нескольких команд или слов данных из «широкой» ячейки. Широкое слово заносится в буферную память (кэш-память) или регистр, где оно расформировывается на отдельные команды или слова данных, которые могут последовательно использоваться процессором без дополнительных обращений к ОП.

В системах с кэш-памятью первого уровня ширина шин данных ОП часто соответствует ширине шин данных кэш-памяти, которая во многих случаях имеет физическую ширину шин данных, соответствующую количеству разрядов в слове. Удвоение и учетверение ширины шин кэш-памяти и ОП удваивает или учетверяет соответственно полосу пропускания системы памяти.

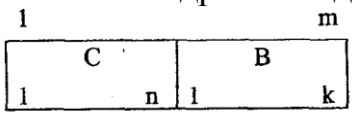
Реализация выборки широким словом вызывает необходимость мультиплексирования данных между кэш-памятью и процессором, поскольку основной единицей обработки данных в процессоре все еще остается слово. Эти мультиплексоры оказываются на критическом пути поступления информации в процессор. Кэш-память второго уровня несколько смягчает эту проблему, т.к. в этом случае мультиплексоры могут располагаться между двумя уровнями кэш-памяти, т.е. вносимая ими задержка не столь критична. Другая проблема, связанная с увеличением разрядности памяти, заключается в необходимости определения минимального объема (инкремента) памяти для поэтапного её расширения, которое часто выполняется самими пользователями во время эксплуатации системы. Удвоение или учетверение ширины памяти приводит к удвоению или учетверению этого минимального инкремента. Кроме того, имеются проблемы и с организацией коррекции ошибок в системах с широкой памятью.

Примером организации широкой ОП является система Alpha AXP21064, в которой кэш второго уровня, шина памяти и сама ОП имеют разрядность 256 бит.

Расслоение обращений

Другой способ повышения пропускной способности ОП связан с построением памяти, состоящей на физическом уровне из нескольких модулей (банков) с автономными схемами адресации, записи и чтения. При этом на логическом уровне управления памятью организуются последовательные обращения к различным физическим модулям. Обращения к различным модулям могут перекрываться, и таким образом образуется своеобразный конвейер. Эта процедура носит название **расслоения памяти.** Целью данного метода является увеличение скорости доступа к памяти посредством совмещения фаз обращений ко многим модулям памяти. Известно несколько вариантов организации расслоения. Наиболее часто используется способ расслоения обращений за счет **расслоения адресов.** Этот способ основывается на свойстве локальности программ и данных, предполагающем, что адрес следующей команды программы на единицу больше адреса предыдущей (линейность программ нарушается только командами перехода). Аналогичная последовательность адресов генерируется процессором при чтении и записи слов данных. Таким образом, типичным случаем распределения адресов обращений к памяти является последовательность вида a, a+ 1, a + 2,... Из

этого следует, что расслоение обращений возможно, если ячейки с адресами а, а+ 1, а + 2,... будут размещаться в блоках 0, 1,2,... Такое распределение ячеек по модулям (банкам) обеспечивается за счет использования адресов вида



,

где В - к-разрядный адрес модуля (младшая часть адреса) и С - п-разрядный адрес ячейки в модуле В (старшая часть адреса).

Принцип расслоения обращений иллюстрируется на рис. 4.20,а. Все программы и данные «размещаются» в адресном пространстве последовательно. Однако ячейки памяти, имеющие смежные адреса, находятся в различных физических модулях памяти. Если ОП состоит из 4-х модулей, то номер модуля кодируется двумя младшими разрядами адреса. При этом полные т-разрядные адреса 0, 4, 8,... будут относиться к блоку 0, адреса 1, 5, 9, ... - к блоку 1, адреса 2, 6, 10,... — к блоку 2 и адреса 3, 7, 11,... - к блоку 3. В результате этого последовательность обращений к адресам 0, 1, 2, 3, 4, 5, ... будет расслоена между модулями 0,1,2,3,0,1,....

Поскольку каждый физический модуль памяти имеет собственные схемы управления выборкой, можно обращение к следующему модулю производить, не дожидаясь ответа от предыдущего. Так на временной диаграмме (рис. 4.20,6) показано, что время доступа к каждому модулю составляет т = 4Т, где Т = t_{i+1}-t_i - длительность такта. В каждом такте следуют непрерывно обращения к модулям памяти в моменты времени t_1, t_2, t_3 ... .

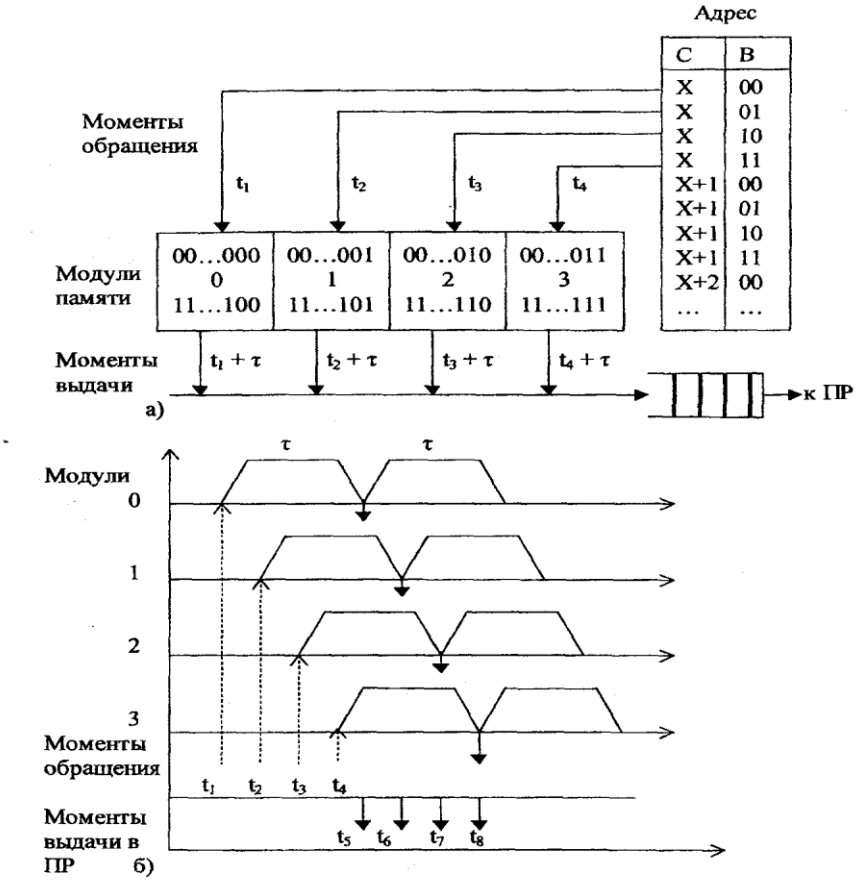


Рис. 4.20. Расслоение памяти: а) организация адресного пространства; б) временная диаграмма работы модулей

При наличии четырех модулей темп выдачи квантов информации из памяти в процессор будет соответствовать одному такту Т, при этом скорость выдачи информации из каждого модуля в четыре раза ниже.

Задержка в выдаче кванта информации относительно момента обращения также составляет 4Т, однако задержка в выдаче каждого последующего кванта относительно момента выдачи предыдущего составит Т.

При реализации расслоения по адресам число модулей памяти может быть произвольным и необязательно кратным степени 2. В некоторых компьютерах допускается произвольное отключение модулей памяти, что позволяет исключать из конфигурации неисправные модули.

В современных высокопроизводительных компьютерах число модулей обычно составляет 4 -16, но иногда превышает 64.

Так как схема расслоения по адресам базируется на допущении о локальности, она дает эффект в тех случаях, когда это допущение справедливо, т.е. при решении одной задачи.

Для повышения производительности мультипроцессорных систем, работающих в многозадачных режимах, реализуют другие схемы, при которых **различные процессоры обращаются к различным модулям** памяти. Необходимо помнить, что процессоры ввода-вывода также занимают циклы памяти и вследствие этого могут сильно влиять на

производительность системы. Для уменьшения этого влияния обращения центрального процессора и процессоров ввода-вывода можно организовать к разным модулям памяти.

Обобщением идеи расслоения памяти является возможность реализации нескольких независимых обращений, когда несколько контроллеров памяти позволяют модулям памяти (или группам расслоенных модулей памяти) работать независимо.

Если система памяти разработана для поддержки множества независимых запросов (как это имеет место при работе с кэш-памятью, при реализации многопроцессорной и векторной обработки), эффективность системы будет в значительной степени зависеть от частоты поступления независимых запросов к разным модулям. Обращения по последовательным адресам, или в более общем случае обращения по адресам, отличающимся на нечетное число, хорошо обрабатываются традиционными схемами расслоения памяти. Проблемы возникают, если разница в адресах последовательных обращений четная. Одно из решений, используемое в больших компьютерах, заключается в том, чтобы статистически уменьшить вероятность подобных обращений путем значительного увеличения количества модулей памяти. Например, в суперкомпьютере NEC SX/3 используются 128 модулей памяти.

Прямое уменьшение числа конфликтов за счет организации чередующихся обращений к различным модулям памяти может быть достигнуто путем **размещения программ и данных в разных модулях.** Доля команд в программе, требующих ссылок к находящимся в ОП данным, зависит от класса решаемой задачи и от архитектурных особенностей компьютера. Для большинства ЭВМ с традиционной архитектурой и задач научно-технического характера эта доля превышает 50 %. Поскольку обращения к командам и элементам данных чередуются, то разделение памяти на память команд и память данных повышает быстродействие машины подобно рассмотренному выше механизму расслоения. Разделение памяти на память команд и память данных широко используется в системах управления или обработки сигналов. В подобного рода системах в качестве памяти команд нередко используются постоянные запоминающие устройства (ПЗУ), цикл которых меньше цикла устройств, допускающих запись, это делает разделение программ и данных весьма эффективным. Следует отметить, что обращения процессоров ввода-вывода в режиме прямого доступа в память логически реализуются как обращения к памяти данных.

Выбор той или иной схемы расслоения для компьютера (системы) определяется целями (достижение высокой производительности при решении множества задач или высокого быстродействия при решении одной задачи), архитектурными и структурными особенностями системы, а также элементной базой (соотношением длительностей циклов памяти и узлов обработки). Могут использоваться комбинированные схемы расслоения.

4.4.5. Методы защиты памяти

Подсистема защиты памяти представляет собой комплекс аппаратно-программных средств, обеспечивающих предотвращение взаимного

искажения одновременно находящихся в ОП программ и несанкционированного доступа к любой хранящейся в ОП информации. В общем случае защита осуществляется как при записи для предотвращения искажения информации, не относящейся к выполняемой в данный момент программе, так и при считывании для исключения возможности использования информации, не принадлежащей данному пользователю, т.е. для предотвращения несанкционированного доступа к информации.

Независимо от принятых принципов построения подсистемы защиты памяти в основе её функционирования заложена проверка всех адресов, поступающих для обращения к ОП. В результате такой проверки формируется сигналы управления, разрешающий обращение к ОП, если адрес относится к выделенной для данной программы области памяти, в противном случае вырабатывается сигнал, запрещающий выполнение данной команды (при этом посылается запрос на прерывание реализуемой программы с целью установления причины нарушения границ разрешенной для использования области памяти).

Реализация идеи защиты памяти в любом случае не должна сопровождаться заметным снижением производительности машины и не требовать больших аппаратных затрат.

Различают три способа защиты памяти: по граничным адресам, по маскам и по уровням привилегий (ключам).

Защита памяти по граничным адресам

Защита памяти по граничным адресам осуществляется с помощью регистров и узлов сравнения кодов, размещаемых в блоке защиты памяти (БЗП). Реализация этого способа защиты предусматривает выделение для каждой программы определенной области ОП, составленной из ячеек с последовательными адресами. Границы области отмечаются фиксированием адресов её начальной и конечной ячеек. Граничные адреса вводятся в регистры БЗП управляющей программой операционной системы перед началом выполнения каждой рабочей программы. При выполнении данной рабочей программы каждый поступающий в ОП исполнительный адрес с помощью узлов сравнения кодов сравнивается с граничными адресами. По результатам сравнения устанавливается возможность обращения к ОП по поступившему адресу: если он находится в пределах граничных адресов, то разрешается доступ к соответствующей ячейке памяти, в противном случае вырабатывается сигнал запроса на прерывание выполняемой программы.

Преимущество данного способа защиты памяти состоит в том, что он позволяет защищать области памяти произвольной длины. Кроме того, блок защиты достаточно прост, а его функционирование не приводит к значительным временным затратам. Однако необходимость размещения программ в областях памяти с последовательными номерами ячеек и ограниченных двумя граничными адресами существенно снижает возможности программирования и даже эффективность работы ЭВМ. Поэтому способ защиты памяти по граничным адресам в настоящее время применяется редко, при статическом распределении памяти, когда для каждой из параллельно выполняемых рабочих программ заранее (до начала их выполнения) отводится определенная область памяти.

Защита памяти по маскам

Защита памяти по маскам используется при страничной организации ОП. Для каждой программы перед её выполнением указываются номера страниц, отведенные для размещения её команд и всех необходимых данных. Указание о номерах отведенных страниц для данной программы задается управляющей программой операционной системы в виде кода маски или кода признаков страниц. Код маски формируется для каждой рабочей программы. Под маской программы понимается n-разрядный двоичный код, разрядность которого определяется количеством страниц ОП. Каждый i-й разряд маски указывает о принадлежности i-й страницы ОП данной программе: если в i-м разряде задано значение 1, то при обращении к ОП разрешен доступ к любой ячейке i-ой страницы, если же i-й разряд маски содержит ноль, то выполняемой программе доступ к i-й странице запрещен.

Перед выполнением программы её код маски по специальной команде засылается в регистр маски РМ (рис. 4.21) блока защиты. При каждом обращении к ОП код номера страницы из исполнительного адреса загружается в регистр номера страницы РС и затем расшифровывается дешифратором номера страницы ДШС. На одном из выходов этого дешифратора, номер которого равен номеру страницы, появляется единичный сигнал. Если в соответствующем этой странице двоичном разряде кода маски программы задана единица, то схема сравнения выдает сигнал разрешения передачи адреса ячейки в ОП, в противном случае схема сравнения вырабатывает сигнал прерывания программы.

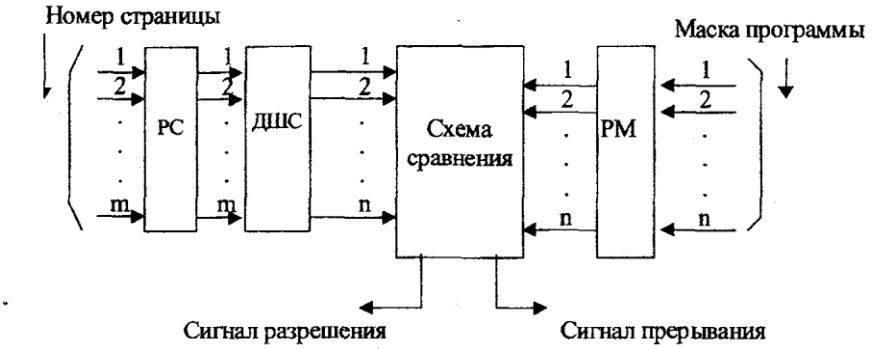


Рис.4.21. Защита памяти по маскам

По сравнению с защитой по граничным адресам защита памяти по маскам отличается большей гибкостью при организации распределения ОП. Для своей реализации данный метод не требует сложного оборудования.

Однако при большой емкости ОП, состоящей из значительного количества страниц, она становится неэффективной. Это связано с многоразрядно-стью кода маски, возрастанием сложности дешифратора номера страниц и всего БЗП, а также заметным увеличением времени работы БЗП по формированию управляющих сигналов.

Защита памяти по ключам

Защита памяти по ключам (уровням привилегий) используется в большинстве современных многопрограммных ЭВМ со страничной организацией памяти и динамическим её распределением между параллельно выполняемыми программами. В её основе лежит применение специальных кодов (уровней) для проверки соответствия используемых

массивов ячеек памяти номеру выполняемой программы.

Каждой рабочей программе ОС придает специальный ключ — **ключ программы.** Все выделенные для данной рабочей программы страницы отмечаются одним и тем же ключом страницы или **ключом защиты.** В качестве ключа защиты обычно указывается двоичный код номера программы. В процессе обращения к ОП производится сравнение ключа выполняемой про граммы с ключами защиты соответствующих страниц памяти. Обращение разрешается только при совпадении сравниваемых кодов ключей. Защита памяти по ключам применяется не только при работе ОП с процессором, но и в ходе обмена информацией с ВЗУ через каналы ввода-вывода. Тогда вместо ключей программ используются **ключи каналов.** Разрядность кодов ключей определяется максимальным количеством параллельно выполняемых программ.

Структура БЗП по ключам приведена на рис. 4.22. Его основу составляет память ключей защиты ПКЗ адресного типа. Емкость ПКЗ строго соответствует количеству страниц. Разрядность ячеек ПКЗ равна разрядности кодов ключей (k) с добавлением одного или нескольких разрядов для задания режима защиты (j). Ввод кодов защиты в ПКЗ осуществляется под управлением ОС при каждом распределении поля ОП между параллельно выполняемыми программами и каналами ввода-вывода, а также при любом перераспределении поля ОП. Выборка информации **из** ПКЗ производится по номерам страниц, представляемых старшими разрядами кода адреса ячейки ОП, по которому идет обращение к ОП.

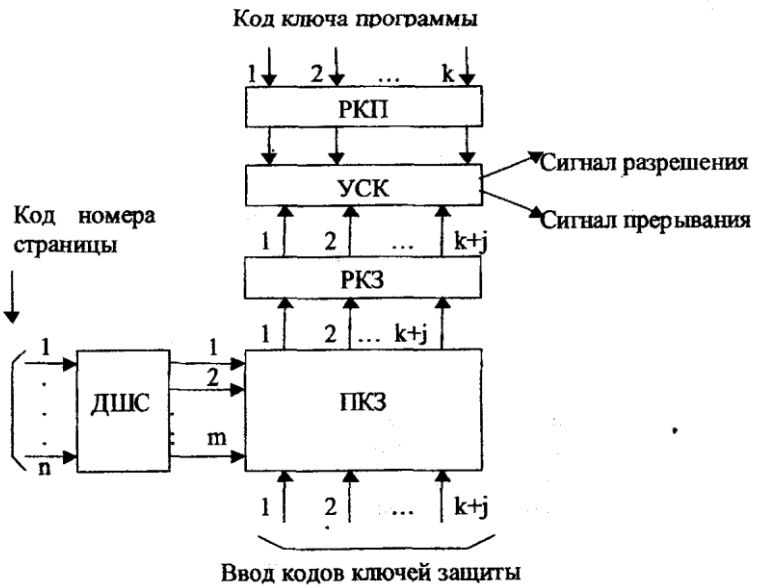


Рис.4.22. Защита памяти по ключам

Кроме ПКЗ в состав блока защиты памяти входят: дешифратор номера страниц ДШС для расшифровки кодов номеров страниц; регистр ключей защиты РКЗ для фиксации выбираемых из ПКЗ кодов; регистр ключей программ РКП для приема и хранения кодов ключей программ, поступающих из регистра слова состояния программы (системного регистра) процессора, или кодов ключей каналов; узел сравнения ключей УСК для сравнения ключей защиты с ключами программ (или каналов).

Код ключа некоторой программы вводится в РКП операционной систе-

мой при каждой инициализации этой программы, т.е. при каждом переходе к выполнению её команд. При работе с каналом в РКП вводится ключ канала. В УСК, представляющем собой комбинационную схему, производится сравнение ключа защиты, выбранного из ПКЗ при данном обращении к ОП и зафиксированного в РКЗ, и кода ключа программы (ключа канала), поступающего от РКП. По результатам сравнения узел сравнения кодов формирует либо сигнал разрешения обращения к ОП, либо сигнал прерывания выполняемой программы.

Функционирование БЗП начинается с ввода в ДШС кода номера страницы ОП, к которой производится обращение. По номеру страницы из ПКЗ выбирается соответствующий ключ защиты, код которого помещается в РКЗ. В УСК код ключа защиты сравнивается с кодом ключа программы (канала) и формируются управляющие сигналы разрешения или прерывания.

Защита памяти по ключам (уровням привилегий) является наиболее универсальной и гибкой, особенно эффективной при страничной или сегментно-страничной организации виртуальной памяти и динамическом её распределении. Однако такой способ защиты требует для своей технической реализации заметных дополнительных аппаратных затрат, прежде всего необходима ПКЗ с очень небольшим временем выборки кода ключей. Память ключей защиты строится так, чтобы время выборки кода было практически на порядок меньше времени выборки из ОП.

4.4.6. Методы ускорения процессов обмена между ОП и ВЗУ

Эффективная скорость обмена между оперативным и внешним уровнями памяти в значительной степени определяется затратами на поиск секторов или блоков в накопителе ВЗУ. Для уменьшения влияния затрат времени поиска информации на скорость обмена используют традиционные методы **буферизации и распараллеливания.** Метод буферизации заключается в использовании так называемой дисковой кэш-памяти. **Дисковый кэш** уменьшает среднее время обращения к диску. Это достигается за счет того, что копии данных, находящихся в дисковой памяти, заносятся в полупроводниковую память. Когда необходимые данные оказываются находящимися в кэше, время обращения значительно сокращается. За счет исключения задержек, связанных с позиционированием головок, время обращения может быть уменьшено в 2 -10 раз.

Дисковый кэш может быть реализован программно или аппаратно. Программный дисковый кэш — это буферная область в ОП, предназначенная для хранения считываемой с диска информации. При поступлении запроса на считывание информации с диска вначале производится поиск запрашиваемой информации в программном кэше.

При наличии в кэше требуемой информации, она передается в процессор. Если она отсутствует, то осуществляется поиск информации на диске. Считанный с диска информационный блок заносится в буферную область ОП (программный дисковый кэш). Программа, управляющая дисковой кэшпамятью, осуществляет также слежение и за работой диска. Весьма хорошую производительность показывают программы Smart Drv, Ncache и Super PC-Kwik. Иногда для программного кэша используется

дополнительная или расширенная память компьютера.

Аппаратный дисковый кэш — это встроенный в контроллер диска кэш-буфер с ассоциативным принципом адресации информационных блоков. По запросу на считывание информации вначале производится поиск запрашиваемого блока в кэше. Если блок находится в кэше, то он передается в ОП. В противном случае информационный блок считывается с диска и заносится в кэш для дальнейшего использования. При поступлении запроса на запись информационный блок из ОП заносится вначале в дисковый кэш и лишь затем после выполнения соответствующих операций по поиску сектора — на диск, при этом обычно копия блока в дисковом кэше сохраняется. Запись информационного блока из ОП в кэш производится на место блока, копия которого сохранена на диске. Для управления процессами копирования вводятся специальные указатели, которые определяют, сохранена ли данная копия на диске, к какому информационному блоку обращение производилось ранее других и т.п. Копирование блока на диск производится по завершению операции поиска и не связано непосредственно с моментом поступления запроса.

Второй способ, позволяющий уменьшить снижение эффективной скорости обмена, вызванное операциями поиска на диске, связан с **использованием нескольких накопителей на диске.** Все информационные блоки распределяются по нескольким накопителям, причем так, чтобы суммарная интенсивность запросов по всем накопителям была одинаковой, а запросы по возможности чередовались. Если известны интенсивности запросов к «каждому информационному блоку, то можно ранжировать эти блоки, а если при этом известны и логические связи между блоками, то связанные блоки с примерно одинаковыми интенсивностями запросов должны размещаться в разных накопителях. Это позволяет совместить операции обмена между ОП и одним из накопителей с операциями поиска очередного блока в других накопителях.

5. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПОДСИСТЕМЫ ВВОДА-ВЫВОДА

5.1. Проблемы организации подсистем ввода-вывода

Производительность и эффективность использования ЭВМ определяются не только возможностями ее процессора и характеристиками основной памяти, но в очень большой степени составом ее периферийных устройств (ПУ), их техническими данными и способом организации их совместной работы с ядром (процессором и основной памятью) ЭВМ.

Связь устройств ЭВМ друг с другом осуществляется с помощью интерфейсов.

Интерфейс представляет собой совокупность линий и шин, сигналов, электронных схем и алгоритмов (протоколов), предназначенную для осуществления обмена информацией между устройствами. От характеристик интерфейсов во многом зависят производительность и надежность ЭВМ.

При разработке систем ввода-вывода должны быть решены следующие проблемы:

1. Должна быть обеспечена возможность реализации машин с переменным составом оборудования (машин с переменной конфигурацией). В первую очередь, с различным набором периферийных устройств с тем, чтобы пользователь мог выбирать состав оборудования (конфигурацию) машины в соответствии с ее назначением, легко дополнять машину новыми устройствами.

2. Для эффективного и высокопроизводительного использования оборудования ЭВМ должны реализовываться параллельная во времени работа процессора над программой и выполнение ПУ процедур ввода-вывода.

3. Для пользователя необходимо упростить и стандартизировать программирование операций ввода-вывода, обеспечить независимость программирования ввода-вывода от особенностей того или иного ПУ.

4. Необходимо обеспечить автоматическое распознавание и реакцию ядра ЭВМ на многообразие ситуаций, возникающих в ПУ (готовность устройства, отсутствие носителя, различные нарушения нормальной работы и Др.)

Особенно актуально решение этих проблем для машин, содержащих большое число разнообразных ПУ.

Отметим основные пути решения указанных проблем.

Модульность. Средства современной ВТ проектируются на основе модульного (или агрегатного) принципа. Он заключается в том, что отдельные устройства выполняются в виде конструктивно законченных модулей (агрегатов), которые могут сравнительно просто в нужных количествах и номенклатуре объединяться, образуя ЭВМ.

Присоединение нового устройства не должно вызывать в существующей части машины никаких изменений, кроме изменения кабельных соединений и некоторых корректировок программ.

Унифицированные (не зависящие от типа ПУ) **форматы данных,** которыми ПУ обмениваются с ядром ЭВМ, в том числе унифицированный формат сообщения, которое ПУ посылает в ядро о своем состоянии**.** Преобразование унифицированных форматов данных в индивидуальные, приспособленные для отдельных ПУ, производится в самих ПУ, точнее, в блоках управления ПУ (контроллерах, адаптерах).

Унифицированный интерфейс, т.е. унифицированный по составу и назначению набор линий и шин, унифицированные схемы подключения, сигналы и алгоритмы (протоколы) управления обменом информацией между ПУ и ядром ЭВМ.

Унифицированные (не зависящие от типа ПУ) **формат и набор команд** процессора для операций ввода-вывода. Операция ввода-вывода с любым ПУ представляет для процессора просто операцию передачи данных независимо от особенностей принципа действия данного ПУ, типа его носителя и т.п.

Унификация распространяется на семейство (ряд, систему) моделей ЭВМ.

Для обеспечения параллельной во времени работы ПУ с выполнением программы процессором схемы управления вводом-выводом отделяют от процессора и придают им достаточную степень автономности.

Многие функции управления операциями ввода-вывода, как например управление прямым доступом к памяти, являются общими, они не зависят от типа ПУ. Другие являются специфичными для данного типа устройств.

Выполнение общих функций возлагают на общие для групп ПУ унифицированные устройства — контроллеры прямого доступа к памяти, процессоры (каналы) ввода-вывода, а специфических — на специализированные для данного типа ПУ электронные блоки управления (адаптеры).

5.2. Способы организации передачи данных

В подсистемах ввода-вывода ЭВМ используются два основных способа организации передачи данных между памятью и ПУ: **программно-управляемая передача и прямой доступ к памяти** (ЦДЛ).

Программно-управляемая передача данных осуществляется при непосредственном участии и под управлением процессора, который при этом выполняет специальную подпрограмму ввода-вывода. Операция ввода-вывода может инициироваться центральным процессором, т.е. текущей командой программы, или запросом прерывания от ПУ. Первый случай является простым в реализации, но при обработке команды ввода-вывода ЦП бесполезно тратит время, ожидая готовности ПУ. Это значительно снижает производительность ЭВМ. Программно-управляемая передача, инициируемая запросом прерывания от ПУ, позволяет организовать более гибкое взаимодействие между ЦП и ПУ. Предположим, что в качестве ПУ используется клавиатура (клавишное устройство), предназначенная для ввода в ЭВМ команд, инструкций и данных. Каждый раз, когда пользователь (оператор) нажимает клавишу, ПУ выдает в ЦП запрос на прерывание. ЦП приостанавливает работу по текущей программе и передает управление подпрограмме ввода-вывода. Подпрограмма обрабатывает запрос и по ее завершению ЦП возвращается к работе по текущей программе. Выполнение текущей программы продолжается до следующего нажатия клавиши, и далее процесс повторяется. В этом случае преимущество от использования прерывания очевидно.

При программно-управляемой передаче данных ЦП на все время этой передачи отвлекается от выполнения основной программы. Операция пересылки данных логически слишком проста, чтобы эффективно загружать логически сложную быстродействующую аппаратуру процессора. Вместе с тем при пересылке блока данных ЦП приходится для каждой единицы передаваемых данных (байт, слово) выполнять довольно много инструкций, чтобы обеспечить буферизацию данных, преобразование форматов, подсчет количества переданных данных, формирование адресов в памяти и т.п. В результате скорость передачи данных при пересылке блока данных под управлением процессора оказывается недостаточной. Поэтому для быстрого ввода-вывода блоков данных и разгрузки ЦП от управления операциями ввода-вывода используют прямой доступ к памяти.

Прямой доступ к памяти

Прямой доступ к памяти — это такой способ обмена данными, который обеспечивает автономно от ЦП установление связи и передачу

данных между ОП и ПУ. Прямой доступ к памяти освобождает процессор от управления операциями ввода-вывода, позволяет осуществлять параллельно во времени выполнение процессором программы с обменом данными между ОП и ПУ, производить этот обмен со скоростью, ограничиваемой только пропускной способностью ОП или ПУ.

Таким образом, ПДП, разгружая процессор от обслуживания ввода-вывода, способствует возрастанию общей производительности ЭВМ. Повышение предельной скорости ввода-вывода информации делает машину более приспособленной для работы в системах реального времени. Прямым доступом к памяти управляет контроллер ПДП (рис. 5.1), который выполняет следующие функции:

1. Управление инициируемой процессором или ПУ передачей данных между ОП и ПУ.

2. Задание размера блока данных, который подлежит передаче и области памяти, используемой при передаче.

3. Формирование адресов ячеек ОП, участвующих в передаче.

4. Подсчет числа единиц данных (байт, слов), передаваемых от ПУ в ОП или обратно, и определение момента завершения заданной операции ввода-вывода.

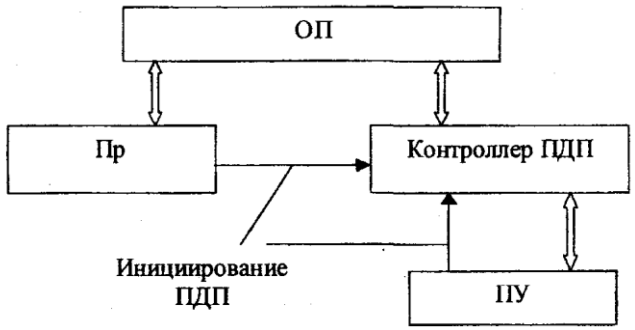


Рис.5.1. Прямой доступ к памяти

Контроллер ПДП обычно имеет более высокий приоритет в занятии цикла памяти по сравнению с процессором. Управление памятью переходит к контроллеру ПДП как только завершится цикл ее работы, выполняемый для текущей команды процессора.

ПДП обеспечивает высокую скорость обмена данными за счет того, что управление обменом производится не программным путем, а аппаратурными средствами.

В современных ЭВМ используется как программно-управляемая передача данных, так и прямой доступ к памяти.

Программно-управляемый обмен сохраняют для операций ввода-вывода отдельных байт (слов), которые выполняются быстрее, чем при ПДП, так как исключаются потери времени на программно-управляемую установку начальных состояний регистров и счетчиков контроллера ПДП (инициализация).

5.3. Унификация средств обмена и интерфейсы ЭВМ

5.3.1. Общая характеристика и классификация интерфейсов

Объединение отдельных подсистем (устройств, модулей) ЭВМ в единую систему основывается на многоуровневом принципе с

унифицированным сопряжением между всеми уровнями — стандартным интерфейсом. Под стандартными интерфейсами понимают такие интерфейсы, которые приняты и рекомендованы в качестве обязательных отраслевыми или государственными стандартами, различными международными комиссиями, а также крупными зарубежными фирмами.

Интерфейсы характеризуются следующими параметрами:

1) пропускной способностью интерфейса — количеством информации которая может быть передана через интерфейс в единицу времени;

2) максимальной частотой передачи информационных сигналов через интерфейс;

3) информационной шириной интерфейса — числом бит или байт данных, передаваемых параллельно через интерфейс;

4) максимально допустимым расстоянием между соединяемыми устройствами;

5) динамическими параметрами интерфейса — временем передачи отдельного слова или блока данных с учетом продолжительности процедур подготовки и завершения передачи;

6) общим числом проводов (линий) в интерфейсе.

В настоящее время не существует однозначной классификации интерфейсов. Можно выделить следующие четыре классификационных признака интерфейсов:

- способ соединения компонентов системы (радиальный, магистральный, смешанный);
- способ передачи информации (параллельный, последовательный, параллельно-последовательный);
- принцип обмена информацией (асинхронный, синхронный);
- режим передачи информации (двусторонняя поочередная передача, односторонняя передача).

На рис. 5.2 представлены радиальный и магистральный интерфейсы, соединяющие центральный модуль (ЦМ) и другие модули (компоненты) системы (M_1, ..., M_n).

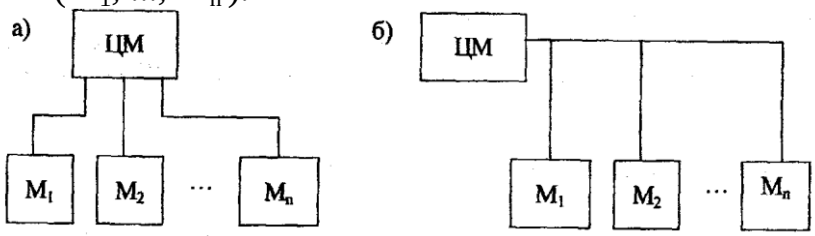


Рис.5.2. Радиальный (а) и магистральный (б) интерфейсы

Радиальный интерфейс позволяет всем модулям (м[, . . ., MJ работать независимо, но имеет максимальное количество шин. Магистральный интерфейс (общая шина) использует принцип разделения времени для связи между ЦМ и другими модулями. Он сравнительно прост в реализации, но лимитирует скорость обмена.

Параллельные интерфейсы позволяют передавать одновременно определенное количество бит или байт информации по многопроводной

линии. Последовательные интерфейсы служат для последовательной передачи по двухпроводной линии.

В случае синхронного интерфейса моменты выдачи информации передающим устройством и приема ее в другом устройстве должны синхронизироваться, для этого используют специальную линию синхронизации. При асинхронном интерфейсе передача осуществляется по принципу "запрос-ответ". Каждый цикл передачи сопровождается последовательностью управляющих сигналов, которые вырабатываются передающим и приемным устройствами. Передающее устройство может осуществлять передачу данных (байта или нескольких байтов) только после подтверждения приемником своей готовности к приему данных.

Классификация интерфейсов по назначению отражает взаимосвязь с архитектурой реальных средств вычислительной техники. В соответствии с этим признаком в ЭВМ и вычислительных системах можно выделить несколько уровней сопряжении:

- машинные системные интерфейсы;
- локальные шины;
- интерфейсы периферийных устройств (малые интерфейсы);
- межмашинные интерфейсы.

Машинные (внутримашинные) системные интерфейсы предназначены для организации связей между составными компонентами ЭВМ на уровне обмена информацией с центральным процессором, ОП и контроллерами (адаптерами) ПУ.

Локальной шиной называется шина, электрически выходящая непосредственно на контакты микропроцессора, и предназначенная для увеличения быстродействия видеоадаптеров и контроллеров дисковых накопителей. Она обычно объединяет процессор, память, схемы буферизации для системной шины и ее контроллер, а также некоторые вспомогательные схемы. Типичными примерами локальных шин являются VLB и PCI.

Назначение интерфейсов периферийных устройств (малых интерфейсов) состоит в выполнении функций сопряжения контроллера (адаптера) с конкретным механизмом ПУ.

Межмашинные интерфейсы используются в вычислительных системах и сетях.

С целью снижения стоимости некоторые компьютеры имеют единственную шину (общая шина) для памяти и устройств ввода-вывода. Персональные компьютеры первых поколений, как правило, строились на основе одной системной шины в стандартах ISA, EISA или MCA. Необходимость сохранения баланса производительности по мере роста быстродействия микропроцессоров привела к многоуровневой организации шин на основе использования нескольких системных и локальных шин. В современных компьютерах шины интерфейсов делят на шины, обеспечивающие организацию связи процессора с памятью, и шины ввода-вывода. Шины процессор-память сравни тельно короткие, обычно высокоскоростные и соответствуют организаций подсистемы памяти для обеспечения максимальной пропускной способности канала память-процессор. Шины ввода-вывода могут иметь большую протя-

женность, поддерживать подсоединение многих типов устройств и обычно следуют одному из шинных стандартов. Обычно количество и типы уст- ройств ввода-вывода в вычислительных системах не фиксируются, что дает возможность пользователю самому подобрать необходимую конфигурацию Шина ввода-вывода компьютера рассматривается как шина расширения; обеспечивающая постепенное наращивание устройств ввода-вывода. Поэтому стандарты играют огромную роль, позволяя разработчикам компьютеров и устройств ввода-вывода работать независимо.

53.2. Типы и характеристики стандартных шин

Типы и характеристики стандартных шин, используемых в настоящее время, приведены в табл. 5.1.

Таблица 5.1 Характеристики стандартные шин

Тип шины	Разрядно	Тактовая ча	Пропускн (Мб/сек)
ISA	16	8	16
EISA	32	8	33
MCA	32	10	-
VLB(VESA	32	40	130
VLB2	64*		400*
PCI	32	33,66	120, 133
VME32	32	-	32
VME64	64	-	160
Sbus	32,64	20,25	80,100
Mbus	64	50	125 (400)
XDBus	64	-	310(400)
AGP	32	133	533
PCI-X	64	133	1060

Системная шина ISA (Industry Standard Architecture) впервые стала применяться в ПК IBM PC/AT на базе процессора i2826. Данная шина позволяет передавать параллельно 16 бит данных и обращаться к 16 Мбайт системной памяти. В современных компьютерах используется как шина ввода/вывода для организации связи с медленно действующими периферийными устройствами.

С появлением процессоров i386, i486 системная шина ISA стала "узким местом" ПК на их основе. Другая системная шина EISA (Extended Industry Standard Architecture), разработанная в 1988 году, обеспечивает адресное пространство в 4 Гбайта, 32-битовую передачу данных, тактируется частотой около 8 Мгц, имеет максимальную теоретическую скорость передачи данных 33 Мбайт/с и совместима с шиной ISA.

Шина MCA также обеспечивает 32-разрядную передачу данных, тактируется частотой 10 МГц, но не совместима с шиной ISA и используется только в компьютерах компании IBM.

Локальная шина VESA-Local-Bus (VLB) предназначалась для увеличения быстродействия видеоадаптеров и контроллеров дисковых накопителей. Она подключалась непосредственно к процессору i486, и только к нему. После появления процессора Pentium ассоциация VESA приступила к работе над новым стандартом VLB версии 2, который предусматривает использование 64-битовой шины данных и увеличение количества разъемов расширения. Ожидаемая скорость передачи данных — до 400 Мбайт/сек.

Шина PCI (Peripheral Component Interconnection) в первом варианте использовалась как локальная шина и предназначалась для тех же целей, что и предыдущая шина (VLB). В действующем втором варианте шина PCI относится к шинам ввода/вывода. В данном случае соединение шин центрального процессора и PCI осуществляется через так называемую PCI-перемычку, мост PCI или контроллер, которые согласуют шину центрального процессора с шиной PCI. Это означает, что PCI может работать с процессорами различных платформ и поколений.

Шина УМЕ приобрела большую популярность как шина ввода/вывода в рабочих станциях и серверах на базе RISC-процессоров. Эта шина высоко стандартизирована, имеет несколько версий этого стандарта: VME32,
VME64.

В однопроцессорных и многопроцессорных рабочих станциях и серверах на основе микропроцессоров архитектуры SPARC одновременно используются несколько типов шин: Sbus, Mbus и XDBus, причем шина Sbus применяется в качестве шины ввода/вывода, а Mbus и XDBus — в качестве шин для объединения большого числа процессоров и памяти.

Спустя почти четыре года с того времени, когда шина PCI стала стандартом в настольных ПК, корпорация Intel объявила о новой, предназначенной исключительно для графики, шине AGP, способной повысить производительность видео-, 2D-, 3D-приложений. Шина AGP (Accelerated Graphics Port) относится к локальным шинам. Для использования технологии AGP необходим набор микросхем Intel 440LX (появившийся в 1997 году), который позволяет разгрузить сравнительно "узкую" (133 Мб/с) шину PCI от жадного на ресурсы видеоадаптера и подключить последний к специально предназначенной для него более "широкой" (528 Мб/с) шине AGP. На долю же PCI остаются более медленные устройства, функционирование которых существенно улучшается благодаря отключению от шины более быстродействующих устройств, то и дело создающих "пробки" в стремительном потоке данных. Набор 440LX не только имеет поддержку AGP, но и допускает использование в машинах на базе Pentium II быстродействующей памяти SDRAM, которая обеспечивает более высокую производительность, чем ОЗУ типа EDO DRAM, применяемое в машинах Pentium II со старым набором микросхем 440 FX. Конструктивно 440 LX состоит из двух устройств: микросхемы 82443LX (PAC или PCI AGP Controller) и многофункционального моста 82371AB (PIIX4 или PCI, ISA, IDE Accelerator).

В целом же шинная архитектура настольного ПК нового (на ближайшие два-три года) поколения содержит несколько шин (рис. 5.3) с различной

пропускной способностью: шины (1 Гб/с), соединяющей ядро Pentium II с кэш-. памятью второго уровня, трех шин (528 Мб/с), соединяющих новый набор AGPset с ядром процессора, SDRAM и графическим акселератором, а также шины PCI (133 Мб/с).

Применение такой шиной организации увеличивает быстродействие компьютеров при выполнении целочисленных операций, действий с плавающей запятой и работе с мультимедиа-приложениями.

В 1998 году три крупнейшие компьютерные компании — Compaq, Hewlett-Packard и IBM — разработали новую спецификацию — расширение шины PCI, названную PCI-X, которая работает на тактовой частоте 133 МГц. Шина PCI-X обладает обратной совместимостью с PCI, требует нового набора микросхем Intel 450 NX, кроме того, благодаря новой схеме обмена регистр-регистр достигается пропускная способность 1,06 Гб/с (8 Гбит/с), что обеспечивает почти шестикратный выигрыш в производительности. В первую очередь PCI-X предназначена для подключения высокопроизводительных адаптеров типа Gigabit Ethernet, Ultra 3SCSI и Fibre Channel (FC-AL).

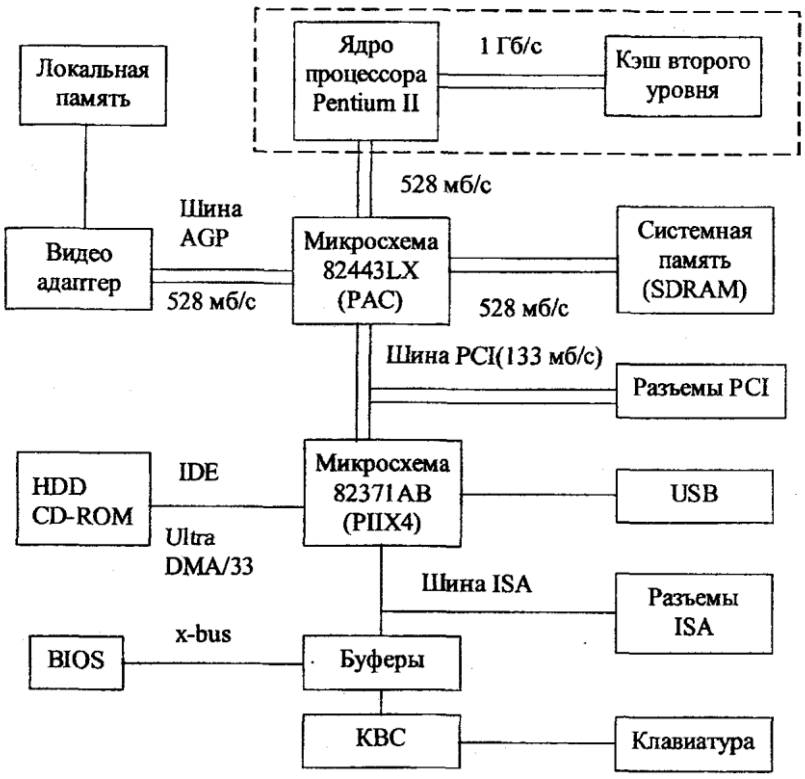


Рис.5.3. Шинная архитектура ПК на базе набора микросхем 440LX

5.4. Современные и перспективные структуры подсистем ввода-вывода

Учитывая дальнейший рост производительности микропроцессоров, а также недостатки и ограничения топологии "общая шина", осенью 1998 г. корпорация Intel обнародовала принципиально иную архитектуру, которую скромно адресовала следующему поколению подсистем ввода-вывода — Next Generation I/O (NGVO). Собственно, это было ответом на

спецификацию PCI-X.

Основными чертами новой архитектуры являются последовательный обмен данными, канальная технология ввода-вывода и матричная топология. Таким образом, в интеловской архитектуре компьютеров появляются каналы ввода-вывода, которые были на время забыты (хотя до сих пор используются в мэйнфреймах). Базовый микропроцессор не будет сам заниматься рутинной работой по обмену данными с ПУ, а станет только инициировать прием или передачу, давая соответствующие указания процессору (контроллеру) канала. Немаловажно и то, что ПУ будут иметь доступ к ОП исключительно через контроллер канала.

Топология матричной коммутации, заложенная в NGI/0, позволяет взаимодействовать всем устройствам, входящим в матрицу, по принципу "каждый с каждым". Ее задачей является распределение данных по каналам. Ключи матрицы временно образуют коммутационный канал между компьютером и ПУ, организуя обмен "точка-точка".

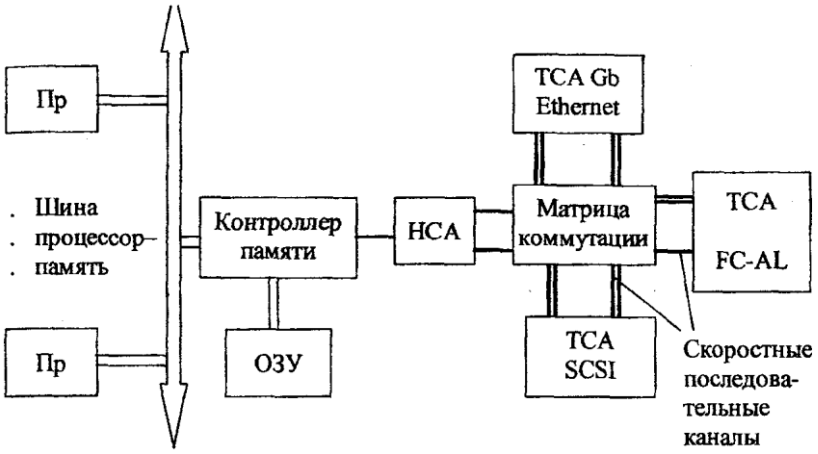
Благодаря этой технологии исключается проблема арбитража и конфликтов, "горячая" замена устройств становится действительно автоматической, существенно облегчается конфигурирование контроллеров (причем общая производительность не ухудшается из-за неправильного конфигурирования одного из них), расстояние между периферийным контроллером и контроллером памяти может быть увеличено до 30 м. Стоит также отметить, что трудностей с расширением при использовании этой топологии практически не существует. По некоторым данным с помощью NGI/0 к системе (серверу) можно подключить до 64 тыс. устройств.

В качестве интерфейса контроллера памяти сервера (рис. 5.4) служит главный канальный адаптер HCA (Host Channel Adapter). Он содержит процессор прямого доступа к памяти (DMA). Для связи между матрицей коммутации и контроллерами ввода-вывода ПУ предназначены объектные канальные адаптеры TCA (Target Channel Adapter). Канальный адаптер может подключаться к другому адаптеру или ключу. Эти ключи и образуют матрицу коммутации.

Скорость передачи для одного канала NGI/0 оценивается на уровне 1,25 - 2,5 Гбит/с, однако при увеличении числа каналов до четырех она соответственно возрастет до 10 Гбит/с.

Сразу после объявления NGI/0 по инициативе корпорации IBM был создан альянс для разработки открытого стандарта на архитектуру под названием Future I/O. Уже известно, что для данной архитектуры, как и для NGI/0 планируется использовать матричную топологию, позволяющую получить соединение типа "точка-точка". Правда в отличие от NGI/0 в спецификации Future I/O допускается применение PCI-адаптеров. Сделано это для того, чтобы продлить жизнь своему детищу - PCI-X. Максимальная производительность одного соединения может достигать 2 Гб/с по медному кабелю на расстоянии 5 - 10 м, а по оптоволоконному - до 300 м.

В начале 2000 г. спецификация должна быть утверждена, а первые результаты применения новой технологии могут быть получены не ранее 2001 г.



Рнс.5.4. Архитектура подсистемы ввода-вывода NGI/0

6. МНОГОПРОЦЕССОРНЫЕ И МНОГОМАШИННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

6.1. Архитектуры вычислительных систем

Точно также, как однопроцессорные компьютеры, представлены архитектурами с одним потоком данных SISD и множеством потоков данных SIMD, так и многопроцессорные системы могут быть представлены двумя базовыми типами архитектур в зависимости от параллелизма данных:

MISD — множество потоков команд — один поток данных;

MIMD — множество потоков команд — множество потоков данных. Класс MISD долгое время пустовал, поскольку не существовало практических примеров реализации систем, в которых одни и те же данные обрабатываются большим числом параллельных процессов. В дальнейшем для MISD нашлась адекватная организация вычислительной системы — распределенная мультипроцессорная система с общими данными.

Наиболее простая и самая распространенная система этого класса — обычная локальная сеть персональных компьютеров, работающая с единой базой данных, когда много процессоров обрабатывают один поток данных. Впрочем, тут есть одна тонкость. Как только в такой сети все пользователи переключаются на обработку собственных данных, недоступных для других абонентов сети, MISD — система превращается в систему с множеством потоков команд и множеством потоков данных, соответствующую MIMD - архитектуре. Так как только MIMD-архитектура включает все уровни параллелизма от конвейера операций до независимых заданий и программ, то любая вычислительная система этого класса в частных приложениях может выступать как SISD и SIMD-система. Например, если многопроцессорный комплекс выполняет одну-единственную программу без каких-либо признаков векторного параллелизма данных, то в этом конкретном случае он функционирует как обычный SISD-компьютер, и весь его потенциал остается невостребованным. Таким образом, употребляя термин «MIMD», надо иметь в виду не только много процессоров, но и множество вычислительных процессов, одновременно выполняемых в системе. MIMD-системы по

способу взаимодействия процессоров (рис. 6.1) делятся на системы с сильной и слабой связью.

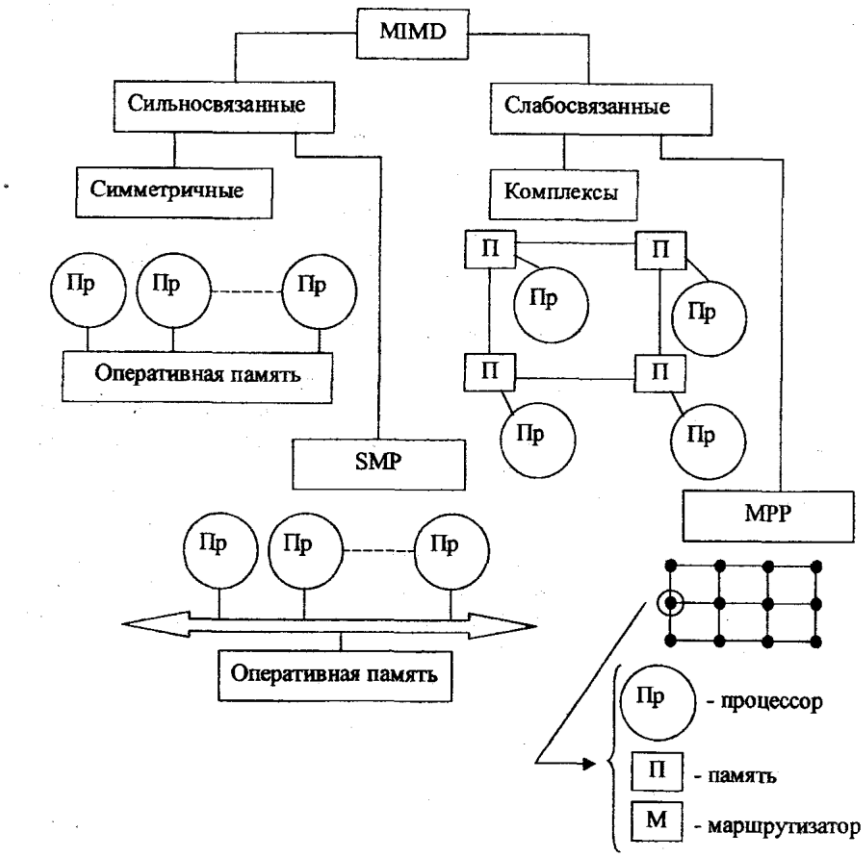


Рис.6.1. Классификация вычислительных систем с MIMD-архитектурой

Системы с сильной связью (иногда их называют «истинными» мультипроцессорами) основаны на объединении процессоров на общем поле оперативной памяти.

Системы со слабой связью, как правило, представляются многомашинными комплексами, в которых отдельные компьютеры объединяются либо с помощью сетевых средств, либо с помощью общей внешней памяти (обычно — дисковые накопители большой емкости). Разница организации MIMD-систем с сильной и слабой связью проявляются при обработке приложений, отличающихся интенсивностью обменов между процессами.

6.2. Сильно связанные многопроцессорные системы

В таких системах, как правило, число параллельных процессов невелико (не больше 16) и управляет ими централизованная операционная система. Процессы обмениваются информацией через общую оперативную память. При этом возникают задержки из-за межпроцессорных конфликтов. При создании больших мультипроцессорных ЭВМ (мэйнфреймов, суперЭВМ) предпринимаются огромные усилия по увеличению пропускной способности оперативной памяти. В результате аппаратные затраты возрастают чуть ли не в квадратичной зависимости, а производительность системы упорно «не желает» увеличиваться пропорционально числу процессоров. Так, сложнейшие средства снижения межпроцессорных конфликтов в

оперативной памяти суперкомпьютеров серии CRAY X-MP/Y-MP позволяют получить коэффициент ускорения не более 3,5 для четырехпропессррной конфигурации системы.

То, что могут себе позволить дорогостоящие и сложные мэйнфреймы и суперкомпьютеры, не годится для компактных многопроцессорных серверов. Для простой и «дешевой» поддержки многопроцессорной организации была предложена архитектура SMP — мультипроцессирование с разделением памяти, предполагающая объединение процессоров на общей шине оперативной памяти. За аппаратную простоту реализации средств SMP приходится расплачиваться процессорным временем ожидания в очереди к шине оперативной памяти. В большинстве случаев пользователи готовы добавить в сервер один или более процессоров (но редко — более четырех) в надежде увеличить производительность системы. Стоимость этой операции ничтожна по сравнению со стоимостью всего сервера, а результат чаще всего оправдывает ожидания пользователя. Архитектура SMP стала своего рода стандартом для всех современных многопроцессорных серверов (например, HP9000 и DEC Alpha Server AXP). Стремительное увеличение пропускной способности системных шин предопределяет широкое распространение SMP архитектуры.

6.3. Слабосвязанные многопроцессорные системы

Слабосвязанные процессы время от времени обмениваются небольшими блоками данных, т.е. не предъявляют больших требований к пропускной способности межпроцессорных связей. Теоретически наиболее удачным архитектурным решением для обработки подобных процессов являются системы с массовым параллелизмом (MPP), состоящие из десятков, сотен, а иногда и тысяч процессорных узлов. Каждый узел такой системы содержит процессор и модуль памяти, в котором хранится процесс — совокупность команд, исходных и промежуточных данных вычислений, а также системные идентификаторы процесса. Узлы массово-параллельной системы объединяются коммутационными сетями самой различной формы — от простейшей двумерной решетки до гиперкуба или трехмерного тора. В отличие от архитектуры фон Неймана, передача данных между узлами коммутационной сепг происходит по готовности данных процесса, а не под управлением некоторой программы. Отсюда еще одно название подобных систем — «системы с управлением потоком данных» (иногда просто «потоковые машины»).

К достоинствам данной архитектуры относится то, что она использует стандартные микропроцессоры и обладает неограниченным быстродействием (терафлопсы).

Однако есть и недостатки — программирование коммутаций процессов является слабо автоматизированной и очень сложной процедурой. Так что для коммерческих задач и даже для подавляющего большинства инженерных приложений системы с массовым параллелизмом недоступны.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Каган Б.М. Электронные вычислительные машины и системы. Учеб. пособие для вузов. - 3-е изд. - М.: Энергоатомиздат, 1991.
2. Вычислительные машины, комплексы и сети: Учебник /А.П. Пятибратов, С.Н. Беляев, Г.М. Козырева и др.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 1991.
3. Пятибратов А.П., Гудыко Л.П., Кириченко А.А. Вычислительные системы, сети и телекоммуникации. Учебник / Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 1998.
4. Вычислительные машины и системы: Учебник для вузов /В.Д. Ефремов, В.Ф. Мелехин, К.П. Дурандин и др.; Под ред. В.Д. Ефремова. - М.: Высш. шк., 1993.
5. Водяхо А.И., Горнец Н.Н., Пузанков Д.В. Высокопроизводительные системы обработки данных: Учеб. пособие для вузов. - М.: Высш. шк., 1997.
6. Компьютерные системы и сети. Учеб. пособие /В.П. Косарев и др. — М.: Финансы и статистика, 1999.
7. Информатика: Учебник /Под ред. Н.В. Макаровой. - М.: Финансы и статистика, 1997.

8. Амамия М., Танака К>. Архитектура ЭВМ и искусственный интеллект /Пер. с японск. - М.: Мир, 1996.
9. Перспективы развития вычислительной техники: В 11 кн.: Справ, пособие /Под ред. Ю.М. Смирнова. - М.:Высш. шк., 1989. - Кн.З: ЭВМ общего назначения. /Ю.С. Ломов, К.С. Ораевский, А.П. Заморин, А.И. Слуцкий.
Ю.Мячев А.А., Степанов В.Н. Персональные ЭВМ и микро ЭВМ. Основы организации: Справочник./Под ред. А.А. Мячева.-М.:Радио и связь, 1991.
П.Гук М. Процессоры Pentium II, Pentium Pro, просто Pentium. - СПб.: Питер КОМ, 1999.
12. Корнеев В.В. Параллельные вычислительные системы. - М.: «Нолидиж», 1999.
13-Леонтьев В. Новейшая энциклопедия ПК. - М.: ОЛМА-ПРЕСС, 1999.
14-Кодесов А. Новшества архитектуры Katmaill PC Week/RE № 42-43,1998.
15.Бройтман Д. Микропроцессор Pentium. Часть *11* Монитор, № 4,1993.
16.Бродин В.Б., Шагурин И.И. Микропроцессор i486. Архитектура, программиро-вание, интерфейс.-М.: «ДИАЛОГ-МИФИ», 1993.
17.Супер ЭВМ. Аппаратная и программная организация/ Пер. с англ.; Под ред. С. Фернбаха. - М.: Радио и связь, 1991.

ОГЛАВЛЕНИЕ

Андрей Дмитриевич Чередов

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Учебное пособие

Научный редактор: профессор, докт. техн. наук В.М. Разин

Редактор: Н. Т. Синельникова

Компьютерный набор выполнила Т.Н. Калинина

Подписано к печати 4.09.2000. Формат 60 х 84 /16, Бумага ксероксная. Плоская печать. Усл. печ. л. 7,91. уч.-изд. л. 7,16.. Тираж 300 экз. Заказ № 111 . Цена свободная. ИПФ ТПУ Лицензия ЛТ №1 от 18.07.94 Типография ТПУ. 634034, г. Томск, пр. Ленина, 30.