ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Н.И. Лиманова

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И КОМПЬЮТЕРНЫХ СЕТЕЙ

Учебное пособие

Самара
2017

УДК 681.3
ББК 32.973.2
Л

Печатается по решению Методического совета ПГУТИ,
протокол № 55 от 27.03.17.

Рецензент: профессор, д.т.н.., зав. каф. информационных систем и технологий Самарского национального исследовательского университета им. акад. С.П.Королева, заслуженный работник высшей школы РФ Прохоров С.А.

Л Лиманова Н.И.
Архитектура вычислительных систем и компьютерных сетей. Учебное пособие. Н.И. Лиманова. – Самара: ПГУТИ, 2017. – 197 с.: ил.

Учебное пособие предназначено для студентов и аспирантов, обучающихся по техническим направлениям подготовки и соответствует Государственному стандарту РФ.

Пособие содержит сведения об эволюции вычислительной техники от ENIAC и БЭСМ до современных кластеров и суперкомпьютеров. Подробно рассмотрены арифметические и логические основы организации вычислительных систем, различные типы процессоров и многопроцессорные системы. Описаны принципы организации памяти и ввода/вывода в вычислительных системах. Рассмотрены основные типы архитектур вычислительных систем и сетевые архитектуры.

ISBN

СПИСОК СОКРАЩЕНИЙ

АЛУ - арифметико-логическое устройство
АЦП - аналого-цифровой преобразователь
АВМ - аналогово-вычислительная машина
БПЗ - блок обработки чисел в формате с плавающей запятой
БИС - большая интегральная схема
СБИС - сверх-большая интегральная схема
БУС - блок усилителей считывания
БУЗ - блок усилителей записи
БУП - блок управления памятью
БЭСМ - Большая Электронно-Счётная Машина
ПЭВМ - Персональная электронная вычислительная машина
ЗМ - Запоминающий массив
ЗЭ - запоминающий элемент
МПС - мульти-процессорная система
ППЗУ - программируемое постоянное запоминающее устройство
ПЭ - Процессорный элемент
РАП - Регистр ассоциативного признака
РИА - Регистр индикаторов памяти
СОЗУ - Сверхоперативное запоминающее устройство
СУПО - система управления пакетной обработкой
УУ - устройство управления
AFP — Appletalk Filing Protocol
ATA - Advanced Technology Attachment
CISC - компьютер с полным набором команд
CAS - Column-Access Strobe
CBR - CAS before RAS
CPI - количество тактов синхронизациина одну команду
DSP - Digital Signal Processor - Цифровой сигнальный процессор
EISA - Extended Industry Standard Architecture
ENIAC - электронный цифровой интегратор и вычислитель
EPP - Enchanced Parallel Port - Улучшенный параллельный порт
FPU - сопроцессор для операций с плавающей точкой
FSB - Front Side Bus - Шина для работы с процессором
GDPS - Географически-распределённый параллельный Sysplex
IDE - Integrated Drive Electronics - встроенный контроллер
ISO - International Standart Organisation
ILLIAC - ILLInois Automatic Computer
MA - Multiplexed-Address - Мультиплексированная шина

MIPS -миллионы операций в секунду
MISC - компьютер с минимальным набором команд
MISD - Множественный поток команд/Одиночный поток данных
MIMD - Множественный поток данных
MPP - массивно-параллельная архитектура
NNTP - сетевой протокол передачи новостей
NMI - non-maskable interrupt - Прерывание
PROM - программируемая память только для чтения
PCI-E - Взаимосвязь периферийных компонентов - Экспресс
RAID - Избыточный массив независимых дисков
RISC - компьютер с сокращённым набором команд
SAM - память с последовательным доступом
SCSI - Small Computer System Interface
SEC/DED - Single Error Correction/Double Error DetectionSIP
SISD - Одиночный поток команд/одиночный поток данных
SIMD - Одиночный поток команд/Множественный поток данных
SMP - Симметричные мультипроцессоры
STM - Synchronous Transfer Mode - синхронный режим передачи
SOC - System-on-a-Chip - Однокристальная система
Sysplex - System Complex - системный комплекс

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время трудно найти такую сферу деятельности человека, где не применялись бы вычислительные системы (ВС) и не использовались компьютерные сети (КС) или ту, которая хотя бы косвенно не зависела от их применения. Данное учебное пособие написано, чтобы помочь вам получить действительно необходимые знания о ВС. Можно пользоваться компьютером, сетью или ВС, не разбираясь в них по-настоящему. Чем больше, однако, у вас будет знаний в данной области, тем легче вы раскроете потенциал вычислительной техники и тем легче преодолеете проблемы, время от времени возникающие при работе с КС. Кроме того, спрос на людей, обладающих знаниями в области ВС и КС, весьма велик. Вспомните, по аналогии, ранние времена автомобилизма, когда для того, чтобы уверенно путешествовать на машине, приходилось становиться механиком. Сегодня для успешной работой с ВС и сетью нужна хоть какая-то степень квалификации, и чем больше ее у вас, тем легче вы справитесь с проблемами, которые иногда случаются. Разбираясь в вычислительной технике, можно добиться большей эффективности в работе с ней.

В первой главе приводятся сведения об эволюции средств вычислений от докомпьютерной эпохи до современных кластеров и суперкомпьютеров, рассматриваются открытия, предшествовавшие созданию компьютеров и поколения ЭВМ, приводится классификация современных вычислительных машин и систем. Рассматриваются характеристики и области применения компьютеров, кластеров, суперкомпьютеров и систем распределенных вычислений.

Вторая глава посвящена арифметическим основам организации вычислительных систем: особенностям представления целых чисел и чисел с плавающей точкой в ВС. Вы узнаете о прямом, обратном и дополнительном кодах, о системе со смещением, познакомитесь с кодированием звука, текстовых и графических данных и с тем, как происходит сложение, вычитание, умножение и деление чисел.

Третья глава напомнит понятия и законы алгебры логики и пояснит логику организации ВС. Вы уясните основные принципы минимизации устройств ВС с помощью карт Карно.

В четвертой главе рассмотрены базовые архитектуры ЭВМ. Концепция многоуровневой работы ВС является ключом к пониманию уровней организации вычислительных процессов и взаимодействия уровней. Показано, что многоуровневая организация упрощает

реализацию ЭВМ и управление вычислительным процессом, но снижает эффективность работы за счет «накладных расходов» на управление. Мультипроцессоры и компьютеры параллельного действия получили широкое распространение в настоящее время, поэтому особое внимание уделено рассмотрению базовых параллельных архитектур, используемых в современных ВС.

В пятой главе рассматриваются процессоры и функциональные устройства ВС, в шестой и седьмой — принципы организации памяти и ввода/вывода в ВС.

Описание разновидностей архитектур компьютерных сетей, типичных конфигураций локальных сетей и коммуникационного оборудования приведено в восьмой главе учебного пособия.

Чтобы разъяснить, как принципы, изложенные в учебном пособии, могут применяться на практике, в нем приведены многочисленные примеры реализаций узлов ВС, отдельных компьютеров и ВС в целом. В конце каждой главы приводятся вопросы и задания для самоконтроля, которые способствуют лучшему усвоению пройденного материала. Автор настоятельно рекомендует ответить на приведенные вопросы и выполнить задания, прежде, чем переходить к следующему разделу.

Всем, держащим в руках данное пособие, желаю успехов в освоении одноименной дисциплины.

1. ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ. КЛАССИФИКАЦИЯ СОВРЕМЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И СИСТЕМ

1.1. Понятия вычислительной машины и вычислительной системы

Вычислительная машина — это комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей.

Вычислительная система — это совокупность взаимосвязанных и взаимодействующих процессоров или вычислительных машин (ВМ), периферийного оборудования и программного обеспечения, предназначенную для подготовки и решения задач пользователей.

Под архитектурой вычислительной машины обычно понимается логическое построение ВМ, то есть то, какой машина представляется программисту. Из рассмотрения выпадают вопросы физического построения вычислительных средств: состав устройств, число регистров процессора, емкость памяти, наличие специального блока для обработки вещественных чисел, тактовая частота центрального процессора и т.д. Этот круг вопросов принято определять понятием организация вычислительной машины.

Уровни детализации структуры вычислительной машины

На первом уровне вычислительная машина рассматривается как устройство, способное хранить и обрабатывать информацию, а также обмениваться данными с внешним миром (см. рис. 1.1, а). ВМ представляется «черным ящиком», который может быть подключен к коммуникационной сети и к которому, в свою очередь, могут подсоединяться периферийные устройства.

Уровень общей архитектуры (см. рис. 1.1,6) предполагает представление ВМ в виде четырех составляющих: центрального процессора (ЦП), основной памяти (ОП), устройства ввода/вывода и системы шин.

На третьем уровне детализируется каждое из устройств второго уровня. Для примера взят центральный процессор (см. рис. 1.1, в). В простейшем варианте в нем можно выделить: арифметико-логическое устройство (АЛУ), обеспечивающее обработку целых чисел; блок обработки чисел в формате с плавающей запятой (БПЗ); регистры

процессора, использующиеся для краткосрочного хранения команд, данных и адресов; устройство управления (УУ), обеспечивающее совместное функционирование устройств ВМ; внутренние шины.

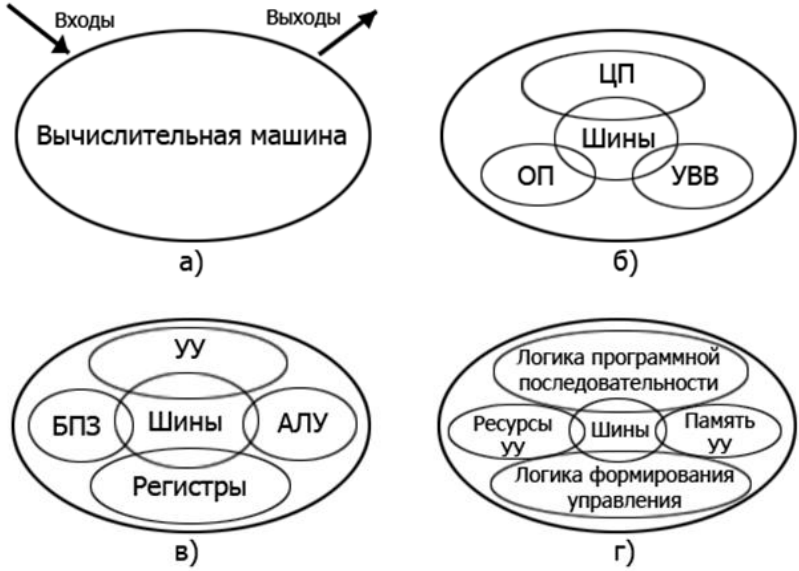


Рис 1.1. Уровни детализации вычислительной машины:
а) уровень «черного ящика»
б) уровень общей архитектуры
в) уровень архитектуры центрального процессора
г.) уровень архитектуры устройства управления

На четвертом уровне детализируются элементы третьего уровня. Так, на рис. 1.1, г раскрыта структура устройства управления. УУ представлено в виде четырех составляющих: логики программной последовательности — электронных схем, обеспечивающих выполнение команд программы в последовательности, предписываемой программой; регистров и дешифраторов устройства управления; управляющей памяти; логики формирования управления, генерирующей все необходимые управляющие сигналы.

Применительно к параллельным и распределенным многопроцессорным и многомашинным вычислительным системам зачастую вводят понятие «метауровня».

1.2. История докомпьютерной эпохи

Еще во времена древнейших культур человеку приходилось решать задачи, связанные с торговыми расчетами, с исчислением времени, с определением площади земельных участков и т. д. Рост размеров этих расчетов приводил даже к тому, что из одной страны в другую приглашались специально обученные люди, отлично владеющие техникой арифметического счета. Поэтому существовала реальная необходимость в устройствах, облегчающих выполнение повседневных расчетов.

Так, в старой Греции и в старом Риме были сделаны приспособления для счета, называемые абак. Абак называют также римскими счетами. Он представлял собой доску, покрытую пылью или песком. На ней можно было чертить линии и перекладывать камешки. Основное его назначение состояло в выполнении простых арифметических операций простым перемещением счетных элементов. Абак служил преимущественно для выполнения денежных расчетов. Счет велся в двоично-пятеричной системе счисления. Внешний вид этого устройства для счета показан на рис. 1.2.

В странах старого Востока существовали китайские счеты. Счет осуществлялся единицами и пятерками. В России для арифметических вычислений применялись российские счеты, появившиеся в 16 веке, но кое-где счеты можно встретить и сейчас.

Рис 1.2. Абак — устройство для счета

Развитие приспособлений для счета шло в ногу с достижениями математики. Скоро после открытия логарифмов в 1623 г. Была изобретена логарифмическая линейка, ее автором был английский математик Эдмонд Гантер. Логарифмической линейке суждена была долгая жизнь: от 17 века и приблизительно до конца XX века.

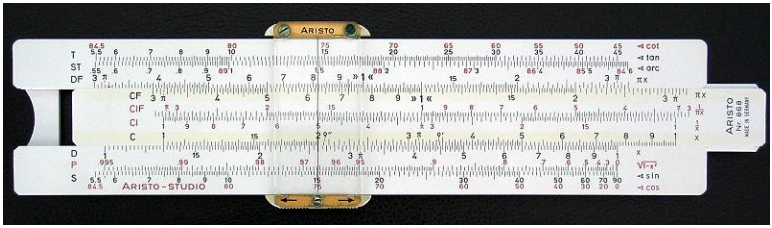


Рис 1.3. Логарифмическая линейка

Ни абак, ни счеты, ни логарифмическая линейка не означают механизации процесса вычислений. В 17 веке выдающимся французским ученым Блезом Паскалем было изобретено принципиально новое счетное устройство - арифметическая машина. В базу ее работы Б. Паскаль положил известную до него идею выполнения вычислений с помощью металлических шестеренок. В 1645 г. Им была построена первая суммирующая машина, а в 1675 г. Паскалю удается сделать реальную машину, выполняющую все четыре арифметических деяния. Практически сразу с Паскалем в 1660 - 1680 гг. Сконструировал счетную машину великий германский математик Готфрид Лейбниц.

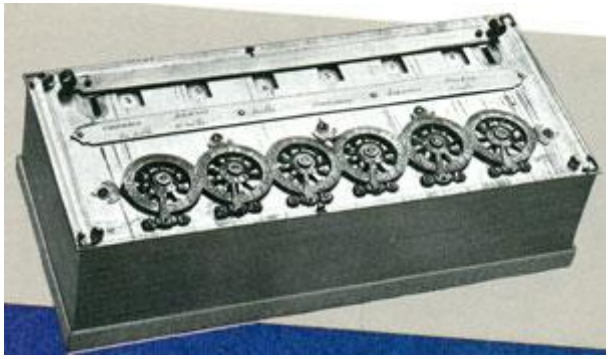


Рис 1.4. Арифметическая машина Б. Паскаля

Счетные машины Паскаля и Лейбница стали прообразом арифмометра. Первый арифмометр для четырех арифметических действий, нашедший арифметическое применение, удалось выстроить лишь через сто лет, 1790 г., германскому часовому мастеру Гану. Потом устройство арифмометра совершенствовалось многими механиками из Англии, Франции, Италии, России, Швейцарии. Арифмометры применялись для выполнения сложных вычислений при проектировании и строительстве кораблей, мостов, зданий, при проведении денежных операций. Но производительность работы на арифмометрах оставалась низкой, настоятельным требованием времени была автоматизация вычислений.

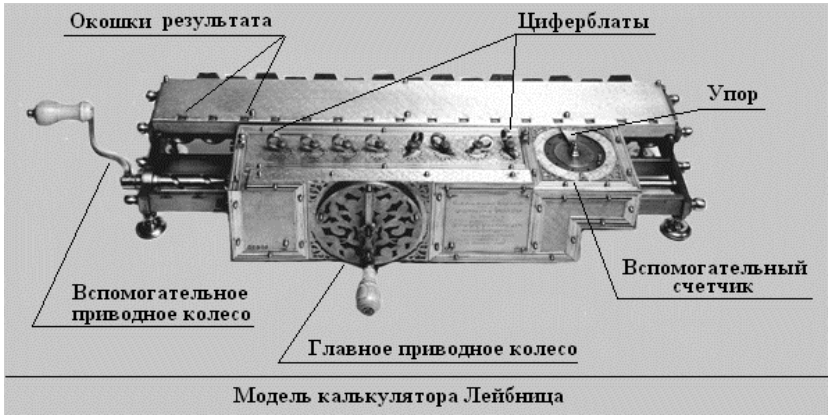


Рис 1.5. Арифмометр

В 1834 г. английский ученый Чарльз Бэббидж закончил описание машины, он назвал ее «аналитической машиной». По его плану, эта машина обязана была стать огромным арифмометром с программным управлением, она была способна не просто считать, но и управлять ходом собственной работы в зависимости от заложенной программы, то есть он пытался воплотить идею программного управления вычислительным процессом.

В машине Бэббиджа предусмотрены были также арифметические и запоминающие устройства. Его машина стала прообразом будущих компьютеров. Это изобретение опередило эпоху на 100 лет. Но в ней использовались далеко не совершенные узлы, к примеру, для запоминания разрядов десятичного числа в ней применялись зубчатые колеса. Выполнить свой проект Бэббиджу не

удалось из-за недостаточного развития техники, и «аналитическая машина» на время была забыта.

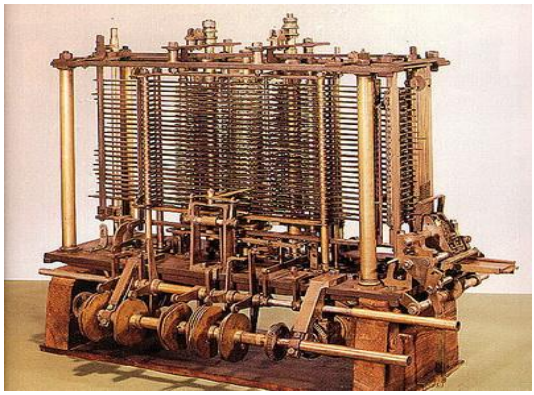


Рис 1.6. Аналитическая машина Ч. Бэббиджа

В 1887 году Герман Холлерит изобрел устройство названное табулятором - вычислительная машина, предназначавшаяся для автоматической обработки числовой и буквенной информации, записанной на перфокартах.

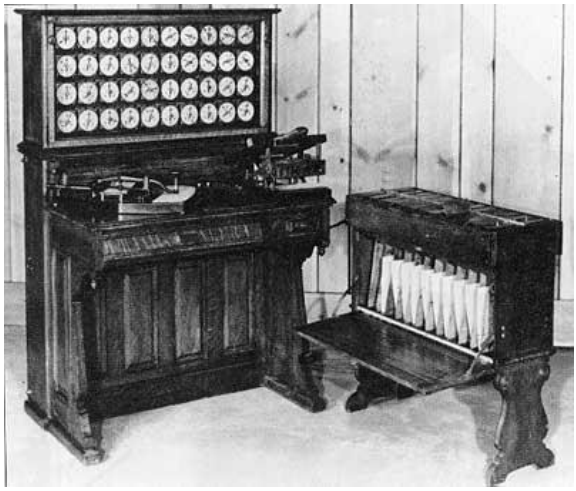


Рис 1.7. Табулятор Г. Холлерита

В конце 30 - х годов XX века германский инженер Конрад Цузе разработал первую двоичную цифровую машину Z1. В ней обширно использовались электромеханические реле, то есть механические переключатели, приводимые в действие электрическим током. В 1941г. К. Цуцзе создал машину Z3, полностью управляемую с помощью программы.

В 1944 г. Американец Говард Айкен на одном из предприятий компании IBM выстроил мощную по тем временам машину «Марк - 1». В данной машине для представления чисел использовались механические элементы - счетные колеса, а для управления применялись электромеханические реле.

Рис 1.8. Тестирование вычислительной машины «Марк – 1»

Таким образом, краткая история докомпьютерной эпохи показывает, что человечество стремилось изобрести устройства, облегчающие математические расчета. Счетные машины XVII- XVIII веков шли в ногу с развитием математики. К сожалению,

недостаточный уровень развития техники не позволил практически и в полной мере реализовать все великие идеи.

1.3. Эволюция ЭВМ в XX веке. Поколения ЭВМ

Открытия, предшествовавшие созданию компьютеров

Компьютер — величайшее изобретение XX века. Его созданию предшествовали открытия в области физики, математики, техники.

Во-первых, в конце XIX века получила развития математическая физика. Стали нужны машины, способные производить многократно повторяющиеся вычисления.

Во-вторых, в 1800 году американский изобретатель Т. Эдисон открыл явление термоэлектронной эмиссии, что послужило основой для создания в 1904 году английским физиком Дж. Флемингом диода, прибора обладающего односторонней проводимостью электрического тока. Несколько позже был создан еще один вакуумный прибор — триод.

В-третьих, английский математик Дж. Буль еще в 1884 году описал правила логики, впоследствии названной его именем — булева алгебра. В соответствии с логикой алгебраические элементы могут принимать только два значения — истина (1) или ложь (0) . Благодаря этой логике стало возможным конструирование логических схем.

И, в-четвертых, в 1918 году русский ученый М.А. Бонч - Бруевич и, независимо от него, английские ученые создали электронное реле, которое могло находиться в одном из двух состояний — 0 или 1, на базе которого был создан триггер.

Можно сказать, что к XX веку все было подготовлено для создания компьютера. Выше перечисленные события имели большое значение. Они создали предпосылки для появления компьютера.

Первое поколение ЭВМ

В вычислительной технике существует своеобразная периодизация развития электронных вычислительных машин. Всю электронно-вычислительную технику принято делить на поколения. ЭВМ относят к тому либо иному поколению в зависимости от типа главных используемых в ней частей либо от технологии их производства. От элементной базы зависит мощность компьютера, что в свою очередь привело к изменениям в архитектуре ЭВМ, расширению круга ее задач, к изменению способа взаимодействия пользователя и компьютера. Ясно, что границы поколений в смысле времени сильно размыты, так как в одно и то же время практически

выпускались ЭВМ разных типов; для отдельной же машины вопрос о её принадлежности к тому либо иному поколению решается довольно просто.

Предшественниками ЭВМ были релейные вычислительные машины. Реле позволяло кодировать информацию в двоичном виде состояниями включено-выключено. В процессе работы такой машины тысячи реле переключались из одного состояния в другое. Такие машины работами с низкой скоростью (50 сложений или 20 умножений в секунду).

С развитием в первой половине XX века радиотехники связан переход от релейных вычислительных машин к машинам на электронно-вакуумных лампах, которые стали элементарной базой вычислительных машин первого поколения.

Первая ЭВМ создавалась в 1943 - 1946 гг. Самой знаменитой была машина созданная в США и называлась она ENIAC (электронный цифровой интегратор и вычислитель). Эта машина содержала около 18 тысяч электронных ламп, множество электромеханических реле. Ее создателями были Дж. Моучли.

ЭВМ первого поколения располагались в огромных машинных залах, потребляли много электроэнергии и требовали остывания с помощью массивных вентиляторов. Программы для этих ЭВМ необходимо было составлять в машинных кодах, и этим могли заниматься лишь мастера, понимающие в деталях устройство ЭВМ.

В 1945 году известный математик и физик - теоретик фон Нейман определил общие принципы работы универсальных вычислительных устройств. Согласно фон Нейману вычислительная машина обязана была управляться программой с последовательным выполнением команд, а сама программа - храниться в памяти машины. Первая ЭВМ с хранимой в памяти программой была построена в Англии в 1949 г.

В СССР созданием компьютеров занимался академик С.А. Лебедев. Его машины БЭСМ — 1, БЭСМ-3М, БЭСМ-4, М- 220 были признаны лучшими в мире.

ЭВМ постоянно совершенствовались, благодаря чему к середине 50-х годов их быстродействие удалось повысить от нескольких сотен до нескольких десятков тысяч операций в секунду. Но при этом электронная лампа оставалась самым надежным элементом ЭВМ. На смену лампам пришли полупроводниковые приборы, тем самым завершился первый этап развития ЭВМ. Вычислительные машины этого этапа принято именовать ЭВМ первого поколения

Таким образом, машины первого поколения имели внушительные размеры, потребляли огромную мощность, имели сравнимо маленькое быстродействие, малую емкость оперативной памяти, невысокую надежность работы и недостаточно развитое программное обеспечение (ПО). В ЭВМ этого поколения были заложены базы логического построения машин и продемонстрированы способности цифровой вычислительной техники. Но использование в качестве элементной базы электронно-вакуумных ламп тормозило развитие и совершенствование ЭВМ, ограничивало область их применения. Они использовались в основном для инженерных и научных расчетов, не связанных с переработкой больших объемов информации.

В таблице 1.1 приведена краткая характеристика ЭВМ I-го поколения:

Таблица 1.1. ЭВМ первого поколения

Характеристики	I поколение
Годы	1945- 1958 гг.
Элементная база	Электронно-вакуумные лампы
Размер (габариты)	Громоздкие сооружения, занимавшие сотни квадратных метров, потреблявшие сотни киловатт электроэнергии и содержащие в себе тысячи ламп.
Максимальное быстродействие процессора	От нескольких сотен до нескольких десятков тысяч операций в секунду.
Максимальный объем ОЗУ	Несколько тысяч команд программы
Периферийные	Перфоленты и перфокарты
Программное обеспечение	Программы составлялись на языке машинных команд, поэтому программирование было доступно не всем. Существовали библиотеки стандартных программ.
Области применения	Инженерные и научные расчеты, не связанные с переработкой больших объемов информации
Примеры	Mark I, ENIAC, БЭСМ.

Второе поколение ЭВМ

Создатели ЭВМ постоянно следовали за прогрессом в электронной технике. В 1949 году в США был создан транзистор — первый полупроводниковый прибор, заменивший электронную лампу. Транзисторы были компактнее, имели большой срок службы, значительно меньше потребляли электроэнергию, выделяли меньше тепла при работе. С внедрением цифровых элементов на полупроводниковых устройствах началось создание ЭВМ второго поколения. Благодаря применению более совершенной элементной базы начали создаваться относительно небольшие ЭВМ, вышло естественное разделение вычислительных машин на большие, средние и малые.

В СССР были разработаны и обширно использовались серии малых. Рекордной посреди российских машин этого поколения и одной из наилучших в мире была БЭСМ-6 («большая электронно - счетная машина»), которая была создана коллективом академика С.А. Лебедева. Производительность БЭСМ-6 была на два — три порядка выше, чем у малых и средних ЭВМ, и составляла более 1 млн. операций в секунду. За рубежом более распространенными машинами второго поколения были «Элиот» (Англия), «Сименс» (ФРГ), «Стретч» (США).

Одновременно с развитием ЭВМ развивались и периферийные устройства: внешняя память на магнитных барабанах и лентах. Совершенствовались языки программирования, появились языки высокого уровня: Фортран, Алгол, Кобол. Программы и программирование стало проще, понятнее и доступнее. Расширилась область применения, стали создаваться электронно-справочные и информационные системы.

В ниже приведенной таблице 1.2 приведена краткая характеристика ЭВМ II-го поколения:

Таблица 1.2. ЭВМ второго поколения

Характеристики	II поколение
Годы	1959 — 1963 гг.
Элементная база	Транзисторы
Размер (габариты)	Стали компактнее, надежнее, менее энергоемкие
Максимальное быстродействие процессора	Десятки и сотни тысяч операций в секунду
Максимальный объем	Увеличился в сотни раз

ОЗУ	
Периферийные	Внешняя память на магнитных барабанах и лентах
Программное обеспечение	Программы и программирование стало проще, понятнее и доступнее. Стали развиваться языки высокого уровня программирования.
Области применения	Создание информационно-справочных и информационных систем
Примеры	М-220, Мир, БЭСМ-4, IBM-7094

ЭВМ третьего поколения

Революцию технологии производства ЭВМ вызвало создание интегральных схем, на которых транзисторы, конденсаторы и резисторы собрались в едином объеме полупроводника. Это произошло в конце 30-х годов XX века. Операция изготовления интегральных схем все время совершенствовалась, и в результате на одной кремневой пластинке стало возможным разместить сотни кристаллов интегральных схем. Произошел переход к третьему поколению ЭВМ.

Применение интегральных микросхем позволило увеличить количество электронных частей в компьютере без роста их настоящих размеров. Быстродействие ЭВМ возросло до 10 миллионов операций в секунду. Не считая этого, составлять программы для ЭВМ стало по силам обычным пользователям, а не только специалистам — электронщикам. При проектировании процессора стали использовать технику микропрограммирования — составление сложных команд процессора из простых.

В машинах третьего поколения в качестве средства общения стали использоваться видеотерминальные устройства — дисплеи. В третьем поколении возникли крупные серии ЭВМ, различающиеся собственной производительностью и назначением. Это семейство больших и средних машин IBM360/370, разработанных в США. В приведенной ниже таблице 1.3 приведена краткая характеристика ЭВМ III-го поколения:

Таблица 1.3. ЭВМ третьего поколения

Характеристики	III поколение
Годы	1964 — 1976 гг.
Элементная база	Интегральные схемы
Размер (габариты)	ЭВМ делятся на большие, средние, мини и

	микро
Максимальное быстродействие процессора	До10 миллионов операций в секунду
Максимальный объем ОЗУ	До 16 Мбайт. Появляются ПЗУ
Периферийные	Внешняя память на магнитных дискетах, дисплей.
Программное обеспечение	Появились операционные системы и множество прикладных программ. Многопрограммный режим — возможность выполнять несколько программ одновременно.
Области применения	Базы данных, первые системы искусственного интеллекта, системы автоматизированного управления и проектирования
Примеры	БЭСМ-6, IBM/360

ЭВМ четвертого поколения

Новые технологии создания интегральных схем в конце 70-х — начале 80-х годов XX века позволили разработать большие интегральные схемы — БИС.

Технология производства БИС постоянно совершенствовалась, это привело к созданию сверхбольших интегральных схем (СБИС) с памятью 1 Мбайт. СБИС позволили создать микропроцессор, который произвел очередную революцию в мире вычислительной техники и привел к появлению ЭВМ четвертого поколения. Микропроцессор способен выполнять функции основного блока компьютера — процессора. Он работает по заложенной в него программе и может встраиваться в различные технические устройства.

Одним из революционных достижений в области вычислительной техники явилось создание персональных ЭВМ, которые можно отнести к отдельному классу машин четвертого поколения. Именно с этого момента в нашем языке вместо «ЭВМ» утвердился термин «персональный компьютер» — ПК.

Сегодня ПК пользуются такой популярностью, что становятся такой же привычной бытовой техникой, как и телевизор или магнитофон.

В нижеприведенной таблице 1.4 приведена краткая характеристика ЭВМ IV-го поколения:

Таблица 1.4. ЭВМ четвертого поколения

Характеристики	IV поколение
Годы	1977 — 1990 гг
Элементная база	БИС и СБИС
Размер (габариты)	Микро ЭВМ — малые габариты, сравнимые с размерами бытового телевизора; суперкомпьютеры — состоят из отдельных блоков и центрального процессора
Максимальное быстродействие процессора	От 2,5 МГц и больше
Максимальный объем ОЗУ	От 16 Мбайт и больше
Периферийные	Цветной графический дисплей, манипуляторы типа «мышь», «джойстик», клавиатура, магнитные и оптические диски, принтеры и пр.
Программное обеспечение	Пакеты прикладного, сетевого, мультимедиа программного обеспе-чения
Области применения	Все сферы научной, производственной и учебной деятельности, отдых и развлечение, Интернет
Примеры	IBM PC, Macintosh, Gray, Эльбрус

ЭВМ пятого поколения

Конец 90-х превратился в настоящую гонку конкурирующих титанов — производителей компьютерной техники. Стремительно повышается тактовая частота процессоров и их модификации. Возрастающая скорость работы процессоров стимулировала совершенствование других узлов и периферийных устройств компьютерного «железа». Некоторые специалисты считают, что в 90-х годах XX века появился компьютер V поколения, представляющий собой: ЭВМ на сверхсложных микропроцессорах с параллельно-векторной структурой, одновременно выполняющих десятки последовательных команд программы, что позволяет строить эффективные системы обработки знаний.

5-е поколение, 90-е гг: ЭВМ с многими десятками параллельно работающих микропроцессоров. Примерная характеристика компьютеров пятого поколения приведена в таблице 1.5.

Таблица 1.5. ЭВМ пятого поколения

Характеристики	V поколение
Годы	1990 — наши дни
Элементная база	ЭВМ на сверхсложных микропроцессорах с параллельно-векторной структурой, одновременно выполняющих десятки последовательных команд программы; многоядерность
Размер (габариты)	Появление карманных компьютеров
Максимальное быстродействие процессора	от 4 ГГц и больше
Максимальный объем ОЗУ	от 2000Mb и выше
Периферийные	Картридер, flash-память, геймпады, многофункциональные устройства
Программное обеспечение	Развитие существующих пакетов прикладного, сетевого, мультимедиа и пр. программного обеспечения
Области применения	Расширение сферы научной, производственной и учебной _деятельности, отдых и развлечение, Интернет
Примеры	Pentium 4, Athlon

1.4. Классификация современных вычислительных машин и систем

Персональные компьютеры

Персональной стала называться однопользовательская Микро ЭВМ. Характеристики ПЭВМ:

- невысокая стоимость;
- наличие периферийных устройств, необходимых для ввода-вывода и хранения информации;
- наличие аппаратных ресурсов для решения реальных задач;
- поддержка языков программирования высокого уровня;
- наличие операционной системы, которая облегчает взаимодействие пользователя с ЭВМ;
- «дружественность» по отношению к пользователю;

Годом рождения ПЭВМ считается 1976 — год создания ПЭВМ Apple, а началом массового выпуска — год создания фирмы Apple Computer Inc., специально организованной в США для производства

ПЭВМ. Создатели этой фирмы Стив Джобс — специалист по электронным играм, и Стив Возняк — специалист по Микро ЭВМ. Первая ПЭВМ Apple была ориентирована в основном на игровое применение и имела цветной монитор и синтезатор звукового сопровождения.

В августе 1981 года мир облетело сообщение, последствие которого в то время едва ли кто-нибудь мог предвидеть и оценить по достоинству: «Нью-Йорк. 12 августа. Корпорация IBM (International Business Machine) сегодня объявила о выпуске своей самой компактной и недорогой компьютерной системы — IBM Personal Computer. Сконструированная специально для применения в бизнесе, школе и дома, эта простая в использовании система продается по цене всего лишь 1565 долларов...». Новизна этого сообщения состояла прежде всего в том, что тогда впервые было произнесено словосочетание «персональный компьютер», это была совершенно новая концепция ЭВМ. Менее чем за один год корпорации IBM удалось стремительно наладит производство и сбыт своих PC. Изготовители микрокомпьютеров очень скоро признали безусловным лидером именно компьютер корпорации IBM. ПЭВМ стала персональным инструментом делового человека.

Важным фактором успеха IBM PC стала так называемая «открытая архитектура», позволяющая другим фирмам приложить свои творческие и предпринимательские способности в его пополнении все новыми и новыми возможностями и программами, и тем самым способствовать утверждению персонального компьютера в качестве мирового стандарта. Появились сотни фирм, выпускающих машины, полностью совместимые с IBM PC, или предлагающих для него дополнительные устройства.

Рабочие станции

Рабочей станцией называется совокупность аппаратных и программных средств, предназначенных для решения профессиональных задач. Это специализированный высокопроизводительный компьютер для тех, кому необходима надежная и производительная система, гарантирующая стабильную и эффективную работу приложений. Использование рабочих станций позволяет вывести ваше предприятие на новый профессиональный уровень вне зависимости от того, в какой области вы развиваетесь.

Рабочие станции решают широкий спектр задач:

- инженерно-технические задачи — 3D-проектирование и конструирование, расчетные работы;

- профессиональная работа с трехмерной графикой — визуализация, 3D-моделирование, мультипликация, спецэффекты;
- цифровая обработка фото и видео материала - верстка, монтаж, дизайн;
- работа с большими объемами данных — статистика, аналитика, прогнозирование.

Основные преимущества:

- эффективность. Решения, использующие последние технологии, позволяют рабочим станциям более эффективно справиться с высокими вычислительными нагрузками. Рабочие станции адаптированы на решение профессиональных задач за счет оптимизации как аппаратной части, так и драйверов;
- надежность. Повышенная надежность достигается за счет использования только высококачественной компонентной базы, длительному стресс-тестированию на этапе разработки и тотальному контролю качества при производстве изделия;
- специализация. Отдельным сегментом в линейке рабочих станций являются графические станции, оснащаемые профессиональными видеоадаптерами, созданными специально для решения профессиональных задач, связанных со сложной визуализацией, конструированием и 3D-моделированием, разработкой и производством, созданием медиа контента и научной деятельностью;
- адаптация к программному обеспечению. Графические станции проходят тестирование и сертифицирование на совместимость и эффективную работу с приложениями от ведущих разработчиков профессионального профильного программного обеспечения, таких как Catia и SolidWorks от Dassault Systemes, AutoCAD и Inventor от Autodesk, Компас 3D от Аскон, ProEngineer от ProTechnologies, NX от Siemens PLM Software, с продуктами компаний ANSYS, Adobe и многих других;
- возможности расширения. Платформы рабочих станций предоставляют большую гибкость в модернизации. Большее количество слотов PCI и PCI-E дает возможность установки профильных плат расширения. Большее количество слотов памяти и возможность установки второго процессора в двухпроцессорных системах увеличивает диапазон выбора производительности.

X-терминалы

X-терминалы представляют собой комбинацию бездисковых рабочих станций и стандартных ASCII-терминалов. Бездисковые рабочие станции часто применялись в качестве дорогих дисплеев и в этом случае не полностью использовали локальную вычислительную мощь. Одновременно многие пользователи ASCII-терминалов хотели улучшить их характеристики, чтобы получить возможность работы в многооконной системе. Совсем недавно, как только стали доступными очень мощные графические рабочие станции, появилась тенденция применения "подчиненных" X-терминалов, которые используют рабочую станцию в качестве локального сервера.

На компьютерном рынке X-терминалы занимают промежуточное положение между персональными компьютерами и рабочими станциями. Поставщики X-терминалов заявляют, что их изделия более эффективны в стоимостном выражении, чем рабочие станции высокого ценового класса, и предлагают увеличенный уровень производительности по сравнению с персональными компьютерами. Быстрое снижение цен, прогнозируемое иногда в секторе X-терминалов, в настоящее время идет, очевидно, благодаря обострившейся конкуренции в этом секторе рынка. Многие компании начали активно конкурировать за распределение рынка, а быстрый рост объемных поставок создал предпосылки для создания такого рынка.

Как правило, стоимость X-терминалов составляет около половины стоимости сравнимой по конфигурации бездисковой машины и примерно четверть стоимости полностью оснащенной рабочей станции.

Типовой X-терминал включает следующие элементы:

- экран высокого разрешения: обычно размером от 14 до 21 дюйма по диагонали;
- микропроцессор на базе Motorola 68xxx или RISC-процессор типа Intel i960, MIPS R3000 или AMD29000;
- отдельный графический сопроцессор в дополнение к основному процессору, поддерживающий двухпроцессорную архитектуру, которая обеспечивает более быстрое рисование на экране и прокручивание экрана;
- базовые системные программы, на которых работает система X-Windows и выполняются сетевые протоколы;
- программное обеспечение сервера X11;

- переменный объем локальной памяти (от 2 до 8 Мбайт) для дисплея, сетевого интерфейса, поддерживающего TCP/IP и другие сетевые протоколы;
- порты для подключения клавиатуры и мыши;

X-терминалы отличаются от ПК и рабочих станций не только тем, что не выполняют функции обычной локальной обработки. Работа X-терминалов зависит от главной (хост) системы, к которой они подключены посредством сети. Для того чтобы X-терминал мог работать, пользователи должны инсталлировать программное обеспечение многооконного сервера X11 на главном процессоре, выполняющем прикладную задачу (наиболее известная версия — X11 Release 5). X-терминалы отличаются также от стандартных алфавитно-цифровых ASCII и традиционных графических дисплейных терминалов тем, что они могут быть подключены к любой главной системе, которая поддерживает стандарт X-Windows. Более того, локальная вычислительная мощь X-терминала обычно используется для обработки отображения, а не обработки приложений (называемых клиентами), которые выполняются удаленно на главном компьютере (сервере). Вывод такого удаленного приложения просто отображается на экране X-терминала.

Серверы

Сервером называется компьютер, выделенный из группы персональных компьютеров (или рабочих станций) для выполнения какой-либо сервисной задачи без непосредственного участия человека. Сервер и рабочая станция могут иметь одинаковую аппаратную конфигурацию, так как различаются лишь по участию в своей работе человека за консолью.

Некоторые сервисные задачи могут выполняться на рабочей станции параллельно с работой пользователя. Такую рабочую станцию условно называют невыделенным сервером.

Консоль (обычно — монитор/клавиатура/мышь) и участие человека необходимы серверам только на стадии первичной настройки, при аппаратно-техническом обслуживании и управлении в нештатных ситуациях (штатно, большинство серверов управляются удаленно). Для нештатных ситуаций серверы обычно обеспечиваются одним консольным комплектом на группу серверов (с коммутатором, например KVM-переключателем, или без такового).

В результате специализации (см. ниже), серверное решение может получить консоль в упрощенном виде (например, коммуникационный порт), или потерять её вовсе (в этом

случае первичная настройка и нештатное управление могут выполняться только через сеть, а сетевые настройки могут быть сброшены в состояние по умолчанию).

В зависимости от задач и условий использования, сервер имеет следующие основные характеристики:

1. Производительность — является основной характеристикой сервера, которая зависит от его аппаратной конфигурации.

Для повышения производительности серверов применяются технологии, основанные на последних достижениях в области компьютерной техники. Например:

- четыре процессорных разъема на одной материнской плате;
- многоканальный режим работы оперативной памяти;
- независимые шины PCI-Express x16;
- жесткие диски с интерфейсом SAS и высокой скоростью вращения шпинделя (10000-15000 об/мин);
- объединение жестких дисков в RAID-массивы.

Производительность сервера также можно увеличить при помощи построения подсистем памяти и ввода-вывода, максимально эффективно использующих возможности архитектуры процессоров.

2. Надежность. Серверное оборудование зачастую предназначено для обеспечения работы сервисов в режиме 24/7 (или «пять девяток» — 99,999 %), поэтому оно часто комплектуется дублирующими элементами. Время недоступности сервера составляет менее 6 минут в год. Для этого конструкторами при создании серверов разрабатываются специальные решения, отличные от принципов построения обычных компьютеров:
   - память обеспечивает повышенную устойчивость к сбоям. Например, для i386-совместимых серверов, модули оперативной памяти и кэша имеют усиленную технологию коррекции ошибок (англ. Error Checking and Correction, ECC). На некоторых других платформах, например SPARC (Sun Microsystems), коррекцию ошибок имеет вся память. Для собственных мэйнфреймов IBM разработала специальную технологию Chipkill;
   - повышение надёжности сервера достигается резервированием, в том числе с горячими подключением и заменой (англ. Hot-swap) критически важных компонентов;
   - при необходимости вводится дублирование процессоров (например, это важно для непрерывности выполнения сервером задачи долговременного расчёта — в случае отказа одного процессора вычисления не обрываются, а

продолжаются, пусть и на меньшей скорости), блоков питания, жёстких дисков в составе массива RAID и самих контроллеров дисков; групп вентиляторов, обеспечивающих охлаждение компонентов сервера.

Существует два вида надежности:

- физическая надежность — стабильность работы достигается надежными комплектующими и качественной сборкой в целом.
- аппаратная надежность — отсутствие программных сбоев за счет стабильности работы аппаратной части.

3. Масштабируемость — это возможность увеличить вычислительную мощность сервера или операционной системы (в частности, их способность выполнять больше операций или транзакций за определенный период времени, либо запускать больше различных служб) за счет установки большего числа процессоров, оперативной памяти и т.д. или их замены на более производительное оборудование. Это масштабируемость аппаратная. За программную масштабируемость функционала (служб) отвечает программное обеспечение, то есть серверная операционная система.
4. Управляемость. Возможность удаленного мониторинга состояния, удаленное включение и перезагрузка сервера. Возможность удаленной диагностики сервера даже в выключенном состоянии, а также работа системы прогнозирования неполадок.

Мейнфреймы

Мейнфре́йм (также мэйнфрейм, от англ. mainframe) — большой универсальный высокопроизводительный отказоустойчивый сервер со значительными ресурсами ввода-вывода, большим объёмом оперативной и внешней памяти, предназначенный для использования в критически важных системах (англ. mission-critical) с интенсивной пакетной и оперативной транзакционной обработкой.

Основной разработчик мейнфреймов — корпорация IBM. Самые известные мейнфреймы выпущены ею в рамках продуктовых линеек System/360, 370, 390, zSeries. В разное время мейнфреймы производили Hitachi, Bull, Unisys, DEC, Honeywell, Burroughs, Siemens, Amdahl, Fujitsu, в странах СЭВ выпускались мейнфреймы ЕС ЭВМ.

Особенности и характеристики современных мейнфреймов:

1. Среднее время наработки на отказ.
   Время наработки на отказ современных мейнфреймов оценивается в 12-15 лет. Надёжность мейнфреймов — это

результат их почти 60-летнего совершенствования. Группа разработки операционной системы VM/ESA затратила 20 лет на удаление ошибок, и в результате была создана система, которую можно использовать в самых ответственных случаях.

2. Повышенная устойчивость систем.
   Мейнфреймы могут изолировать и исправлять большинство аппаратных и программных ошибок за счёт использования следующих принципов:
   - дублирование: Два резервных процессора, резервные модули памяти, альтернативные пути доступа к периферийным устройствам;
   - горячая замена всех элементов вплоть до каналов, плат памяти и центральных процессоров;
   - целостность данных. В мейнфреймах используется память с коррекцией ошибок. Ошибки не приводят к разрушению данных в памяти, или данных, ожидающих вывода на внешние устройства. Дисковые подсистемы построенные на основе RAID-массивов с горячей заменой и встроенных средств резервного копирования защищают от потерь данных.
3. Рабочая нагрузка. Рабочая нагрузка мейнфреймов может составлять 80-95 % от их пиковой производительности. Операционная система мейнфрейма будет обрабатывать всё сразу, причём все приложения будут тесно сотрудничать и использовать общие компоненты ПО.
4. Пропускная способность. Подсистемы ввода-вывода мейнфреймов разработаны так, чтобы работать в среде с высочайшей рабочей нагрузкой на ввод-вывод данных.
5. Масштабирование. Масштабирование мейнфреймов может быть как вертикальным, так и горизонтальным. Вертикальное масштабирование обеспечивается линейкой процессоров с производительностью от 5 до 200 MIPS и наращиванием до 12 центральных процессоров в одном компьютере. Горизонтальное масштабирование реализуется объединением ЭВМ в Sysplex (System Complex) — многомашинный кластер, выглядящий с точки зрения пользователя единым компьютером. Всего в Sysplex можно объединить до 32 машин. Географически распределённый Sysplex называют GDPS. В случае использования операционной системы VM для совместной работы можно объединить любое количество компьютеров. Программное масштабирование заключается в том, что на одном мейнфрейме может быть сконфигурировано фактически бесконечное число различных

серверов. Причем все серверы могут быть изолированы друг от друга так, как будто они выполняются на отдельных выделенных компьютерах и, в то же время, совместно использовать аппаратные и программные ресурсы и данные.

6. Доступ к данным. Поскольку данные хранятся на одном сервере, прикладные программы не нуждаются в сборе исходной информации из множества источников, не требуется дополнительное дисковое пространство для их временного хранения, не возникают сомнения в их актуальности. Требуется небольшое количество физических серверов и значительно более простое программное обеспечение. Всё это, в совокупности, ведёт к повышению скорости и эффективности обработки.
7. Защита. Встроенные в аппаратуру возможности защиты, такие как криптографические устройства, и Logical Partition, и средства защиты операционных систем, дополненные программными продуктами RACF или VM:SECURE, обеспечивают надёжную защиту.
8. Пользовательский интерфейс. Пользовательский интерфейс у мейнфреймов всегда оставался наиболее слабым местом. Сейчас же стало возможно для прикладных программ мейнфреймов в кратчайшие сроки и при минимальных затратах обеспечить современный веб-интерфейс.
9. Сохранение инвестиций. Использование данных и существующих прикладных программ не влечёт дополнительных расходов по приобретению нового программного обеспечения для другой платформы, переучиванию персонала, переноса данных и т. д.

Минисуперкомпьютеры

Минисуперкомпьютеры составляют класс компьютеров, появившийся в середине 1980-х годов. Так как использование векторных процессоров для научных расчётов становилось более популярным, необходимость в менее дорогих системах, которые могли бы использоваться на уровне подразделений, а не на уровне предприятий, создала удачную возможность для появления новых производителей на рынке. В общем, целевая цена этих малых компьютеров составляла 1/10 от цены больших суперкомпьютеров. Эти компьютерные системы характеризовались комбинированием векторной и многопроцессорной (небольшое количество процессоров) обработки данных.

Появление ещё более дешёвых научных рабочих станций на основе микропроцессоров с высокопроизводительными блоками

вычислений с плавающей запятой (FPU) в 1990-х годах (таких как MIPS R8000 и IBM POWER2) ослабила спрос на этот класс компьютеров.

Производители минисуперкомпьютеров:

- Alliant Computer Systems (англ.) (основана в 1982 году под названием Dataflow Systems; обанкротилась в 1992 году);
- Convex Computer (англ.) (основана в 1982 году под названием Parsec; куплена компанией Hewlett-Packard в 1995 году);
- Cydrome (основана в 1984 году под названием Axiom Systems; закрылась в 1988 году);
- Elxsi (основана в 1979 году);
- Encore Computer (основана в 1983 году; куплена компанией Systems Engineering Laboratories);
- Floating Point Systems (основана в 1970 году; куплена компанией Cray Research в 1991 году);
- Multiflow Computer (основана в 1984 году; прекратила деятельность в 1990 году);
- Scientific Computer Systems (основана в 1983 году; в 1989 году переключилась на разработку высокоскоростного сетевого оборудования; ныне не существует);
- Supertek Computers (основана в 1985 году; куплена компанией Cray Research в 1990 году).

Суперкомпьютеры

Суперкомпью́тер (англ. supercomputer, СуперЭВМ) — вычислительная машина, значительно превосходящая по своим техническим параметрам большинство существующих компьютеров. Как правило, современные суперкомпьютеры представляют собой большое число высокопроизводительных серверных компьютеров, соединённых друг с другом локальной высокоскоростной магистралью для достижения максимальной производительности в рамках подхода распараллеливания вычислительной задачи.

Суперкомпьютеры используются во всех сферах, где для решения задачи применяется численное моделирование; там, где требуется огромный объём сложных вычислений, обработка большого количества данных в реальном времени, или решение задачи может быть найдено простым перебором множества значений множества исходных параметров.

Совершенствование методов численного моделирования происходило одновременно с совершенствованием вычислительных машин: чем сложнее были задачи, тем выше были требования к

создаваемым машинам; чем быстрее были машины, тем сложнее были задачи, которые на них можно было решать. Поначалу суперкомпьютеры применялись почти исключительно для оборонных задач: расчёты по ядерному и термоядерному оружию, ядерным реакторам. Потом, по мере совершенствования математического аппарата численного моделирования, развития знаний в других сферах науки — суперкомпьютеры стали применяться и в «мирных» расчётах, создавая новые научные дисциплины, например: численный прогноз погоды, вычислительная биология и медицина, вычислительная химия, вычислительная гидродинамика, вычислительная лингвистика и проч., — где достижения информатики сливались с достижениями прикладной науки.

Ниже приведён далеко не полный список областей применения суперкомпьютеров:

1. математические проблемы:
   - криптография;
   - статистика;
2. физика высоких энергий:
   - процессы внутри атомного ядра, физика плазмы, анализ данных экспериментов, проведенных на ускорителях;
   - разработка и совершенствование атомного и термоядерного оружия, управление ядерным арсеналом, моделирование ядерных испытаний;
   - моделирование жизненного цикла ядерных топливных элементов, проекты ядерных и термоядерных реакторов;
3. наука о Земле:
   - прогноз погоды, состояния морей и океанов;
   - предсказание климатических изменений и их последствий;
   - исследование процессов, происходящих в земной коре, для предсказания землетрясений и извержений вулканов;
   - анализ данных геологической разведки для поиска и оценки нефтяных и газовых месторождений, моделирование процесса выработки месторождений;
   - моделирование растекания рек во время паводка, растекания нефти во время аварий;
4. вычислительная биология: фолдинг белка, расшифровка ДНК;
5. вычислительная химия и медицина: поиск и создание новых лекарств;
6. физика:

- газодинамика: турбины электростанций, горение топлива, аэродинамические процессы для создания совершенных форм крыла, фюзеляжей самолетов, ракет, кузовов автомобилей;
- гидродинамика: течение жидкостей по трубам, по руслам рек;
- материаловедение: создание новых материалов с заданными свойствами, анализ распределения динамических нагрузок в конструкциях, моделирование крэш-тестов при конструировании автомобилей.

Производительность суперкомпьютеров чаще всего оценивается и выражается в количестве операций с плавающей точкой в секунду (flops). Это связано с тем, что задачи численного моделирования, под которые и создаются суперкомпьютеры, чаще всего требуют вычислений, связанных с вещественными числами с высокой степенью точности, а не с целыми числами. Поэтому для суперкомпьютеров неприменима мера быстродействия обычных компьютерных систем — количество миллионов операций в секунду (MIPS). При всей своей неоднозначности и приблизительности, оценка в флопсах позволяет легко сравнивать суперкомпьютерные системы друг с другом, опираясь на объективный критерий.

Первые суперкомпьютеры имели производительность порядка 1 кфлопс, т.е. 1000 операций с плавающей точкой в секунду. Компьютер CDC 6600, имевший производительность в 1 миллион флопсов (1 Мфлопс) был создан в 1964 году. Планка в 1 миллиард флопс (1 Гигафлопс) была преодолена суперкомпьютером NEC SX-2 в 1983 с результатом 1.3 Гфлопс. Граница в 1 триллион флопс (1 Тфлопс) была достигнута в 1996 году суперкомпьютером ASCI Red. Рубеж 1 квадриллион флопс (1 Петафлопс) был взят в 2008 году суперкомпьютером IBM Roadrunner. В настоящее время ведутся работы по созданию в 2016 году экзафлопсных компьютеров, способных выполнять 1 квинтиллион операций с плавающей точкой в секунду.

Наиболее распространёнными программными средствами суперкомпьютеров, также как и параллельных или распределённых компьютерных систем являются интерфейсы программирования приложений (API) на основе MPI и PVM, и решения на базе открытого программного обеспечения, наподобие Beowulf и openMosix, позволяющего создавать виртуальные суперкомпьютеры даже на базе обыкновенных рабочих станций и персональных компьютеров. Для быстрого подключения новых вычислительных узлов в состав узкоспециализированных кластеров применяются технологии наподобие ZeroConf. Примером может служить реализация рендеринга

в программном обеспечении Shake, распространяемом компанией Apple. Для объединения ресурсов компьютеров, выполняющих программу Shake, достаточно разместить их в общем сегменте локальной вычислительной сети.

В настоящее время границы между суперкомпьютерным и обычным программным обеспечением сильно размыты и продолжают размываться ещё более вместе с проникновением технологий параллелизации и многоядерности в процессорные устройства персональных компьютеров и рабочих станций. Исключительно суперкомпьютерным программным обеспечением сегодня можно назвать лишь специализированные программные средства для управления и мониторинга конкретных типов компьютеров, а также уникальные программные среды, создаваемые в вычислительных центрах под «собственные» уникальные конфигурации суперкомпьютерных систем.

Кластеры

Кластер — группа компьютеров, объединённых высокоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс. Кластер — это слабо связанная совокупность нескольких вычислительных систем, работающих совместно для выполнения общих приложений, и представляющихся пользователю единой системой. Один из первых архитекторов кластерной технологии — Грегори Пфистер, дал кластеру следующее определение:

«Кластер — это разновидность параллельной или распределенной системы, которая:

1. состоит из нескольких связанных между собой компьютеров;
2. используется как единый, унифицированный компьютерный ресурс.

Обычно различают следующие основные виды кластеров:

- отказоустойчивые кластеры (High-availability clusters, кластеры высокой доступности);
- кластеры с балансировкой нагрузки (Load balancing clusters)
- вычислительные кластеры (High perfomance computing clusters, HPC);
- системы распределенных вычислений».

Кластеры высокой доступности обозначаются аббревиатурой HA (англ. High Availability — высокая доступность). Создаются для обеспечения высокой доступности сервиса, предоставляемого кластером. Избыточное число узлов, входящих в кластер, гарантирует

предоставление сервиса в случае отказа одного или нескольких серверов. Типичное число узлов — два, это минимальное количество, приводящее к повышению доступности. Создано множество программных решений для построения такого рода кластеров.

Принцип действия кластеров распределения нагрузки (Network Load Balancing, NLB) строится на распределении запросов через один или несколько входных узлов, которые перенаправляют их на обработку в остальные вычислительные узлы. Первоначальная цель такого кластера — обеспечение высокой производительности, однако, в них часто используются также и методы, повышающие надёжность. Подобные конструкции называются серверными фермами. Программное обеспечение может быть как коммерческим (OpenVMS, MOSIX, Platform LSF HPC, Solaris Cluster, Moab Cluster Suite, Maui Cluster Scheduler), так и свободно распространяемым (OpenMosix, Sun Grid Engine, Linux Virtual Server).

Кластеры используются в вычислительных целях, в частности, в научных исследованиях. Для вычислительных кластеров существенными показателями являются высокая производительность процессора в операциях над числами с плавающей точкой и низкая латентность объединяющей сети, и менее существенными — скорость операций ввода-вывода, которая в большей степени важна для баз данных и web-сервисов. Вычислительные кластеры позволяют уменьшить время расчетов по сравнению с одиночным компьютером, разбивая задание на параллельно выполняющиеся ветки, которые обмениваются данными по связывающей сети.

Системы распределенных вычислений (grid)

Такие системы не принято считать кластерами, но их принципы в значительной степени сходны с кластерной технологией. Их также называют grid-системами (англ. grid — решётка, сеть). Главное отличие — низкая доступность каждого узла, то есть невозможность гарантировать его работу в заданный момент времени (узлы подключаются и отключаются в процессе работы), поэтому задача должна быть разбита на ряд независимых друг от друга процессов. Такая система, в отличие от кластеров, не похожа на единый компьютер, а служит упрощённым средством распределения вычислений. Нестабильность конфигурации в таком случае компенсируется большим числом узлов. Грид-вычисления — это форма распределённых вычислений, в которой «виртуальный суперкомпьютер» представлен в виде соединённых с помощью сети, слабосвязанных гетерогенных компьютеров, работающих вместе для

выполнения огромного количества заданий (операций, работ). Эта технология применяется для решения научных, математических задач, требующих значительных вычислительных ресурсов. Грид-вычисления используются также в коммерческой инфраструктуре для решения таких трудоёмких задач, как экономическое прогнозирование, сейсмоанализ, разработка и изучение свойств новых лекарств.

Грид с точки зрения сетевой организации представляет собой согласованную, открытую и стандартизованную среду, которая обеспечивает гибкое, безопасное, скоординированное разделение вычислительных ресурсов и ресурсов хранения информации, которые являются частью этой среды.

Кластер серверов, организуемых программно

Кластер серверов — это группа серверов, объединённых логически, способных обрабатывать идентичные запросы и использующихся как единый ресурс. Чаще всего серверы группируются посредством локальной сети. Группа серверов обладает большей надежностью и большей производительностью, чем один сервер. Объединение серверов в один ресурс происходит на уровне программных протоколов.

В отличие от аппаратного кластера компьютеров кластеры, организуемые программно, требуют:

- наличия специального программного модуля (Cluster Manager), основной функцией которого является поддержание взаимодействия между всеми серверами — членами кластера;
- синхронизации данных между всеми серверами — членами кластера;
- распределения нагрузки (клиентских запросов) между серверами — членами кластера;
- умения клиентского программного обеспечения распознавать сервер, представляющий собой кластер серверов, и соответствующим образом обрабатывать команды от Cluster Manager. Если клиентская программа не умеет распознавать кластер, она будет работать только с тем сервером, к которому обратилась изначально, а при попытке Cluster Manager перераспределить запрос на другие серверы, клиентская программа может вообще лишиться доступа к этому серверу (результат зависит от конкретной реализации кластера).

В большинстве случаев, кластеры серверов функционируют на раздельных компьютерах. Это позволяет повышать

производительность за счёт распределения нагрузки на аппаратные ресурсы и обеспечивает отказоустойчивость на аппаратном уровне.

Однако, принцип организации кластера серверов (на уровне программного протокола) позволяет исполнять по нескольку программных серверов на одном аппаратном. Такое использование может быть востребовано:

- при разработке и тестировании кластерных решений;
- при необходимости обеспечить доступность кластера только с учётом частых изменений конфигурации серверов — членов кластера, требующих их перезагрузки (причем перезагрузка производится поочерёдно) в условиях ограниченных аппаратных ресурсов.

Метакомпьютеры

Рассмотрим основные свойства, присущие метакомпьютеру.

1. Метакомпьютер обладает огромными ресурсами, которые несравнимы с ресурсами обычных компьютеров. Это касается практически всех параметров: число доступных процессоров, объем памяти, число активных приложений, пользователей и т. п.
2. Метакомпьютер является распределенным по своей природе. Компоненты метакомпьютера могут быть удалены друг от друга на сотни и тысячи километров, что неизбежно вызывает большую латентность и, следовательно, сказывается на оперативности их взаимодействия.
3. Метакомпьютер может динамически менять конфигурацию. Какие-то компьютеры к нему подсоединяются и делегируют права на использование своих ресурсов, какие-то отключаются и становятся недоступными. Но для пользователя работа с метакомпьютером прозрачна. Задача системы поддержки работы метакомпьютера состоит в поиске подходящих ресурсов, проверке их работоспособности, в распределении поступающих задач вне зависимости от текущей конфигурации метакомпьютера в целом.
4. Метакомпьютер неоднороден. При распределении заданий нужно учитывать особенности операционных систем, входящих в его состав. Разные системы поддерживают различные системы команд и форматы представления данных. Различные системы в разное время могут иметь различную загрузку, связь с вычислительными системами идет по каналам с различной пропускной способностью. Наконец, в состав метакомпьютера могут входить системы с принципиально различной архитектурой, начиная с

домашних персональных компьютеров, заканчивая мощнейшими системами.

5. Метакомпьютер объединяет ресурсы различных организаций. Политика доступа и использования конкретных ресурсов может сильно меняться в зависимости от их принадлежности к той или иной организации. Метакомпьютер не принадлежит никому, поэтому политика его администрирования может быть определена лишь в самых общих чертах. Вместе с тем, согласованность работы огромного числа составных частей метакомпьютера предполагает обязательную стандартизацию работы всех его служб и сервисов.

На данный момент в сфере метакомпьютинга решены далеко не все задачи, тем не менее, современное развитие информационных технологий позволяет предположить, что в ближайшие годы в мире появятся полноценные системы GRID.

1.5. История развития суперкомпьютеров

Суперкомпьютеры выполняют поистине титаническую работу — делают сверхточные прогнозы погоды, моделируют сложные физические процессы, помогают в разработке новых удобрений и лекарств. И, конечно же, как и у всего в природе, у эры суперкомпьютеров есть своё начало и свой путь развития. Так как же появились и как развивались эти колоссы компьютерного мира? Чтобы ответить на этот вопрос, нам придётся заглянуть на полстолетия в прошлое, во времена, когда компьютеры уже переставали быть единичным товаром, но до массового их распространения было ещё очень и очень далеко.

Как и в любой истории, в истории суперкомпьютеров не обошлось без своей выдающийся личности. В 1964 году свет увидел первый суперкомпьютер в истории - CDC 6600 производства Control Data Corporation. Руководил работами по созданию этого технического чуда 60-х годов Сеймур Крей, талантливый инженер и негласный "отец суперкомпьютеров". Его CDC 6600 с невиданной по меркам того времени производительностью в 1 МФЛОПС (один миллион операций с плавающей точкой в секунду) вплоть до 1969 года оставался самым мощным и самым совершенным компьютером. В 1969-ом же лавры главного суперкомпьютера планеты перешли к модели CDC 7600 производства всё того же Сеймура Крея. Новая машина была в десять раз производительнее своего предка, что делало её незаменимой для выполнения сложных и трудоёмких расчётов. Однако Крею было

некомфортно в стенах Control Data Corporation. В 1972 году он уходит из CDC, чтобы основать собственную компанию по производству суперкомпьютеров — Cray Research. Уже через четыре года выходит компьютер, который прославил как самого Сеймура Крея, так и его компанию - Cray-1. Это почти шеститонное чудовище с производительностью в 160 МФЛОПС взорвало компьютерный мир того времени и вошло в школьные учебники по информатике. До сих пор именно "Крей-1" приходит нам на ум, когда мы говорим о суперкомпьютерах.

На этом Cray Research не остановилась. Через пять лет был выпущен первый мультипроцессорный суперкомпьютер Cray X-MP, ну а в 1985 году компьютер Cray-2 преодолел планку в один ГФЛОПС, причём сделал это с ну очень серьёзным запасом: почти два миллиарда операций в секунду помогли Cray-2 удержать преимущество на рынке суперкомпьютеров.

Однако не стоит думать, что на этом рынке присутствовала лишь компания Крея. Наоборот, удешевление компьютерной техники в 80-х годах прошлого столетия открыло дорогу в мир суперкомпьютерной индустрии большому числу небольших независимых компаний; как следствие, возросла конкуренция, а, следовательно, скорость прогресса увеличилась. Однако в 1990-х на данном рынке произошёл серьёзный кризис, связанный с изменением геополитической обстановки в мире. Распад СССР и окончание "холодной войны" привели к резкому сокращению заказов от военных инстанций США и стран НАТО. А ведь именно военные были основными покупателями суперкомпьютеров: моделирование ядерных взрывов, расшифровка секретных сообщений вероятного противника и прочие не слишком мирные задачи требовали солидных вычислительных мощностей. Впрочем, вскоре мечи были перекованы на орала, и суперкомпьютеры стали в массовом порядке закупаться различными "мирными" исследовательскими центрами, НИИ и лабораториями.

Перед тем, как приступить к описанию событий, произошедших в мире сверхпроизводительных компьютеров, вернёмся обратно в 60-е, только теперь по нашу сторону "железного занавеса", то есть в СССР. Надо сказать, что тут у нас есть поводы для гордости: по заверениям как отечественных, так и зарубежных исследователей, советская компьютерная промышленность в начале своего развития не отставала от западной, а во многом даже и превосходила её. Однако в рамках данной главы нас интересует не вся компьютерная индустрия СССР, а только её, так сказать, "суперкомпьютерная" часть.

Первым отечественным суперкомпьютером является БЭСМ-6, выпущенный в 1967 году под руководством гениального инженера Сергея Алексеевича Лебедева. Данная машина, по формальной производительности сопоставимая с CDC 6600, реально намного превосходила своего иностранного конкурента. В данном компьютере было заложено так много инновационных решений, что её производство продолжалось на протяжении двадцати лет! Попытка американских инженеров создать что-либо совершеннее БЭСМ-6, носившая имя ILLIAC-IV, окончилась неудачей: данный суперкомпьютер оказалась дороже, сложнее и медленнее "русской машины". БЭСМ-6 не была единственным советским суперкомпьютером. В последние годы своей жизни Лебедев руководил работами по созданию многопроцессорного комплекса "Эльбрус", однако в 1974 году смерть помешала ему увидеть результаты своих трудов. Работы над первым компьютером серии "Эльбрус" завершились в 1979 году, и, хотя по производительности он, равно как и другие компьютеры серии, отставали от зарубежных аналогов, в его процессоре впервые была применена технология суперскалярности. Суперскалярная архитектура, то есть технология параллельного выполнения нескольких команд, независимых друг от друга, вскоре была реализована в большинстве процессоров для персональных компьютеров; таким образом, в процессорах Intel и AMD есть частичка нашего, русского, инженерного знания.

Увы, перестройка, раскол Советского Союза и последовавшие за ним события крайне негативно - если не сказать "губительно", - отразились на отечественной суперкомпьютерной промышленности. Прощальным приветом отечественных инженеров-электронщиков можно считать появившийся в 1990-х процессор Elbrus 2000 (E2K) , который так и не смог выйти на рынок: сначала помешал кризис, ну а затем, когда казалось, что "вот уже чуть-чуть", команду "Эльбруса" на корню купила Intel. На данный момент все существующие в России суперкомпьютеры либо зарубежного производства, либо основаны на зарубежных комплектующих и технологиях.

Вот на такой пессимистичной ноте мы возвращаемся на Запад, чтобы продвинуться далее по временной шкале. Оправившись от кризиса, индустрия производства суперкомпьютеров принялась за штурм новых высот. В 1997 году был создан суперкомпьютер ASCI RED, обладавший неслыханной тогда производительностью в 1,34 ТФЛОПС. Однако самое интересное, что данный компьютер был построен на базе почти десяти тысяч процессоров Pentium II, которые можно было спокойно найти в любом топовом ПК тех лет. Подобная

система объединения вычислительных мощностей относительно недорогих процессоров получила название Massively Parallel Processing, или просто MPP. Преимущество MPP-систем заключается в их гибкости: незагруженные процессорные блоки можно легко отключить, а, по возможности, включить заново и, вдобавок, подключить дополнительные. На данный момент большинство суперкомпьютеров было построено именно на базе данной технологии.

Шло время, и производители выпускали всё более и более новые суперкомпьютеры, которые задавали новые стандарты производительности. Символический барьер в один ПФЛОПС (читается "пентафлопс"; 1 ПФЛОПС = 1000 ТФЛОПС) был преодолён в 2008 году компьютером Roadrunner от IBM. Характеристики данной машины: почти 100 Тб оперативной памяти, около 20 000 процессоров. Причем всё это работает под управлением Linux-систем Red Hat и Fedora, тех же самых версий, что устанавливаются на домашние компьютеры.

Однако самым быстрым в мире суперкомпьютером был признан суперкомпьютер Titan, установленный в Окриджской национальной лаборатории (ORNL) в США. В тестовых испытаниях Linpack его производительность составила 17,59 Пфлопс. В Titan реализована гибридная архитектура CPU-GPU(графический процессор). Система состоит из 18 688 узлов, каждый из которых оснащен 16-ядерным процессором AMD Opteron и графическим ускорителем Nvidia Tesla K20X. В общей сложности используется 560 640 процессоров. Titan представляет собой обновление ранее эксплуатировавшегося в ORNL суперкомпьютера Jaguar и занимает те же серверные шкафы (общей площадью 404 $м^2$).

Возможность использования уже существующих систем питания и охлаждения позволила сэкономить в ходе строительства около 20 млн долларов. Энергопотребление суперкомпьютера составляет 8,2 МВт, что на 1,2 МВт больше показателей Jaguar, при этом его производительность при выполнении операций с плавающей точкой выше почти в 10 раз.

Titan в первую очередь используется для проведения исследований в области науки о материалах и ядерной энергетики, а также исследований, касающихся повышения эффективности работы двигателей внутреннего сгорания. Кроме того, с его помощью будут выполняться моделирование климатических изменений и анализ потенциальных стратегий по устранению связанных с ними негативных последствий.

1.6. Вопросы и задания для самоконтроля

1. Объясните, чем отличается вычислительная машина от вычислительной системы.
2. Перечислите уровни детализации вычислительной машины.
3. Назовите первые счётные устройства. Опишите принцип их работы.
4. Назовите первую машину, выполнявшую все четыре арифметических действия.
5. Перечислите основные характеристики 1-го и 2-го поколения ЭВМ.
6. Перечислите основные характеристики 3-го, 4-го и 5-го поколения ЭВМ.
7. Объясните, какие именно события и каким образом приводили к появлению новых поколений ЭВМ.
8. Кто сформулировал основные принципы работы ЭВМ? Какое общее название они получили? Назовите эти принципы.
9. Назовите критерии производительности, используемые для оценки ЭВМ разного назначения.
10. Назовите первую открытую архитектуру ЭВМ. Расскажите, к чему привело её появление на рынке.
11. Как изменялись со временем области применения ЭВМ?
12. Какие средства ввода-вывода информации были свойственны ЭВМ разных поколений?
13. Что понимается под поколениями ЭВМ? Можно ли определить момент перехода от одного поколения ЭВМ к другому? Можно ли определить фактическую принадлежность ЭВМ к какому-либо из поколений?
14. Опишите, что входит в понятия «персональный компьютер» и «рабочая станция». В чём их отличия?
15. Что представляет из себя X-терминал?
16. Что такое сервер? Опишите основные характеристики этих систем.
17. Перечислите особенности и характеристики современных мейнфреймов.
18. В чём отличие минисуперкомпьютера от суперкомпьютера? Перечислите основные характеристики этих систем.
19. Приведите классификацию кластеров и области их применения.
20. Перечислите основные этапы в развитии суперкомпьютеров: от создания первых систем и до наших дней.

2. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ И СИСТЕМАХ. АРИФМЕТИКА В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

2.1. Представление информации в вычислительных устройствах

В современной вычислительной технике (ВТ) основой представления информации являются электрические сигналы, допускающие в случае использования напряжений постоянного тока две формы представления — аналоговую и дискретную.

В первом случае величина напряжения является аналогом значения некоторой измеряемой переменной, например, подача на вход напряжения в 1.942 в эквивалентна вводу числа 19.42 (при масштабе 0.1). Во втором случае — в виде нескольких различных напряжений, эквивалентных числу единиц в представляемом значении переменной. При аналоговом представлении информации значения измеряемых величин могут принимать любые допустимые значения из заданного диапазона, плавно без разрывов переходя от одного значения к другому. Теоретически, представляется весь бесконечный спектр значений измеряемой величины на заданном отрезке. Таким образом, аналоговые ВМ (АВМ) — вычислительные машины непрерывного действия. Они работают с информацией, представленной в непрерывной (аналоговой) форме, т.е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения).

При дискретном представлении информации значения измеряемых величин носят дискретный (конечный) характер в измеряемом диапазоне. Цифровые ВМ — вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в цифровой форме. Наиболее широкое применение получили цифровые ВМ с электрическим представлением дискретной информации — электронные цифровые вычислительные машины, обычно называемые просто электронными вычислительными машинами, без упоминания об их цифровом характере.

Достоинства аналоговой формы представления информации:

- при создании ВТ аналогового типа требуется меньшее число компонент (ибо одна измеряемая величина представляется одним сигналом);
- аналоговая ВТ более интеллектуальна и производительна за счет возможности легко интегрировать сигнал, выполнять над ним любое функциональное преобразование и т.д.;

- за счет ряда особенностей она позволяет решать ряд классов задач во много раз быстрее, чем дискретная ВТ.

Недостатки аналоговой формы представления информации:

- так как при создании ВТ аналогового типа требуется меньшее число компонент, то сложность ее быстро возрастает за счет необходимости различать значительно большее число (вплоть до бесконечности) состояний сигнала;
- сложность реализации устройств для ее логической обработки, длительного хранения и высокой точности измерения.

Аналоговые вычислительные машины предназначены, в первую очередь, для решения задач, описываемых системами дифференциальных уравнений: управление непрерывными процессами; моделирование в гидро- и аэродинамике; исследование динамики сложных объектов, электромагнитных полей; параметрическая оптимизация и оптимальное управление, и др. Но АВМ не могут решать задач, связанных с хранением и обработкой больших объемов информации различного характера; задач с высокой степенью точности и др., с которыми легко справляются электронные вычислительные машины, использующие дискретную форму представления информации.

Положительные черты обоих типов совмещают гибридные ВМ — вычислительные машины комбинированного действия, работают с информацией, представленной и в цифровой, и в аналоговой форме; они совмещают в себе достоинства АВМ и ЭВМ. Гибридные вычислительные машины целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

С развитием электронно-вычислительной техники большое применение получили двоичная, восьмеричная и шестнадцатеричная системы счисления. Несмотря на то, что десятичная система счисления имеет широкое распространение, ЭВМ строятся на двоичных (цифровых) элементах, так как реализовать элементы с десятью четко различными состояниями сложно.

В двоичной системе счисления используются только две цифры 0 и 1. И значит, имеется только два однозначных числа. Из всех систем счисления особенно проста и поэтому интересна для технической реализации в компьютерах двоичная система счисления. Компьютеры используют двоичную систему потому, что она имеет ряд преимуществ перед другими системами:

- для её реализации нужны технические устройства с двумя устойчивыми состояниями (есть ток — нет тока, намагничен — не намагничен и т.п.), а не с десятью, как в десятичной;
- представление информации посредством только двух состояний надёжно и помехоустойчиво;
- возможно применение аппарата булевой алгебры для выполнения логических преобразований информации;
- двоичная арифметика намного проще десятичной.

2.2. Представление целых чисел. Прямой, обратный и дополнительный коды

Для хранения чисел в памяти компьютера используется два формата: целочисленный (естественная форма) и с плавающей точкой (нормализованная форма) (точка — разделительный знак для целой и дробной частей числа).

Целочисленный формат (формат с фиксированной точкой) используется для представления в компьютере целых (англ. integer) положительных и отрицательных чисел. Для этого, как правило, используются форматы, кратные байту: 1, 2, 4 байта.

В форме с фиксированной запятой числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой (или точки), отделяющей целую часть от дробной.

Эта форма проста и привычна для большинства пользователей, но имеет небольшой диапазон представления чисел и поэтому не всегда пригодна при вычислениях. Если же в результате какой-либо арифметической операции получается число, выходящее за допустимый диапазон, то происходит переполнение разрядной сетки, и все дальнейшие вычисления теряют смысл.

Однобайтовое представление применяется только для положительных целых чисел. В этом формате отсутствует знаковый разряд. Наибольшее двоичное число, которое может быть записано при помощи 1 байта, равно 11111111, что в десятичной системе счисления соответствует числу 25510.

Для положительных и отрицательных целых чисел обычно используется 2 и 4 байта, при этом старший бит выделяется под знак числа: 0 - плюс, 1 - минус.

Самое большое (по модулю) целое число со знаком, которое может поместиться в 2-байтовом формате, это число 0111111111111111, то есть при помощи подобного кодирования можно представить числа от −32 76810 до 32 76710.

Обратите внимание, что если число вышло за указанные границы, произойдет переполнение. Поэтому при работе с большими целыми числами под них выделяется больше места, например 4 байта.

Формат с плавающей точкой (нормализованная форма) используется для представления в компьютере действительных чисел (англ. real). Числа с плавающей точкой размещаются, как правило, в 4 или 8 байтах.

Нормализованная форма представления чисел обеспечивает огромный диапазон их записи и является основной в современных ЭВМ.

Представление целого положительного числа в компьютере

Для представления целого положительного числа в компьютере используется следующее правило:

- число переводится в двоичную систему;
- результат дополняется нулями слева в пределах выбранного формата;
- последний разряд слева является знаковым, в положительном числе он равен 0.

Например, положительное число +13510 в зависимости от формата представления в компьютере будет иметь следующий вид:

- для формата в виде 1 байта - 10000111 (отсутствует знаковый разряд);
- для формата в виде 2 байтов - 0000000010000111;
- для формата в виде 4 байтов - 00000000000000000000000010000111.

Представление целого отрицательного числа в компьютере

Для представления целого отрицательного числа в компьютере используется дополнительный код. Такое представление позволяет заменить операцию вычитания числа операцией сложения с дополнительным кодом этого числа. Знаковый разряд целых отрицательных чисел всегда равен 1.

Для представления целого отрицательного числа в компьютере используется следующее правило:

- число без знака переводится в двоичную систему;
- результат дополняется нулями слева в пределах выбранного формата;
- полученное число переводится в обратный код (нули заменяются единицами, а единицы - нулями);
- к полученному коду прибавляется 1.

Обратный код для положительного двоичного числа совпадает с его прямым кодом, а для отрицательного числа нужно во всех разрядах, кроме знакового, нули заменить единицами и наоборот.

Дополнительный код для положительного числа совпадает с его прямым кодом, а для отрицательного числа образуется путем прибавления 1 к обратному коду.

Отрицательное число может быть представлено в виде 2 или 4 байт. Например, представим число −13510 в 2-байтовом формате:

- 13510 → 10000111 (перевод десятичного числа без знака в двоичный код);
- 0000000010000111 (дополнение двоичного числа нулями слева в пределах формата);
- 0000000010000111 → 1111111101111000 (перевод в обратный код);
- 1111111101111000 → 1111111101111001 (перевод в дополнительный код).

Представление вещественного (действительного) числа в компьютере

Вещественное число может быть представлено в экспоненциальном виде, например:

1600000010=0,16·108

−0,000015610=−0,156·10−4

В этом формате вещественное число (R) представляется в виде произведения мантиссы (m) и основания системы счисления (P) в целой степени (n), называемой **порядком**.

Представим это в общем виде, как: R=m·Pn.

Порядок *n* указывает, на какое количество позиций и в каком направлении должна сместиться в мантиссе точка (запятая), отделяющая дробную часть от целой. Мантисса, как правило, нормализуется, то есть представляется в виде правильной дроби 0< m<1.

Мантисса должна быть правильной дробью, у которой первая цифра после точки (запятой в обычной записи) отлична от нуля. Если это требование выполнено, то число называется **нормализованным**.

При представлении в компьютере действительного числа с плавающей точкой тоже используется нормализованная мантисса и целый порядок. И мантисса и порядок представляются в двоичном виде, как это было описано выше.

Для размещения вещественного числа обычно используется 2 или 4 байта. В 2-байтовом формате представления вещественного числа первый байт и три разряда второго байта выделяются для размещения мантиссы, в остальных разрядах второго байта размещаются порядок числа, знаки числа и порядка.

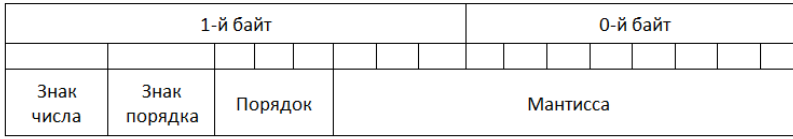


Рис 2.1. Формат представления вещественного числа в ВС

В 4-байтовом формате представления вещественного числа первые три байта выделяются для размещения мантиссы, в четвертом байте размещаются порядок числа, знаки числа и порядка.

Рис 2.2. Формат представления 4-х байтового вещественного числа в вычислительной системе

Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа.

Пример записи числа 6,2510=110,012=0,11001·211, представленного в нормализованном виде, в четырёхбайтовом формате с семью разрядами для записи порядка.

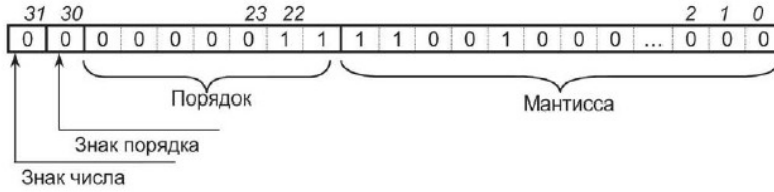


Рис 2.3. Формат представления вещественного числа в нормализованном виде

Основные определения стандарта IEEE-754

Смещенный порядок (biased exponent) — сумма порядка и константы (смещения), выбранной так, что сумма неотрицательна.

Двоичное число с плавающей точкой (binary floating point number) — битовая строка, характеризующаяся тремя составляющими: знаком (sign), знаковым порядком (signed exponent) и мантиссой (significand). Ее численное значение (если оно существует) равно произведению знака, мантиссы и двойки, возведенной в степень, равную порядку.

Денормализованное число (denormalized number) — ненулевое число с плавающей точкой, порядок которого имеет некоторое зарезервированное значение (обычно это минимальное представимое в данном формате число), и чей ведущий (явный или неявный бит) нулевой.

Место назначения (destination) — то, куда записывается результат бинарной или унарной операции. Может быть явно назначено пользователем или неявно — системой (к примеру, промежуточные результаты вычисления выражений). Некоторые языки не позволяют пользователям распоряжаться промежуточными результатами. Тем не менее, стандарт описывает результат операции в терминах целевого формата (destination's format) и значений операндов.

Порядок (exponent) — одна из трех составляющих двоичного числа с плавающей точкой, обычно означающая степень двойки при вычислении значения числа. Иногда порядок называется знаковым (signed) или несмещенным (unbiased). Дробная часть (fraction) — часть мантиссы, лежащая справа от двоичной точки. Не-число (NaN) - символическое значение, кодируемое в плавающем формате. Существует два типа не-чисел.

Сигнализирующие не-числа (signaling NaN) генерируют возникновение исключительной ситуации INVALID в случае, когда они встречаются в качестве операндов. «Тихие» не-числа (quiet NaN) могут без всяких исключений существовать на протяжении почти всех вычислительных операций. Результат (result) — битовая строка (обычно представляющая некоторое число), возвращаемая в место назначения.

Мантисса (significand) — компонента двоичного числа с плавающей точкой, состоящая из явного или неявного ведущего бита слева от подразумеваемой двоичной точки и дробной части справа. Флаг статуса (status flag) — переменная, которая может принимать два значения. Флаг может быть установленным (set), или неустановленным

(clear). Пользователь может очистить флаг, копировать его, или восстановить предыдущее состояние. Будучи установленным, флаг может содержать дополнительную информацию, зависящую от платформы, и, возможно, недоступную некоторым пользователям. Операции, определенные стандартом, могут иметь побочным эффектом выставление следующих флагов: неточный результат (inexact result), исчезновение порядка (underflow), перполнение (overflow), деление на ноль (divide by zero) и неверная операция (invalid operation).

Пользователь (user) — человек, аппаратура или программа, чьи действия не определяются стандартом, имеющий доступ к программному окружению и контролирующий выполнение операций, определяемых стандартом.

Стандарт IEEE-754 на вычисления с плавающей точкой определяет четыре плавающих формата, которые делятся на базовые и расширенные. Уровень соответствия может различаться в зависимости от того, какие форматы реализованы.

Численные значения величин определяются с использованием трех целых параметров:

- p - число значащих битов (precision)
- e_max - максимальное значение порядка
- e_min - минимальное значение порядка

Параметры базовых форматов приведены в таблице:

Формат	p	e_max	e_min	Смещение	Порядок (бит)	Формат (бит)
Single	24	127	-126	127	8	32
Double	53	1023	-1022	1023	11	64

2.3. Арифметика в вычислительных системах

1. При сложении чисел в произвольной позиционной системе счисления с основанием p в каждом разряде производится сложение цифр слагаемых и цифры, переносимой из соседнего младшего разряда, если она имеется. При этом необходимо учитывать, что если при сложении чисел получилось число большее или равное p, то представляем его в виде $pk+b$, где k — частное, а b — остаток от деления полученного числа на основание системы счисления. Число b является количеством единиц в данном разряде, а число k — количеством единиц переноса в следующий разряд.

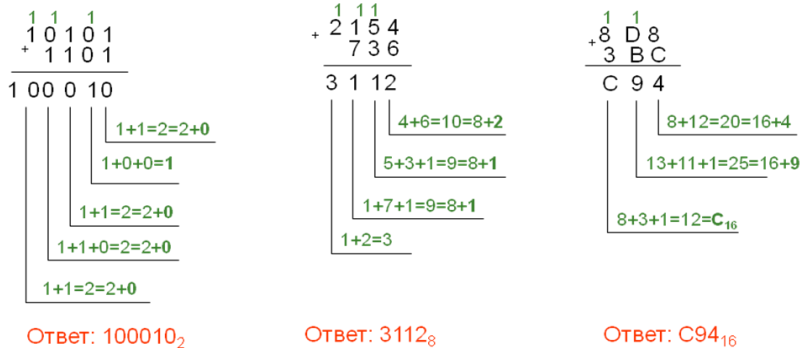


Рис 2.4. Схема сложения

2. При вычитании чисел в *p*-ой системе счисления цифры вычитаются поразрядно. Если в рассматриваемом разряде необходимо от меньшего числа отнять большее, то занимается единица следующего (большего) разряда. Занимаемая единица равна *p* единицам этого разряда.

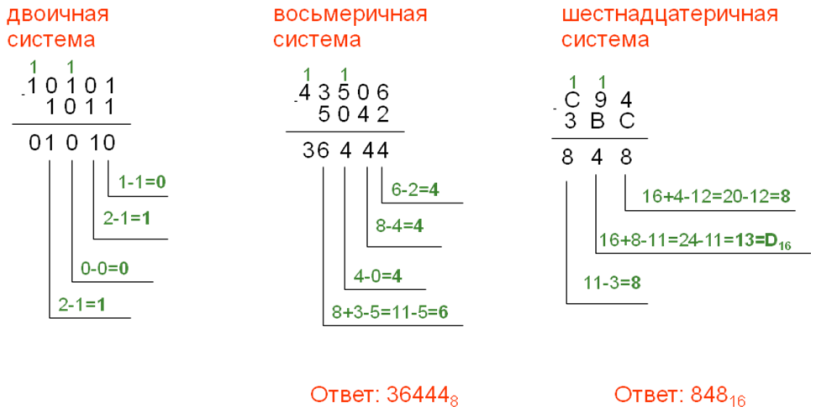


Рис 2.5. Схема вычитания

3. При умножении чисел в *p* –ой системе счисления каждая цифра второго множителя умножается последовательно на цифру каждого из разрядов первого множителя.

При этом необходимо учитывать, что если при сложении чисел получилось число большее или равное *p*, то представляем его в

виде $pk+b$, где k — частное, а b— остаток от деления полученного числа на основание системы счисления. Число b записываем в единицы данного разряда, а число k запоминаем и добавляем его к результату произведения в следующем разряде.

Полученные результаты умножения складываем согласно описанию, представленному в п. 1, и отделяем количество знаков после запятой, равное сумме знаков после запятой у сомножителей.

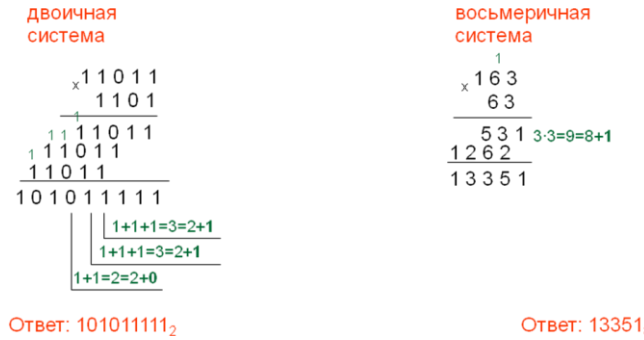


Рис 2.6. Схема умножения

4. Деление чисел в p –ой системе счисления производится так же, как и десятичных чисел, при этом используются правила умножения, сложения и вычитания чисел в p –ой системе счисления (см. пп.1—3).

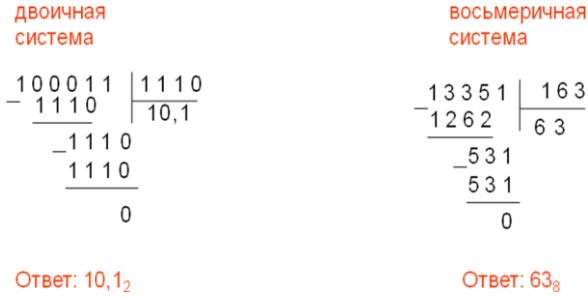


Рис 2.7. Схема деления

2.4. Кодирование текстовых данных

Если каждому символу алфавита поставить в соответствие определенное целое число, то с помощью двоичного кода можно кодировать и текстовую информацию.

8 разрядов — 256 символов: английские и русские буквы строчные и прописные, знаки препинания, арифметических действий и некоторые общепринятые специальные символы (%, №, " и т.д.).

Для того, чтобы весь мир одинаково кодировал текстовые данные, нужны единые таблицы кодирования, а это пока невозможно из-за противоречий между символами национальных алфавитов и противоречий корпоративного характера.

Для английского языка противоречия уже сняты.

Институт стандартизации США (ANSI — American National Standard Institute) ввел в действие систему кодирования ASCII (American Standard Code for Information Interchange — стандартный код информационного обмена США).

ASCII: базовая (0 — 127) и расширенная (128 — 255) таблицы кодирования:

- С 0 по 31 коды отданы производителям аппаратных средств (компьютеров и печатающих устройств), это так называемые управляющие коды, которым не соответствуют никакие символы языка (эти коды не выводятся ни на экран, ни на печать), но они могут управлять выводом других данных;
- 32 — 127: коды символов английского алфавита, цифр и др.

Аналогичные системы кодирования были разработаны и в других странах.

Поддержка производителей оборудования и программ вывела американский код ASCII на уровень международного стандарта, и национальным системам кодирования пришлось отступить на вторую, расширенную часть системы кодирования (128-255 коды). В России:

- Кодировка Windows-1251, введенная компанией Microsoft, — ввиду большого распространения программ этой компании;
- КОИ-8 (код обмена информацией, восьмизначный): произошла в период действия Совета Экономической Взаимопомощи (СЭВ) государств Восточной Европы; имеет широкое распространение в компьютерных сетях на территории России и в российском секторе Интернета.

Международный стандарт, в котором предусмотрена кодировка символов русского алфавита, — кодировка ISO (International Standard Organization — Международный институт стандартизации), на практике используется редко.

На компьютерах, работающих в MS-DOS, могли действовать следующие кодировки: ГОСТ (устаревшая) и ГОСТ-альтернативная (используется и сейчас).

Универсальная система кодирования текстовых данных
Трудности с созданием единой системы кодирования связаны с ограниченным набором кодов (256 — 8 разрядов).

Система, основанная на 16-разрядном кодировании символов, получила название универсальной — Unicode. 16 разрядов позволяют обеспечить уникальные коды для 65 536 различных символов.

Переход на эту систему сдерживался недостаточными ресурсами средств вычислительной техники (в Unicode все текстовые документы автоматически становятся вдвое длиннее), сейчас идет постепенный перевод документов и программных средств на эту систему.

2.5. Кодирование графических данных

Все графические изображения, хранящиеся во внешней памяти компьютера должны быть представлены в числовой форме (т.е. быть закодированными). В зависимости от способа кодирования изображений можно разделить на две большие группы: растровые и векторные.

Растровые

Для кодирования растрового изображения его разбивают на небольшие одноцветные части (пиксели или точки). Все цвета, использованные в изображении, нумеруют, и для каждой части записывают номер ее цвета. Запомнив последовательность расположения частей и номер цвета для каждой части, можно однозначно описать любой рисунок.

Однако, количество цветов в природе бесконечно, и приходится похожие цвета нумеровать одинаковыми числами. В зависимости от количества используемых цветов, можно закодировать достаточно реалистичное изображение. Чем меньше цветов в рисунке, тем меньше чисел приходится использовать при кодировании, и тем проще закодировать изображение. В самом простом случае (черно-белое изображение) для кодировки используется только черный и белый цвет.

Любое растровое изображение характеризуется тремя основными параметрами: глубиной цвета, размером изображения и разрешением. Объем информации, описывающий цвет пикселя, определяет глубину цвета. Чем больше информации отводится для определения цвета пикселя, тем больше оттенков данного цвета существует. Размер изображения описывает его физические габариты,

то есть его высоту и ширину. Можно задать размеры в метрах, миллиметрах, дюймах, пикселях или любых других величинах.

Разрешение — это плотность размещения пикселей, формирующих изображение, то есть количество пикселей на заданном отрезке изображения. Чаще всего разрешение измеряется в количестве точек на дюйм — DPI (Dot Per Inch). Данный способ кодирования изображения несмотря на то, что применяется повсеместно (фотографии и рисунки введенные в компьютер со сканера или фотокамеры, изображения на Web-страницах), обладает одним существенным недостатком. Растровые изображения трудно масштабировать.

При уменьшении растрового изображения несколько соседних точек преобразуются в одну, поэтому теряется разборчивость мелких деталей изображения, а при увеличении масштаба, наоборот, - размер каждого пикселя изображения увеличивается и появляется ступенчатый эффект. Еще одним недостатком растровых изображений является то, что они занимают относительно много места в оперативной и внешней памяти компьютера.

Векторные

Приведенных выше недостатков лишены изображения, закодированные вторым способом. Такие изображения называются векторными.

В векторном способе кодирования изображение разбивается на простые объекты (геометрические фигуры, кривые и прямые линии), которые хранятся в памяти компьютера в виде математических формул и геометрических абстракций, таких как круг, квадрат, эллипс и подобных фигур. Например, чтобы закодировать круг достаточно запомнить его радиус, координаты центра, цвет контура и способ заливки.

Составные части векторного изображения (объекты) можно редактировать независимо друг от друга. С помощью комбинации нескольких объектов, можно создавать новый объект, поэтому объекты могут иметь достаточно сложный вид. Каждый объект векторного изображения характеризуется размерами, кривизной и местоположением, которые хранятся в виде числовых коэффициентов. При масштабировании векторных изображений над их математическими формулами производятся простые математические операции.

Так, например, если необходимо увеличить размеры рисунка вдвое, достаточно просто умножить математические формулы

составляющий его объектов на коэффициент масштабирования. При этом никакой потери качества изображения не происходит. Поэтому, используя векторную графику, Вам можно не задумываться о размерах будущего изображения. Вы всегда при необходимости сможете изменить размеры изображения, как в сторону увеличения, так и в сторону уменьшения без потери качества.

Существенным недостатком векторных изображений является то, что достаточно сложно с помощью векторных объектов закодировать реалистичное изображение, например, фотографию. Для этого понадобилось бы слишком много объектов. Если же попытаться перевести фотографию в векторный формат, то из-за огромного количества получившихся элементарных объектов, составляющих изображение, размер полученного файла окажется гораздо больше, чем соответствующего файла растровой графики.

Фрактальная графика

Фрактальная графика является на сегодняшний день одним из самых быстро развивающихся и перспективных видов компьютерной графики.

Изменяя и комбинируя окраску фрактальных фигур, можно моделировать образы живой и неживой природы (например, ветви дерева или снежинки), а также составлять из полученных фигур «фрактальную композицию». Фрактальная графика, так же как векторная и трёхмерная, является вычисляемой. Её главное отличие в том, что изображение строится по уравнению или системе уравнений. Поэтому в памяти компьютера для выполнения всех вычислений ничего, кроме формулы, хранить не требуется.

Математической основой фрактальной графики является фрактальная геометрия. Здесь в основу метода построения изображений положен принцип наследования от, так называемых, «родителей» геометрических свойств объектов-наследников. Понятия фрактал, фрактальная геометрия и фрактальная графика, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского "fractus" и в переводе означает «состоящий из фрагментов». Оно было предложено математиком Бенуа Мандель-Бротом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому. Одним из основных свойств фракталов является самоподобие. Объект называют самоподобным, когда увеличенные части объекта походят на сам

объект и друг на друга. Перефразируя это определение, можно сказать, что в простейшем случае небольшая часть фрактала содержит информацию обо всем фрактале.

В центре фрактальной фигуры находится её простейший элемент — равносторонний треугольник, который получил название «фрактальный». Затем, на среднем отрезке сторон строятся равносторонние треугольники со стороной, равной 1/3 от стороны исходного фрактального треугольника. В свою очередь, на средних отрезках сторон полученных треугольников, являющихся объектами-наследниками первого поколения, выстраиваются треугольники-наследники второго поколения со стороной 1/9 от стороны исходного треугольника.

Таким образом, мелкие элементы фрактального объекта повторяют свойства всего объекта. Полученный объект носит название «фрактальной фигуры». Процесс наследования можно продолжать до бесконечности. Таким же образом можно описать и такой графический элемент как прямая. Только изменив коэффициенты уравнения, можно получить совершенно другое изображение. Эта идея нашла использование в компьютерной графике благодаря компактности математического аппарата, необходимого для ее реализации. Так, с помощью нескольких математических коэффициентов можно задать линии и поверхности очень сложной формы. Итак, базовым понятием для фрактальной компьютерной графики являются «Фрактальный треугольник». Затем идет «Фрактальная фигура», «Фрактальный объект», «Фрактальная прямая», «Фрактальная композиция», «Объект-родитель» и «Объект наследник». Следует обратить внимание на то, что фрактальная компьютерная графика как вид компьютерной графики двадцать первого века получила широкое распространение не так давно. Её возможности трудно переоценить. Фрактальная компьютерная графика позволяет создавать абстрактные композиции, где можно реализовать множество приёмов: горизонтали и вертикали, диагональные направления, симметрию и асимметрию и др. Сегодня немногие компьютерщики в нашей стране и за рубежом знают фрактальную графику. С чем можно сравнить фрактальное изображение? Ну, например, со сложной структурой кристалла, со снежинкой, элементы которой выстраивается в одну сложную композицию. Это свойство фрактального объекта может быть удачно использовано для создания орнамента или декоративной композиции. Сегодня разработаны алгоритмы синтеза коэффициентов фрактала, позволяющего воспроизвести копию любой картинки сколь угодно близкой к исходному оригиналу. С точки зрения машинной графики,

фрактальная геометрия незаменима при генерации искусственных облаков, гор, поверхности моря. Фактически, благодаря фрактальной графике, найден способ эффективной реализации сложных неевклидовых объектов, образы которых весьма похожи на природные.

Геометрические фракталы на экране компьютера — это узоры, построенные самим компьютером по заданной программе. Помимо фрактальной живописи существуют фрактальная анимация и фрактальная музыка. Создатель фракталов — это художник, скульптор, фотограф, изобретатель и ученый в одном лице. Вы сами задаете форму рисунка математической формулой, исследуете сходимость процесса, варьируя его параметры, выбираете вид изображения и палитру цветов, то есть творите рисунок «с нуля». В этом одно из отличий фрактальных графических редакторов (и в частности — Painter) от прочих графических программ.

2.6. Кодирование звука. Способы сжатия звуковой информации

Звук представляет собой непрерывный сигнал — звуковую волну с меняющейся амплитудой и частотой. Чем больше амплитуда звукового сигнала, тем он громче для человека. Чем больше частота сигнала, тем выше его тон.

1. Частота звуковой волны выражается числом колебаний в секунду и измеряется **в герцах** (Гц, Hz).
2. Человеческое ухо способно воспринимать звуки в диапазоне от 20 Гц до 20 кГц, который называют **звуковым.**
3. Количество бит, отводимое на один звуковой сигнал, называют глубиной кодирования звука. Современные звуковые карты обеспечивают 16-, 32- или 64-битную глубину кодирования звука.

При кодировании звуковой информации непрерывный сигнал заменяется дискретным, то есть превращается в последовательность электрических импульсов (двоичных нулей и единиц).

Процесс перевода звуковых сигналов от непрерывной формы представления к дискретной цифровой форме называют оцифровкой.

Важной характеристикой при кодировании звука является частота дискретизации — количество измерений уровней сигнала за 1секунду:

- 1 (одно) измерение в секунду соответствует частоте 1 Гц.
- 1000 измерений в секунду соответствует частоте 1 кГц.

Частота дискретизации звука — это количество измерений громкости звука за одну секунду. Количество измерений может лежать в

диапазоне от 8 кГц до 48 кГц (от частоты радиотрансляции до частоты, соответствующей качеству звучания музыкальных носителей).

Чем больше частота и глубина дискретизации звука, тем более качественным будет звучание оцифрованного звука. Самое низкое качество оцифрованного звука, соответствующее качеству телефонной связи, получается при частоте дискретизации 8000 раз в секунду, глубине дискретизации 8 битов и записи одной звуковой дорожки (режим «моно»). Самое высокое качество оцифрованного звука, соответствующее качеству аудио-CD, достигается при частоте дискретизации 48 000 раз в секунду, глубине дискретизации 16 битов и записи двух звуковых дорожек (режим «стерео»).

Необходимо помнить, что чем выше качество цифрового звука, тем больше информационный объем звукового файла.

Оценить информационный объём моноаудиофайла (V) можно следующим образом: $V = N \cdot f \cdot k$, где N — общая длительность звучания (секунд), f — частота дискретизации (Гц), k — глубина кодирования (бит).

Например, при длительности звучания 1 минуту и среднем качестве звука (16 бит, 24 кГц):

$V = 60 \cdot 24000 \cdot 16$ бит= 23040000 бит = 2880000 байт=

= 2812,5 Кбайт = 2,75 Мбайт.

При кодировании стереозвука процесс дискретизации производится отдельно и независимо для левого и правого каналов, что, соответственно, увеличивает объём звукового файла в два раза по сравнению с монозвуком.

Например, оценим информационный объём цифрового стереозвукового файла длительностью звучания 1секунда при среднем качестве звука (16 битов, 24000 измерений в секунду). Для этого глубину кодирования необходимо умножить на количество измерений в 1секунду и умножить на 2 (стереозвук):

V=16 бит ·24000·2 = 768000 бит = 96000 байт = 93,75 Кбайт.

Существуют различные методы кодирования звуковой информации двоичным кодом, среди которых можно выделить два основных направления: метод FM и метод Wave-Table.

Метод FM (Frequency Modulation) основан на том, что теоретически любой сложный звук можно разложить на последовательность простейших гармонических сигналов разных частот, каждый из которых представляет собой правильную синусоиду, и, следовательно, может быть описан кодом. Разложение звуковых сигналов в гармонические ряды и представление в виде дискретных

цифровых сигналов выполняют специальные устройства — аналогово-цифровые преобразователи (АЦП).

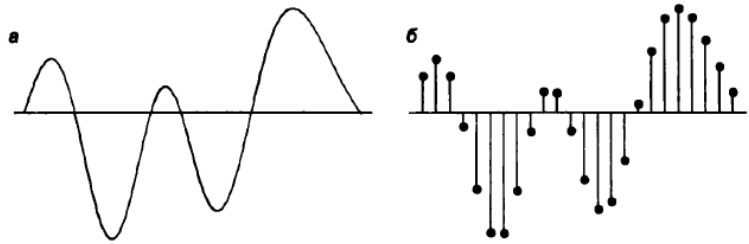


Рис 2.8. Преобразование звукового сигнала в дискретный сигнал: а — звуковой сигнал на входе АЦП; б — дискретный сигнал на выходе АЦП.

Обратное преобразование для воспроизведения звука, закодированного числовым кодом, выполняют цифро-аналоговые преобразователи (ЦАП). Процесс преобразования звука представлен на рис. ниже. Данный метод кодирования не даёт хорошего качества звучания, но обеспечивает компактный код.

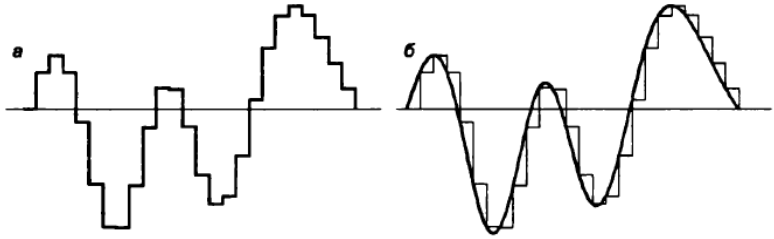


Рис 2.9. Преобразование дискретного сигнала в звуковой сигнал: а — дискретный сигнал на входе ЦАП; б — звуковой сигнал на выходе ЦАП.

Таблично-волновой метод (Wave-Table) основан на том, что в заранее подготовленных таблицах хранятся образцы звуков окружающего мира, музыкальных инструментов и т. д. Числовые коды выражают высоту тона, продолжительность и интенсивность звука и прочие параметры, характеризующие его особенности. Поскольку в качестве образцов используются «реальные» звуки, качество звука, полученного в результате синтеза, получается очень высоким и

приближается к качеству звучания реальных музыкальных инструментов.

Звуковые файлы имеют несколько форматов. Наиболее популярные из них MIDI, WAV, MP3.

Формат MIDI (Musical Instrument Digital Interface) изначально был предназначен для управления музыкальными инструментами. В настоящее время используется в области электронных музыкальных инструментов и компьютерных модулей синтеза.

Формат аудиофайла WAV (waveform) представляет произвольный звук в виде цифрового представления исходного звукового колебания или звуковой волны. Все стандартные звуки Windows имеют расширение WAV.

Формат MP3 (MPEG-1 Audio Layer 3) — один из цифровых форматов хранения звуковой информации. Он обеспечивает более высокое качество кодирования.

2.7. Вопросы и задания для самоконтроля

1. Какие виды представления информации существуют в вычислительной технике?
2. Расскажите о представлении информации в вычислительных машинах. Форматы чисел с фиксированной и плавающей точкой.
3. Как в ЭВМ представляются целые числа со знаком и целые числа без знака?
4. Что такое мантисса?
5. Что такое прямой, обратный и дополнительные коды? Как они вычисляются?
6. Опишите процесс сложения и вычитания в вычислительных системах.
7. Опишите процесс умножения и деления в вычислительных системах.
8. Как кодируют текстовую информацию?
9. Назовите известные вам кодировки текстовой информации.
10. Как кодируется растровое изображение?
11. Как кодируется векторное изображение?
12. В чём различие между растровой и векторной графикой?
13. Что такое фрактальная графика и какие у нее особенности?
14. В чём различие между фрактальной и векторной графикой?
15. Назовите возможные сферы применения фрактальной графики.
16. Опишите процесс кодирования и способы сжатия звуковой информации.

17. Выполните сложение, вычитание, умножение и деление двоичных чисел 10102 и 102 и проверьте правильность выполнения арифметических действий с помощью электронного калькулятора.
18. Выполните сложение, вычитание, умножение и деление двоичных чисел 11002 и 1002 и проверьте правильность выполнения арифметических действий с помощью электронного калькулятора.
19. Сложите восьмеричные числа: 58 и 48, 178 и 418.
20. Сложите восьмеричные числа: 88 и 168, 208 и 98.
21. Выполните вычитание восьмеричных чисел: 158 и 28, 428 и 298.
22. Выполните сложение шестнадцатеричных чисел: С16 и 916, 2916 и 1816.
23. Представьте числа 39510, 17610 в прямом, обратном и дополнительном коде, предполагая, что каждое из них может быть целым неотрицательным, либо целым отрицательным.
24. Представьте числа 23210, 27010 в прямом, обратном и дополнительном коде, предполагая, что каждое из них может быть целым неотрицательным, либо целым отрицательным.
25. Представьте числа 18010, 10410 в прямом, обратном и дополнительном коде, предполагая, что каждое из них может быть целым неотрицательным, либо целым отрицательным.
26. Представьте числа 18510, 23610 в прямом, обратном и дополнительном коде, предполагая, что каждое из них может быть целым неотрицательным, либо целым отрицательным.
27. Выполните операции машинного сложения и вычитания в дополнительных кодах для чисел 13810 и 21810.
28. Выполните операции машинного сложения и вычитания в дополнительных кодах для чисел 14610 и 29310.
29. Сложите числа: 178 и -1716, -418 и 4116.
30. Сложите числа: 208 и -2016, -218 и 1016.

3. ЛОГИЧЕСКИЕ ОСНОВЫ ОРГАНИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

3.1. Основные понятия алгебры логики

Основными элементами булевой алгебры являются двоичные высказывания и логические операции с ними:

P = 2, A∈{true,false} | {yes, no} | {0,1}.

Все булевы операции подразделяются на:

- унарные — исполняемые с одним операндом;
- бинарные — исполняемые с двумя операндами;
- тернарные — исполняемые с тремя операндами.

Результаты исполнения операций определяются так называемыми «таблицами истинности». В этих таблицах будем использовать множество A, B ∈ {0,1}.

Основные булевы функции:

- Отрицание (инверсия) ¬A, Ā, NOT (A).

Таблица истинности функции:

A	¬A
0	1
1	0

Основные свойства отрицания:

¬¬A = A

Пример использования отрицания:

NOT	1	1	0	1	0	0	1	0
	0	0	1	0	1	1	0	1

- Дизъюнкция (логическое сложение — «ИЛИ») A ∨ B, A OR B.

Таблица истинности функции:

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

Иными словами, функция истинна, если хотя бы один из операндов истинен.

Основные свойства дизъюнкции:
$A \vee 0 = A$, $A \vee 1 = 1$, $A \vee A = A$
Пример использования дизъюнкции:

```
    1 0 0 1 0 1 1 0
OR  1 1 0 1 1 0 1 0
-------------------
    1 1 0 1 1 1 1 0
```

На практике эту функцию можно использовать для установки определённых битов двоичного числа по определённой маске, не затрагивая остальные:

```
    a a a a a a a a
OR  0 0 0 0 1 1 1 1
-------------------
    a a a a 1 1 1 1
```

- Конъюнкция (логическое умножение — «И») $A \wedge B$, A AND B.
  Таблица истинности функции:

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

Иными словами, функция истинна, если оба операнда истинны.

Основные свойства конъюнкции:
$A \wedge 0 = 0$, $A \wedge 1 = A$, $A \wedge A = A$

Пример использования конъюнкции:

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

На практике эту функцию можно использовать для очистки определённых битов двоичного числа по определённой маске, не затрагивая остальные:

	a	a	a	a	a	a	a	a
AND	1	1	1	1	0	0	0	0
	a	a	a	a	0	0	0	0

Примеры применения булевых функций для решения типичных задач сегментирования сети.

1. Сетевой администратор выдал вам IP-адрес и маску подсети вида 81.65.86.226/22.

- Назовите адрес подсети для данного IP-адреса.
- Назовите адрес широковещания для данной подсети.
- Назовите максимальное количество хостов, которое можно разместить в данной подсети. Учтите, что адрес подсети и адрес широковещания являются служебными и хостам не назначаются.
- Network Number = IP Address ∧ Subnet Mask

Для нашего примера:

	01010001.01000101.01010110.11100010
and	11111111.11111111.11111100.00000000
	01010001.01000101.01010100.00000000

Таким образом, Network Number = 81.65.84.0/22

- Broadcast Address = IP Address ∨ (¬ Subnet Mask)

Для нашего примера:

not	11111111.11111111.11111100.00000000
	00000000.00000000.00000011.11111111

	01010001.01000101.01010110.11100010
Or	00000000.00000000.00000011.11111111
	01010001.01000101.01010111.11111111

Таким образом, Broadcast Address = 81.65.87.255/22

- Host Number = $2^{32-\text{Mask Length}} - 2$

  Для нашего примера: $2^{32-22} - 2 = 2^{10} - 2 = 1022$ хоста.

2. Компьютеру Alpha назначен IP-адрес 81.198.166.113/21. Сможет ли Alpha связаться с компьютером Bravo с IP-адресом 81.198.156.27 напрямую, или пакеты придётся пересылать через маршрутизатор?

- Компьютер Alpha вычисляет адрес своей подсети:

```
                01010001.11000110.10100110.01110001
and             11111111.11111111.11111000.00000000
                01010001.11000110.10100000.00000000 = 81.198.160.0
```

- Компьютер Alpha вычисляет адрес подсети компьютера Bravo:

```
                01010001.11000110.10011100.00011011
and             11111111.11111111.11111000.00000000
                01010001.11000110.10011000.00000000 = 81.198.152.0
```

- Сравнивая полученные результаты, Alpha видит, что адрес подсети Bravo отличается от его собственного. Это означает, что компьютеры находятся в разных сегментах, поэтому связь возможна только через маршрутизатор.

Производные булевы функции:

- «ИСКЛЮЧАЮЩЕЕ ИЛИ» (сложение по модулю 2): A ⊕ B, A XOR B

Таблица истинности функции:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Иными словами, функция истинна, если операнды различны.
Пример использования функции «ИСКЛЮЧАЮЩЕЕ ИЛИ»:

```
      1 1 0 1 1 0 1 0
XOR   1 0 0 1 0 1 1 0
      ---------------
      0 1 0 0 1 1 0 0
```

На практике эту функцию можно использовать для слияния битов двух двоичных чисел, например, отображая курсор поверх текста:

	0	0	0	0	0	1	1	1
XOR	1	1	1	0	0	0	0	0
	1	1	1	0	0	1	1	1

- Импликация $Y = A \rightarrow B$, $Y = A \supset B$
  Таблица истинности функции:

A	B	A→B
0	0	1
0	1	1
1	0	0
1	1	1

  Операция импликация" выражается через логические функции "ИЛИ", "НЕ" в виде логической формулы:
  $A \rightarrow B = \neg A \vee B$ или $\neg(A \rightarrow B) = A \wedge \neg B$

- Эквивалентность $A \equiv B$, A XNOR B
  Таблица истинности функции:

A	B	A≡B
0	0	1
0	1	0
1	0	0
1	1	1

  Иными словами, функция истинна, если операнды равны.
  Пример использования эквивалентности:

	1	0	0	1	0	1	1	0
XNOR	1	1	0	1	1	0	1	0
	1	0	1	1	0	0	1	1

- Стрелка Пирса $A \downarrow B$, A NOR B

Таблица истинности функции:

A	B	A↓B
0	0	1
0	1	0
1	0	0
1	1	0

Иными словами, функция истинна, если оба операнда ложны. В электронике эта функция реализуется с помощью двух транзисторов, соединённых параллельно:

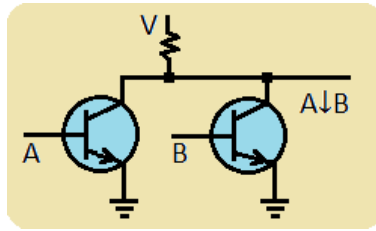


Рис 3.1. Стрелка Пирса

Через NOR можно выразить остальные булевы функции, что делает возможным построение электронных схем на базе единственного стандартного элемента:

¬A ≡ A↓A
A∧B ≡ (A↓A) ↓ (B↓B)
A∨B ≡ (A↓B) ↓ (A↓B)

- Штрих Шеффера A | B, A NAND B
  Таблица истинности функции:

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

Иными словами, функция истинна, если хотя бы один операнд ложен. В электронике эта функция реализуется с помощью двух транзисторов, соединённых последовательно. Схема представлена на рис. 3.2.

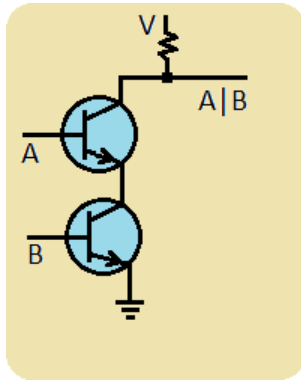


Рис 3.2. Шрих Шеффера

Через NAND можно выразить остальные булевы функции, что делает возможным построение электронных схем на базе единственного стандартного элемента:

¬A ≡ A|A
A∨B ≡ (A|A) | (B|B)
A∧B ≡ (A|B) | (A|B)

Основные законы булевой алгебры:

- Переместительный закон (коммутативность):

A○B = B○A, ○ ∈ {∧,∨,⊕,≡,|,↓}

- Сочетательный закон (ассоциативность):

A○ (B○C) = (A○B) ○C, ○ ∈ {∧,∨,⊕,≡}

- Распределительный закон (дистрибутивность):

A∧ (B∨C) = A∧B ∨ A∧C
A∧ (B⊕C) = A∧B ⊕ A∧C
A∨ B∧C = (A∨B) ∧ (A∨C)

- Законы де Моргана:

Not (A) and Not (B) = Not (A or B)
Not (A) or Not (B) = Not (A and B)

Not (A) and Not (B) and Not (C) = Not (A or B or C)
Not (A) or Not (B) or Not (C) = Not (A and B and C)

3.2. Понятие карты Карно

Карта Карно́ — графический способ минимизации булевых функций, обеспечивающий относительную простоту работы с большими выражениями. Способ представляет собой операции попарного неполного склеивания и элементарного поглощения. Карты Карно рассматриваются как перестроенная соответствующим образом таблица истинности функции. Карты Карно можно рассматривать как определенную плоскую развертку n-мерного булева куба.

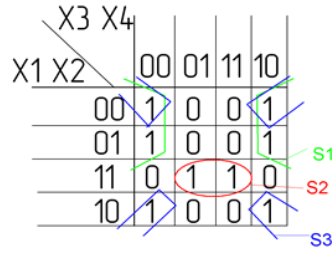


Рис. 3.3. Пример карты Карно

Карты Карно были изобретены в 1952 Эдвардом В. Вейчем и усовершенствованы в 1953 Морисом Карно, физиком из «Bell Labs», и были призваны помочь упростить цифровые электронные схемы.

В карту Карно булевы переменные передаются из таблицы истинности и упорядочиваются с помощью кода Грея, в котором каждое следующее число отличается от предыдущего только одним разрядом.

3.3. Принципы минимизации с помощью карт Карно

Основным методом минимизации логических функций, представленных в виде СДНФ или СКНФ, является операция попарного неполного склеивания и элементарного поглощения. Операция попарного склеивания осуществляется между двумя термами

(членами), содержащими одинаковые переменные, вхождения которых (прямые и инверсные) совпадают для всех переменных, кроме одной. В этом случае все переменные, кроме одной, можно вынести за скобки, а оставшиеся в скобках прямое и инверсное вхождение одной переменной подвергнуть склейке. Например:

$$\overline{X}_1X_2X_3X_4 \vee \overline{X}_1X_2\overline{X}_3X_4 = \overline{X}_1X_2X_4(X_3 \vee \overline{X}_3) = \overline{X}_1X_2X_4.$$

Аналогично для КНФ:

$$(\overline{X}_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X}_1 \vee X_2 \vee \overline{X}_3 \vee X_4) = \overline{X}_1 \vee X_2 \vee X_4 \vee X_3\overline{X}_3 = \overline{X}_1 \vee X_2 \vee X_4.$$

В
озможность поглощения следует из очевидных равенств:

$$A \vee \overline{A} = 1; A\overline{A} = 0.$$

Таким образом, главной задачей при минимизации СДНФ и СКНФ является поиск термов, пригодных к склейке с последующим поглощением, что для больших форм может оказаться достаточно сложной задачей. Карты Карно предоставляют наглядный способ отыскания таких термов.

Как известно, булевы функции *N* переменных, представленные в виде СДНФ или СКНФ, могут иметь в своём составе 2*N* различных термов. Все эти члены составляют некоторую структуру, топологически эквивалентную *N*-мерному кубу, причём любые два терма, соединённые ребром, пригодны для склейки и поглощения.

На рисунке 3.4 изображена простая таблица истинности для функции из двух переменных, соответствующий этой таблице 2-мерный куб (квадрат), а также 2-мерный куб с обозначением членов СДНФ и эквивалентная таблица для группировки термов:

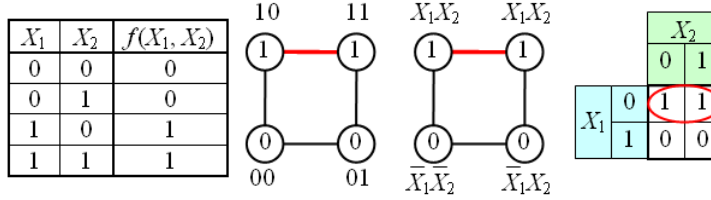


Рис. 3.4. Таблица истинности для булевой функции двух переменных

В случае функции трёх переменных приходится иметь дело с трёхмерным кубом. Это сложнее и менее наглядно, но технически возможно. На рисунке 3.5 в качестве примера показана таблица истинности для булевой функции трёх переменных и соответствующий ей куб.

X_1	X_2	X_3	$f(X_1, X_2, X_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

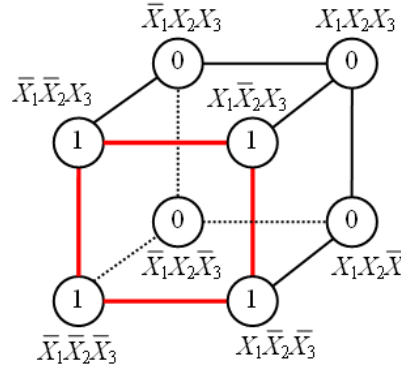


Рис. 3.5. Таблица истинности и куб для булевой функции трёх переменных

Как видно из рисунка 3.5, для трёхмерного случая возможны более сложные конфигурации термов. Например, четыре терма, принадлежащие одной грани куба, объединяются в один терм с поглощением двух переменных:

$$\overline{X}_1\overline{X}_2\overline{X}_3 \vee X_1\overline{X}_2\overline{X}_3 \vee \overline{X}_1\overline{X}_2X_3 \vee X_1\overline{X}_2X_3 =$$
$$= \overline{X}_2(\overline{X}_1\overline{X}_3 \vee X_1X_3 \vee X_1\overline{X}_3 \vee X_1X_3) = \overline{X}_2(\overline{X}_1 \vee X_1)(\overline{X}_3 \vee X_3) = \overline{X}_2$$

В общем случае можно сказать, что $2K$ термов, принадлежащие одной K-мерной грани гиперкуба, склеиваются в один терм, при этом поглощаются K переменных.

Для упрощения работы с булевыми функциями большого числа переменных был предложен следующий удобный приём. Куб, представляющий собой структуру термов, разворачивается на плоскость как показано на рисунке. Таким образом, появляется возможность представлять булевы функции с числом переменных больше двух в виде плоской таблицы. При этом следует помнить, что порядок кодов термов в таблице (00 01 11 10) не соответствует порядку

следования двоичных чисел, а клетки, находящиеся в крайних столбцах таблицы, соседствуют между собой.

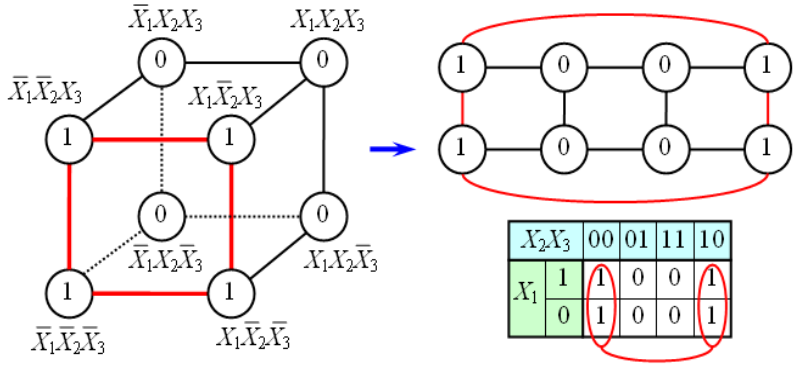


Рис. 3.6. Куб для булевой функции трёх переменных

Аналогичным образом можно работать с функциями четырёх, пяти и более переменных. Примеры таблиц для N=4 и N=5 приведены на рисунке 3.7. Для этих таблиц следует помнить, что соседними являются клетки, находящиеся в соответственных клетках крайних столбцов и соответственных клетках верхней и нижней строки. Для таблиц 5 и более переменных нужно учитывать также, что квадраты 4x4 виртуально находятся друг над другом в третьем измерении, поэтому соответственные клетки двух соседних квадратов 4x4 являются соседними, и соответствующие им термы можно склеивать.

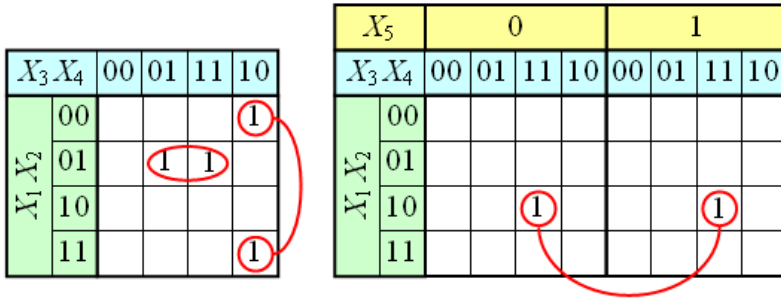


Рис. 3.7. Таблица истинности для булевой функции трёх переменных

Карта Карно может быть составлена для любого количества переменных, однако удобно работать при количестве переменных не более пяти. По сути, карта Карно — это таблица истинности, составленная в 2-х мерном виде. Благодаря использованию кода Грея в ней, верхняя строка является соседней с нижней, а правый столбец — соседним с левым, и вся карта Карно сворачивается в фигуру тор (бублик). На пересечении строки и столбца проставляется соответствующее значение из таблицы истинности. После того, как карта заполнена, можно приступать к минимизации.

Если необходимо получить минимальную ДНФ, то в карте рассматриваем только те клетки, которые содержат единицы. Если работа идет с КНФ, то рассматриваются те клетки, которые содержат нули. Сама минимизация производится по следующим правилам (на примере ДНФ):

1. объединяем смежные клетки, содержащие единицы, в область так, чтобы одна область содержала 2^n клеток (n — целое число в диапазоне 0…∞). Необходимо помнить про то, что крайние строки и столбцы являются соседними между собой, в области не должно находиться клеток, содержащих нули;
2. область должна располагаться симметрично оси (ей) (оси располагаются через каждые четыре клетки);
3. несмежные области, расположенные симметрично оси (ей), могут объединяться в одну;
4. область должна быть как можно больше, а количество областей как можно меньше;
5. области могут пересекаться;
6. возможно несколько вариантов покрытия.

Далее берём первую область и смотрим, какие переменные не меняются в пределах этой области, выписываем конъюнкцию этих переменных; если неизменяющаяся переменная нулевая, проставляем над ней инверсию. Берём следующую область, выполняем то же самое, что и для первой, и т. д. для всех областей. Конъюнкции областей объединяем дизъюнкцией.

Для КНФ выполняем все действия аналогичным образом, только рассматриваем клетки с нулями, неизменяющиеся переменные в пределах одной области объединяем в дизъюнкции (инверсии проставляем над единичными переменными), а дизъюнкции областей объединяем в конъюнкцию. На этом минимизация считается законченной.

3.4. Вопросы и задания для самоконтроля

1. Чем являются основные элементы булевой алгебры? На какие группы по количеству операндов делятся логические функции?
2. Перечислите основные булевы функции. К каждой функции подберите пример.
3. Перечислите производные булевы функции. К каждой функции подберите пример.
4. Приведите примеры применения булевых функций.
5. Каков приоритет выполнения булевых операций? Приведите примеры вычисления булевых функций с учетом приоритета операций.
6. Назовите переместительный, сочетательный и распределительный законы логики.
7. Перечислите законы де Моргана. Приведите примеры применения.
8. Что такое карты Карно? Приведите пример применения.
9. Назовите основные принципы минимизации. Приведите пример минимизации.
10. Что такое СКНФ? Приведите пример составления СКНФ по таблице истинности.
11. Что такое СДНФ? Приведите пример составления СДНФ по таблице истинности.
12. Опишите правила получения КНФ функции по карте Карно.
13. Составить таблицу истинности для следующего выражения:

$$\left(x \wedge y \oplus \overline{z} \vee \overline{x} \wedge y\right) \to z\,.$$

14. Составить таблицу истинности для следующего выражения:

$$\overline{\overline{x \wedge y}} \to \overline{\overline{y \equiv z}}\,.$$

15. Составить таблицу истинности для следующего выражения:

$$(\overline{C} \vee D \oplus A) \oplus (C \oplus B)\,.$$

16. По таблице истинности восстановите логическое выражение:

x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

17. Упростите следующее логическое выражение:

$$\bar{x} \wedge z \vee \overline{x \wedge y} \vee x \wedge z \wedge \bar{y}.$$

18. Упростите логические выражения из задания 13.
19. Составьте КНФ по данной карте Карно:

X3X4 / X1X2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	0
10	1	1	0	1

20. Составьте ДНФ по данной карте Карно:

X3X4 / X1X2	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	0	0
10	1	0	0	1

4. АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ МАШИН И ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

4.1. Типы архитектур: фон Неймана и гарвардская

Вычислительные машины для своей работы нуждаются в командах. Машине шаг за шагом должно быть указано, что выполнять. Сделать это довольно легко, если хранить команды в памяти. Тогда вычислитель должен только выбрать и выполнить команду. Такие вычислители называются машинами «с сохраняемой программой». То есть машина выбирает сначала команду, затем данные, обрабатывает их, а полученный результат вновь передает в память.

Известны два широко распространенных типа архитектур вычислителей: фон Неймана (или принстонская) и гарвардская.

Архитектура фон Неймана

Машины фон Неймана хранят программу и данные в одной и той же области памяти. В машинах этого типа команды содержат указание, что выполнить, и адрес данных, подлежащих обработке. Внутри этой машины имеются два основных функциональных блока. Арифметико-логическое устройство (АЛУ) выполняет самые важные операции: умножение, сложение, вычитание и многие другие. Другой блок — устройство ввода/вывода управляет потоком внешних, относительно машины, данных.

Гарвардская архитектура

Единственное отличие гарвардской архитектуры состоит в том, что память программ и память данных разделены, и они используют физически разделенные линии передачи. Это позволяет подобной машине пересылать команды и данные одновременно. Следовательно, такая конструкция может значительно увеличить производительность цифрового процессора обработки сигналов. Гарвардская машина, как и фон-неймановская, имеет арифметико-логическое устройство и устройство ввода/вывода. К сожалению, расплатой за высокую скорость является высокая цена процессора.

История архитектур

История рассмотренных архитектур весьма интересна. Гарвардская архитектура была разработана Говардом Айхеном (Howard Aiken) в конце 1930-х годов в Гарвардском университете (отсюда - название). Первая машина Harvard Mark-1 работала уже в 1944 году. За ней в 1946 году последовал «электронный числовой

интегратор и калькулятор» (Electronic Numerical Integrator and Calculator - ENIAC), разработанный Пенсильванским университетом.

Джон фон Нейман, математик венгерского происхождения, предложил простейшую архитектуру, а именно — с объединенной памятью программного продукта и данных. С тех пор это простое решение стало стандартом. Машина фон Неймана была создана в Принстонском институте новейших исследований в 1951 году.

Персональные компьютеры общего назначения используют процессоры, построенные согласно архитектуре фон Неймана. Другие семейства компьютеров также используют архитектуру фон Неймана. Гарвардская архитектура является наиболее подходящей для специализированных микропроцессоров, предназначенных для решения прикладных задач в реальном времени.

DSP (Digital Signal Procesors)

Цифровые процессоры обработки сигналов обычно используют гарвардскую архитектуру, хотя существуют и процессоры с фон Неймановской архитектурой. Однако гарвардской архитектуре присущ один существенный недостаток. Вследствие того, что память данных и память программ разделены, на кристалле необходимо было иметь в два раза больше выводов для адреса и данных. К сожалению, кремниевая технология такова, что увеличение числа выводов на кристалле приводит к росту цены. Однако блестящие инженеры-электронщики, которые долго бились над этой проблемой, предложили изящное решение. Оно состоит в том, чтобы для всех внешних данных, включая команды, использовать одну шину, а другую — для адресации, внутри же процессора иметь отдельную шину команд и шину данных и две соответствующих шины адреса. Разделение информации о программе и данных на выводах процессора производится благодаря их временному разделению (мультиплексированию). На это требуется два командных цикла: в первом цикле на выводы поступает информация о программе, а во втором на эти же выводы поступают данные. Затем все повторяется. Такие машины называются процессорами «с модифицированной гарвардской архитектурой».

На рисунках 4.1 и 4.2 приведены структурные схемы вычислителей, построенных согласно архитектуре фон Неймана и гарвардской. Машина на рис. 4.1 состоит из блока управления, арифметико-логического устройства (АЛУ), памяти и устройств ввода-вывода. В ней реализуется концепция хранимой программы: программы и данные хранятся в одной и той же памяти.

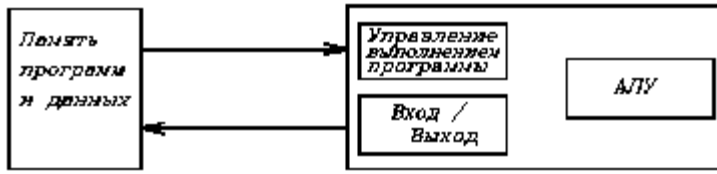


Рис 4.1. Структурная схема Фон Неймановской архитектуры

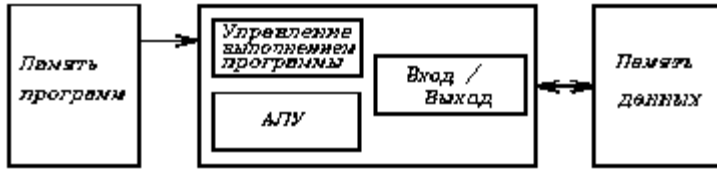


Рис 4.2. Структурная схема гарвардской архитектуры.

Выполняемые действия определяются блоком управления и АЛУ, которые вместе являются основой центрального процессора. Центральный процессор выбирает и исполняет команды из памяти последовательно, адрес очередной команды задается “счетчиком адреса” в блоке управления. Этот принцип исполнения называется последовательной передачей управления. Архитектуры фон Неймана и гарвардская — не единственные варианты построения ЭВМ; есть и другие, которые не соответствуют указанным принципам. Однако подавляющее большинство современных компьютеров основаны именно на указанных принципах, включая и сложные многопроцессорные комплексы, которые можно рассматривать как объединение фон Неймановских машин.

Итак, выделим три основных признака фон Неймановской архитектуры:

- память состоит из последовательности ячеек памяти с адресами;
- хранение команд программы и обрабатываемых ими данных строится на одинаковых принципах (с точки зрения обработки сообщений);

- программа выполняется покомандно, в соответствии с порядком команд. Устройство управления централизовано.

Хотя архитектура фон Неймана была логичным решением проблемы создания первой машины с запоминаемой программой, она не всегда удовлетворяет требованиям, которые предъявляются к программам, написанным на языках высокого уровня. В отличие от перечисленных выше характеристик языки программирования высокого уровня имеют следующие характеристики:

1. память состоит из набора дискретных именуемых переменных;
2. используются многомерные, а не просто линейные данные;
3. существует резкое разграничение между данными и командами;
4. назначение данных является внутренней частью самих данных, т.е. связано с самими данными; тип данных определяет и операции, выполняемые над ними.

Машина фон Неймана оказывается плохим средством для выполнения программ, написанных на языках высокого уровня по следующим причинам:

1. чрезмерный расход программных средств с целью согласования возможностей языка со структурой памяти по фон Нейману;
2. машина фон Неймана универсальна (адресация к чему угодно в памяти и т.д.). Универсальность отсутствует в языках программирования, следовательно, компилятор решает эту проблему;
3. в силу относительной примитивности принципа организации памяти по фон Нейману операции (набор команд), выполняемые машиной, оказываются в равной мере примитивными.

Итак, принципы, на которых основывается архитектура фон Неймана, не согласуются с принципами языков программирования и иногда даже противоречат им.

4.2. Понятие семантического разрыва

Факт различия принципов, лежащих в основе языков программирования высокого уровня и тех принципов, которые определяют архитектуру ЭВМ, называется семантическим разрывом.

Одной из дополнительных причин семантического разрыва является использование двоичной арифметики:

- современные вычислительные системы вынуждены затрачивать много времени на десятично-двоичные преобразования;

- двоичные числа конечной длины — это лишь аппроксимация десятичных чисел (вспомните представление дробных десятичных чисел).

Чтобы определить наличие семантического разрыва, нужно выбрать какой-либо язык и архитектуру ЭВМ, и изучить взаимосвязи между ними. Анализ проводится следующим образом: определяют принципы, положенные в основу языков программирования и пытаются измерить "расстояние" между ними и соответствующими принципами, на которых базируется анализируемая архитектура ЭВМ. Например, по пунктам: массивы (в языках высокого уровня — многомерные, есть возможность обращения к отдельным элементам массива; в архитектуре ЭВМ — использование индексных регистров, как подобие одной из возможностей языка), структуры (в языках высокого уровня — использование наборов разнородных данных — записи; в архитектуре ЭВМ нет адекватных средств), обработка строк, процедуры, блочные структуры, представление данных и т.д.

4.3. Концепция многоуровневой работы вычислительных систем

ЭВМ и ВС, являясь сложными системами, содержат большое число взаимодействующих аппаратных и программных модулей. Сложные системы не имеют простых описаний. Это связано со следующими особенностями: большое число и разнородность элементов системы (модулей), разнообразие и нерегулярность связей между ними, отсутствие единого аппарата для описания поведения различных модулей.

Для описания, проектирования и организации управления в ВС используется иерархический подход. Рассмотрим уровни организации вычислительных процессов (рис. 4.3) в ЭВМ и ВС, а также категории специалистов, использующие соответствующие уровни представления в своей профессиональной деятельности, и процессы, реализующие взаимодействие уровней.

На концептуальном уровне пользователь анализирует задачу, выбирает метод ее решения, разрабатывает алгоритм. Для сложных задач выполняется функциональная декомпозиция, определяются структуры данных, выделяются программные модули, определяются связи между ними.

Рис. 4.3. Уровни организации вычислительных процессов

Далее пишется программа на одном из языков высокого уровня. Такое представление программы является машинно-независимым, то есть независимым от архитектуры ВС и от особенностей аппаратного обеспечения.

Для описания, проектирования и организации управления в ВС используется иерархический подход. Рассмотрим уровни организации вычислительных процессов (рис. 4.3) в ЭВМ и ВС, а также категории специалистов, использующие соответствующие уровни представления в своей профессиональной деятельности, и процессы, реализующие взаимодействие уровней.

На концептуальном уровне пользователь анализирует задачу, выбирает метод ее решения, разрабатывает алгоритм. Для сложных задач выполняется функциональная декомпозиция, определяются

структуры данных, выделяются программные модули, определяются связи между ними.

Далее пишется программа на одном из языков высокого уровня. Такое представление программы является машинно-независимым, то есть независимым от архитектуры ВС и от особенностей аппаратного обеспечения.

На уровне машинных команд обеспечивается связь программных и аппаратных средств. На этом уровне разрабатывается список команд, определяются способы кодирования операций и адресов, число адресных полей в команде и другие параметры, заложенные в структуру ЭВМ. Перевод программы с языка высокого уровня в машинные коды осуществляет специальная программа – компилятор. Для одной программы, написанной на языке высокого уровня, можно породить много программ на уровне машинных команд, которые по выполняемым функциям эквивалентны, но различаются по использованию ресурсов ЭВМ (памяти, процессорного времени). Поэтому актуальна задача построения оптимизирующего компилятора. Чем ближе состав операторов языка высокого уровня к списку команд ВМ, тем проще компилятор и эффективнее полученная для исполнения программа. При программировании часто используют язык Ассемблера. Ассемблер — машинно-зависимый язык программирования. Представление задачи на языке Ассемблера занимает промежуточное положение в иерархии представлений между языками высокого уровня и машинными командами. Язык Ассемблера используют при написании системных программ, кратность исполнения которых существенно выше, чем у прикладных программ, например для МК (микроконтроллеров), ресурсы которых ограничены. Прикладные программы для МК — это, как правило, программы управления оборудованием, которые также исполняются многократно.

Следует отметить, что связь между языками высокого уровня и машинными командами может осуществляться как методом компиляции, так и методом интерпретации. При компиляции программа на языке высокого уровня сначала преобразуется полностью в программу в машинных кодах, а затем может поступать для исполнения с использованием более низких уровней вычислительного процесса. При работе компилятора программа пользователя на языке высокого уровня – это входные данные, а соответствующая программа на языке машинных команд – выходные данные, результат работы компилятора. При исполнении полученная программа в машинных кодах уже используется не как данные, а как программа, управляющая вычислительным процессом.

При интерпретации программа пользователя на языке высокого уровня непосредственно используется для управления вычислительным процессом с использованием системной программы – интерпретатора. Интерпретатор выбирает очередную команду программы на языке высокого уровня, заменяет ее последовательностью машинных команд, выполняющих требуемые функции, и сразу передает эту последовательность машинных команд на исполнение. Затем с учетом признаков, характеризующих результат выполненных преобразований, выбирается следующая команда программы на языке высокого уровня и т.д.

При компиляции требуется более сложная системная программа – компилятор. В процессе компиляции требуется больший объем памяти, но зато исполнение пользовательской программы происходит существенно быстрее. В связи с этим в современных ВМ в основном используется метод компиляции.

На уровне регистровых передач осуществляются микрооперации, выполняемые аппаратурой ВМ. Это операции передачи, запоминания и преобразования кодов, выполняемые пересылкой сигналов между регистрами через комбинационные (логические) схемы (рис. 4.3). Микрооперация — это операция над кодами, выполняемая одной регистровой передачей (за такт). Для настройки схем на выполнение требуемой микрооперации при регистровой передаче требуется сформировать соответствующий набор управляющих сигналов. Код набора управляющих сигналов называют микрокомандой. Последовательность микрокоманд, соответствующая исполнению машинной команды, называют микропрограммой. Каждой машинной команде соответствует своя микропрограмма. Связь между уровнями машинных команд и регистровых передач осуществляется методом интерпретации. Выборкой команд из программы в машинных кодах и интерпретации выбранной команды управляет аппаратное устройство управления центрального процессора. Уровень регистровых передач называют также микропрограммным.

На уровне логических вентилей рассматриваются логические схемы при логическом проектировании аппаратуры ЭВМ. Если на уровне регистровых передач рассматриваются операции с n-разрядными кодами, то на уровне вентилей — с отдельными двоичными переменными.

Многоуровневая организация упрощает реализацию ЭВМ и управление вычислительным процессом, но снижает эффективность работы за счет «накладных расходов» на управление. В процессе работы происходит «челночное» взаимодействие между уровнями:

выборка очередной команды на уровне машинных команд, интерпретация команды на уровне регистровых передач, реализация регистровой передачи передачей сигналов в сети логических элементов, фиксация признаков, характеризующих результат выполнения операции, использование признаков для разветвления при выборке следующей команды, и т.д.

В управлении вычислительным процессом с иерархической организацией участвуют как программные, так и аппаратные средства. Функционирование ЭВМ и ВС заключается в тесном и неразрывном взаимодействии аппаратуры и программ. Грань разделения функций, реализуемых аппаратно и программно, по мере развития архитектуры ЭВМ и технологии их изготовления смещается в сторону аппаратной реализации.

4.4. Не фон Неймановская архитектура

Альтернативной архитектурой является "не фон Неймановская" архитектура, допускающая одновременный анализ более одной команды. Ее создание обусловлено необходимостью распараллеливания выполнения программы между несколькими исполнительными устройствами — процессорами. Счетчик команд при этом не нужен. Порядок выполнения команд определяется наличием исходной информации для выполнения каждой из них. Если несколько команд готовы к выполнению, то принципиально возможно их назначение для выполнения таким же количеством свободных процессоров. Говорят, что такие ВС управляются потоком данных (data flow).Общая схема потоковых ВС представлена на рис. 4.4

Программа или ее часть (сегмент) размещается в памяти команд ПК, состоящей из ячеек команд. Команды имеют следующую структуру:

{код операции, операнд 1, ..., операнд L,
адрес результата 1, ..., адрес результата M}

В командах проверки условия возможно альтернативное задание адреса результата (ИЛИ — ИЛИ). Адреса результатов являются адресами ПК, т.е. результаты выполнения одних команд в качестве операндов могут поступать в текст других команд. Команда не готова к выполнению, если в ее тексте отсутствует хотя бы один операнд. Ячейка, обладающая полным набором операндов, переходит в возбужденное состояние и передает в селекторную сеть

информационный пакет (токен), содержащий код операции и необходимую числовую и связную информацию. Он поступает по сети на одно из исполнительных устройств. Там же операция выполняется, и в распределительную сеть выдается результирующий пакет, содержащий результат вычислений и адреса назначения в ПК (возможно, за счет выбора альтернативы, т.е. такой выбор — тоже результат).

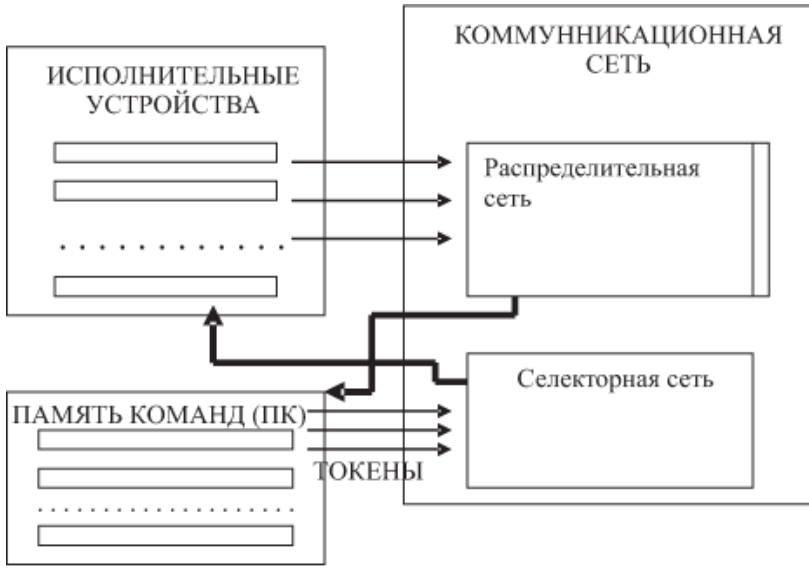


Рис 4.4. "Идеальная" потоковая ВС

По этим адресам в ПК результат и поступает, создавая возможность активизации новых ячеек. После выдачи токена в селекторную сеть операнды в тексте команды уничтожаются, что обеспечивает повторное выполнение команды в цикле, если это необходимо.

Селекторная и распределительная сети образуют коммуникационную сеть ВС.

Ожидаемая сверхвысокая производительность такой системы может быть достигнута за счет одновременной и независимой активизации большого числа готовых команд, проблематичном допущении о бесконфликтной передаче пакетов по сетям и параллельной работы многих исполнительных устройств.

Существует ряд трудностей, в силу которых "не фон Неймановские" архитектуры не обрели технического воплощения для массового применения в "классическом", отраженном выше, исполнении. Однако многие устройства используют данный принцип, но чаще всего взаимодействие процессоров при совместном решении общей задачи и их синхронизация при использовании общих данных основаны на анализе готовности данных для их обработки. Это дает основание многим конструкторам заявлять, что в своих моделях они реализовали принцип "data flow".

4.5. Стековая архитектура, каноническая стековая машина

Стеком называется память, по своей структурной организации отличная от основной памяти ЭВМ, Принципы построения стековой памяти детально рассматриваются позже, здесь же выделим только те аспекты, которые требуются для пояснения особенностей архитектуры на базе стека.

Стек образует множество логически взаимосвязанных ячеек (рисунок 4.5), взаимодействующих по принципу «последним вошел, первым вышел» (LIFO, Last In First Out).

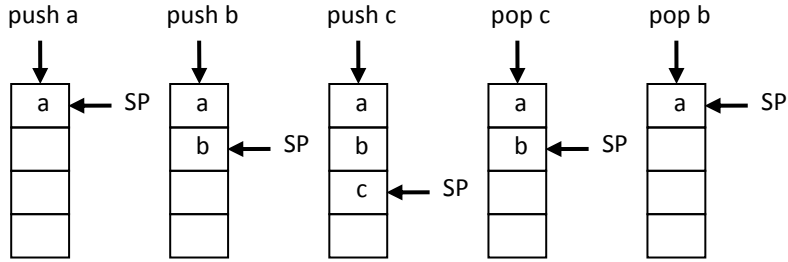


Рис 4.5. Механизм стековой памяти

Верхнюю ячейку называют вершиной стека (SP). Для работы со стеком предусмотрены две операции: push (проталкивание данных в стек) и pop (выталкивание данных из стека). Запись возможна только в ячейку стека с адресом SP-1, при этом указатель стека смещается на одну позицию вниз. Чтение допустимо только из вершины стека (ячейка с адресом SP). Извлеченная информация удаляется из стека, а указатель продвигается вверх – принимает значение SP+1.

В вычислительных машинах, где реализована архитектура на базе стека (их обычно называют стековыми), операнды выбираются из

двух верхних ячеек стековой памяти. Результат операции заносится в вершину стека. Принцип действия стековой машины рассмотрим на примере вычисления выражения:

$$a = a + b + a*c.$$

При описании вычислений с использованием стека обычно используется иная форма записи математических выражений, известная как обратная польская нотация, которую предложил польский математик Я. Лукашевич. Особенность ее в том, что в выражении отсутствуют скобки, а знак операции располагается не между операндами, а следует за ними (постфиксная форма). Последовательность операций определяется их приоритетами. Рассмотренное выше выражение в польской нотации примет вид:

$$a = ab+ ac*+.$$

Данная форма записи однозначно определяет порядок загрузки операндов и выполнения операций в стековой архитектуре. Порядок выполнения команд в ЭВМ со стековой архитектурой представлен на рисунке 4.6.

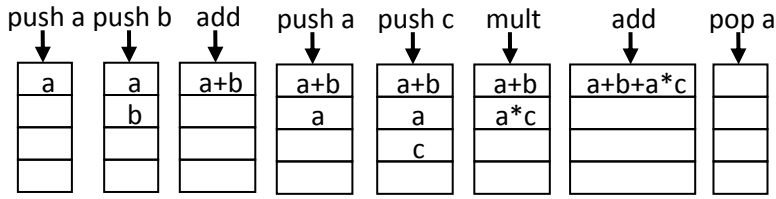


Рис 4.6. Порядок выполнения стековых команд

Основные узлы и информационные тракты одного из возможных вариантов ЭВМ на основе стековой архитектуры показаны на рисунке 4.7.

Информация может быть занесена в вершину стека из памяти или из АЛУ. Для записи в стек содержимого ячейки памяти с адресом *x* выполняется команда *push x*, по которой информация считывается из ячейки памяти, заносится в регистр данных, а затем проталкивается в стек. Результат операции из АЛУ заносится в вершину стека автоматически.

Сохранение содержимого вершины стека в ячейке памяти с адресом *x* производится командой *pop x*. По этой команде содержимое верхней ячейки стека подается на шину, с которой и производится запись в ячейку *x*, после чего производится инкремент указателя стека: SP+1.

Для выполнения арифметической или логической операции на вход АЛУ по дается информация, считанная из двух верхних ячеек стека (при этом содержимое стека продвигается на две позиции вверх, то есть операнды из стека удаляются). Результат операции заталкивается в вершину стека. Возможен вариант, когда результат сразу же переписывается в память с помощью автоматически выполняемой операции *pop x*.

Верхние ячейки стековой памяти, где хранятся операнды и куда заносится результат операции, как правило, делаются более быстродействующими и размещаются в процессоре, в то время как остальная часть стека может располагаться в основной памяти и частично даже на магнитном диске.

К достоинствам ЭВМ на базе стека следует отнести возможность сокращения адресной части команд, поскольку все операции производятся через вершину стека, то есть адреса операндов и результата в командах арифметической и логической обработки информации указывать не нужно. Код программы получается компактным. Достаточно просто реализуется декодирование команд.

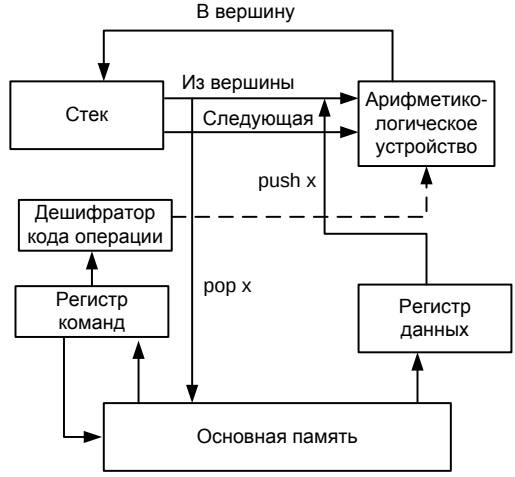


Рис 4.7. ЭВМ на основе стековой архитектуры

С другой стороны, стековая архитектура по определению не предполагает произвольного доступа к памяти, из-за чего компилятору трудно создать эффективный программный код, хотя создание самих компиляторов упрощается. Кроме того, стек становится «узким местом» ЭВМ в плане повышения производительности.

4.6. Классификация параллельных вычислительных систем

Архитектура фон Неймана обладает тем недостатком, что она последовательная. Какой бы огромный массив данных ни требовалось обработать, каждый его байт должен будет пройти через центральный процессор, даже если над всеми байтами требуется провести одну и ту же операцию. Этот эффект называется "узким горлышком фон Неймана". Для преодоления этого недостатка предлагались и предлагаются архитектуры ВС, которые называются параллельными.

Сложилось представление о двух основных уровнях, на которых в ВС применяются практические методы распараллеливания:

- на уровне программ, процессов, процедур (первый уровень распараллеливания);
- на уровне команд и операций (второй уровень распараллеливания).

Эти уровни обусловили уровни структуризации ВС на пути превращения ее в супер-ЭВМ. Современным практическим воплощением первого уровня структуризации являются однородные многопроцессорные ВС на общей (разделяемой) оперативной памяти. Они получили название симметричных ВС за обеспечение "равноправия" составляющих модулей. Окончательное признание симметричных ВС положило конец поиску "экзотических" архитектур, эффективных лишь при решении определенных классов задач. Универсальность симметричных ВС, возможность реализации на них любых вычислительных процессов с высокой эффективностью оборудования иллюстрируются многими применениями.

Уровень команд и операций наиболее ярко представлен многофункциональными АЛУ и их обобщением — решающими полями, представляющими разделяемый вычислительный ресурс многих процессоров. Некоторые современные проекты ВС в разной степени предполагают такой ресурс. Здесь основная проблема — полная загрузка отдельных исполнительных устройств (ИУ) в процессе выполнения программы.

Различают два способа реализации такой загрузки: динамический и статический.

Динамическая загрузка осуществляется аппаратурой в процессе выполнения программы. Она использует упрощенные алгоритмы распараллеливания.

Статическая загрузка предусматривается при трансляции программы. Транслятор оптимизирует использование оборудования, также решая задачи распараллеливания. Это выражается в формировании "длинных" командных слов, задающих работы устройствам АЛУ в каждом машинном такте.

Основная сложность распараллеливания заключается в соблюдении частичной упорядоченности распределяемых работ. Поэтому решение задач синхронизации параллельного вычислительного процесса сопровождает все усилия по организации совместной работы устройств. Это сказывается и при решении всех задач эффективного использования расслоенной многоуровневой памяти ВС.

Отечественный опыт создания семейства МВК (многопроцессорных вычислительных комплексов) "Эльбрус", модели которого относятся к симметричным ВС, и последующее проектирование позволили проанализировать, разработать и применить ряд существенно новых, важных и перспективных решений в распараллеливании как самого вычислительного процесса, так и работы отдельных устройств.

Потоки команд и потоки данных

Общепринята удачная классификация ВС, которую предложил в 1966 г. М.Флинн (США). Основным определяющим архитектурным параметром он выбрал взаимодействие потока команд и потока данных (операндов и результатов).

В ЭВМ классической архитектуры ведется последовательная обработка команд и данных. Команды поступают одна за другой (за исключением точек ветвления программы), и для них из ОЗУ или регистров так же последовательно поступают операнды. Одной команде (операции) соответствует один необходимый ей набор операндов (как правило, два для бинарных операций). Этот тип архитектуры — " один поток команд — один поток данных", (SISD - "Single Instruction, Single Data") условно изображен на рис. 4.8. SISD — архитектура компьютера, в которой один процессор выполняет один поток команд, оперируя одним потоком данных. Относится к фон Неймановской архитектуре.

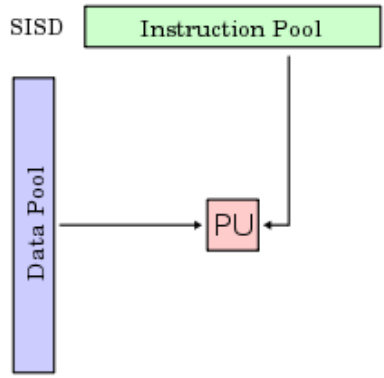


Рис 4.8. Структурная схема архитектуры SISD

SIMD — один поток команд, много потоков данных;

SIMD (англ. single instruction, multiple data — одиночный поток команд, множественный поток данных, SIMD) — принцип компьютерных вычислений, позволяющий обеспечить параллелизм на уровне данных.

SIMD-компьютеры состоят из одного командного процессора (управляющего модуля), называемого контроллером, и нескольких модулей обработки данных, называемых процессорными элементами. Управляющий модуль принимает, анализирует и выполняет команды. Если в команде встречаются данные, контроллер рассылает на все процессорные элементы команду, и эта команда выполняется на нескольких или на всех процессорных элементах. Каждый процессорный элемент имеет свою собственную память для хранения данных. Одним из преимуществ данной архитектуры считается то, что в этом случае более эффективно реализована логика вычислений. До половины логических инструкций обычного процессора связано с управлением выполнением машинных команд, а остальная их часть относится к работе с внутренней памятью процессора и выполнению арифметических операций. В SIMD компьютере управление выполняется контроллером, а «арифметика» отдана процессорным элементам.

SIMD-процессоры называются также векторными. Отечественные векторные ВС — ПС-2000, ПС-2100. Допускают организацию матричной обработки. Классический пример матричной архитектуры — ILLIAC-1V (США).

MISD — много потоков команд, один поток данных.

MISD-архитектура (англ. Multiple Instruction stream, Single Data stream, Множественный поток Команд, Одиночный поток Данных, MISD) — тип архитектуры параллельных вычислений, где несколько функциональных модулей (два или более) выполняют различные операции над одними данными.

Отказоустойчивые компьютеры, выполняющие одни и те же команды избыточно с целью обнаружения ошибок, как следует из определения, принадлежат к этому типу. К этому типу иногда относят конвейерную архитектуру, но не все с этим согласны, так как данные будут различаться после обработки на каждой стадии в конвейере. Некоторые относят систолический массив процессоров к архитектуре MISD.

Было создано немного ЭВМ с MISD-архитектурой, поскольку MIMD и SIMD чаще всего являются более подходящими для общих методик параллельных данных. Они обеспечивают лучшее масштабирование и использование вычислительных ресурсов, чем архитектура MISD. Тип архитектуры MISD условно изображен на рис. 4.9.

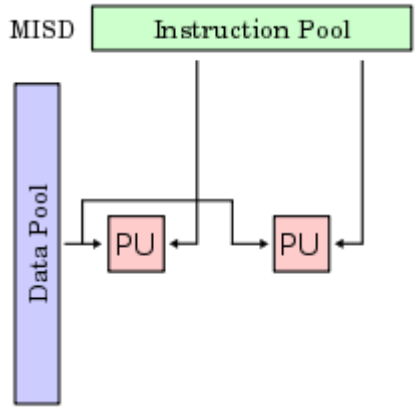


Рис 4.9. Структурная схема архитектуры MISD

К типу MISD принято относить векторный конвейер (обычно в составе ВС, чтобы подчеркнуть основной используемый принцип вычислений), например, в составе ВС Crey -1, "Электроника ССБИС". На векторном конвейере производится последовательная обработка одного потока данных многими обрабатывающими устройствами (ступенями, станциями) конвейера. К такому же типу относится ВС,

реализующая макроконвейер (ВС "Украина"). В ней задача, решаемая циклически, "разрезается" на последовательные этапы, закрепляемые за отдельными процессорами. Запускается конвейер многократного выполнения цикла, составляющего задачу.

MIMD — много потоков команд, много потоков данных (англ. Multiple Instruction stream, Multiple Data stream — Множественный поток Команд, Множественный поток Данных, сокращённо MIMD) — концепция архитектуры компьютера, используемая для достижения параллелизма вычислений. Машины имеют несколько процессоров, которые функционируют асинхронно и независимо. В любой момент, различные процессоры могут выполнять различные команды над различными частями данных. MIMD-архитектуры могут быть использованы в целом ряде областей, таких как системы автоматизированного проектирования автоматизированное производство, моделирование, а также комуникатор связей (communication switches). MIMD машины могут быть либо с общей памятью, либо с распределяемой памятью. Эта классификация основана на том, как MIMD-процессоры получают доступ к памяти. Этот класс предполагает, что в вычислительной системе есть несколько устройств обработки команд, объединенных в единый комплекс и работающих каждое со своим потоком команд и данных. Обработка разделена на несколько потоков, каждый с собственным аппаратным состоянием процессора, в рамках единственного определённого программным обеспечением процесса или в пределах множественных процессов. Поскольку система имеет несколько потоков, ожидающих выполнения (системные или пользовательские потоки), эта архитектура эффективно использует аппаратные ресурсы.

В MIMD могут возникнуть проблемы взаимной блокировки и состязания за обладание ресурсами, так как потоки, пытаясь получить доступ к ресурсам, могут столкнуться непредсказуемым способом. MIMD требует специального кодирования в операционной системе компьютера, но не требует изменений в прикладных программах, кроме случаев когда программы сами используют множественные потоки (MIMD прозрачен для однопоточных программ под управлением большинства операционных систем, если программы сами не отказываются от управления со стороны ОС). И системное и пользовательское программное обеспечение, возможно, должны использовать программные конструкции, такие как семафоры, чтобы препятствовать тому, чтобы один поток вмешался в другой в случае, если они содержат ссылку на одни и те же данные. Такое действие

увеличивает сложность кода, снижает производительность и значительно увеличивает количество необходимого тестирования, хотя обычно не настолько, чтобы свести на нет преимущества многопроцессорной обработки.

Подобные конфликты могут возникнуть на аппаратном уровне между процессорами, и должны решаться аппаратными средствами, или с помощью программного обеспечения и оборудования. Тип архитектуры MIMD условно изображен на рис. 4.10.

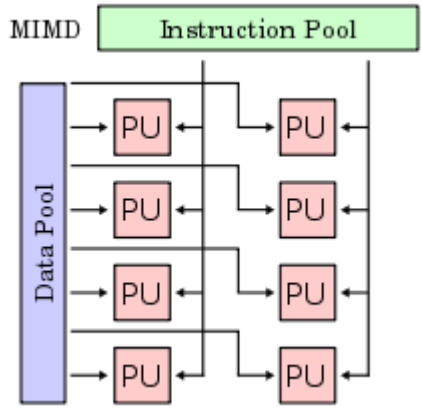


Рис 4.10. Структурная схема архитектуры MIMD

Таким образом, архитектурный MIMD соответствует более полному и независимому распараллеливанию. К этому типу относятся, например, все многопроцессорные вычислительные комплексы семейства "Эльбрус".

4.7. Вопросы и задания для самоконтроля

1. Что такое архитектура? Что этот термин обозначает в различных системах?
2. Сформулируйте основные признаки фон Неймановской архитектуры.
3. Опишите гарвардскую архитектуру и ее особенности.
4. В чём преимущества Гарвардской архитектуры относительно фон Неймановской?
5. В чём вы видите причины того, что максимального распространения достигла не гарвардская, а фон Неймановская

архитектура? В каких областях есть место для применения гарвардской архитектуры?

6. Назовите этапы развития архитектур и их различия.
7. Опишите «не фон Неймановские» архитектуры и их особенности.
8. Что такое «семантический разрыв»?
9. Что такое параллельная архитектура?
10. Опишите архитектуру SISD и её особенности.
11. Опишите архитектуру MISD и её особенности.
12. Опишите архитектуру SIMD и её особенности.
13. Опишите архитектуру MIMD и её особенности.
14. Какому типу архитектуры по классификации Флинна соответствует компьютер фон Неймана? А какому соответствует компьютер Гарвардской архитектуры?
15. В вычислительной системе 1 выполнение команды занимает 10 нс, в то время как в вычислительной системе 2 выполнение команды занимает 4 нс. Можно ли с уверенностью сказать, что ВС1 работает быстрее? Аргументируйте ответ.
16. Когда развитие технологии позволило размещать больше транзисторов на каждой микросхеме, Intel и AMD решили начать изготовление многоядерных процессоров для персональных компьютеров. Из какой категории в какую (согласно классификации Флинна) при этом перешли персональные компьютеры на этих процессорах?
17. Утром пчелиная матка созывает рабочих пчел и сообщает им, что сегодня нужно собрать нектар ноготков. Рабочие пчелы вылетают из улья и летят в разных направлениях в поисках ноготков. Что это за система: SIMD или MIMD?

5. ЦЕНТРАЛЬНЫЕ ПРОЦЕССОРЫ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

5.1. Классификация процессоров. Основные характеристики центральных процессоров

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

1. по используемой системе счисления:
   - работающие в позиционной системе счисления;
   - работающие в непозиционной системе счисления (например, СОК);
2. по способу обработки разрядов:
   - с параллельной обработкой разрядов;
   - с последовательной обработкой;
   - со смешанной обработкой (последовательно-параллельной);
3. по составу операций:
   - процессоры общего назначения;
   - проблемно-ориентированные;
   - специализированные;
4. по месту процессора в системе:
   - центральный процессор (ЦП);
   - сопроцессор;
   - периферийный процессор;
   - канальный процессор (контроллер канала ввода/вывода);
   - процессорный элемент (ПЭ) многопроцессорной системы;
5. по организации операционного устройства:
   - с операционным устройством процедурного типа (I-процессоры, M-процессоры);
   - процессоры с блочным операционным устройством;
   - процессоры с конвейерным операционным устройством (с арифметическим конвейером);
6. по организации обработки адресов:
   - с общим операционным устройством;
   - со специальным (адресным) операционным устройством;
7. по типу операндов:
   - скалярный процессор;

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

8. по логике управления процессором:
   - с жесткой логикой управления;
   - с микропрограммным управлением;
9. по составу (полноте) системы команд:
   - RISC;
   - CISC;
10. по организации управления потоком команд / способу загрузки исполнительных устройств:
    - с последовательной обработкой команд;
    - с конвейером команд;
    - суперскалярные процессоры;
    - процессоры с длинным командным словом (VLIW(очень длинная машинная команда)) и т. д.

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

Рассмотрим **основные характеристики** процессоров**.**

- Производительность.

Определяется скоростью выполнения команд программы. Используются показатели: количество миллионов операций в секунду над дробными числами.

- Тактовая частота.

Это количество циклов устройства в работе за единицу времени. Чем выше частота, тем выше скорость выполнения операции. Измеряется в ГГц.

- Разрядность.

Это число двоичных разрядов, обрабатываемых одновременно при выполнении одной команды: 16, 32, 64.

- Система команд.

Общее количество разновидностей команд (более 100-200).

- Наличие кэш-памяти.

Кэш-память располагается в процессоре, характеризуется быстродействием. Для ускорения доступа к данным, находящимся в оперативной памяти, используют кэш-память. Она имеет несколько уровней (первый, второй и третий). Кэш первого уровня имеет наименьший объем и максимальную скорость доступа к памяти.

- Параллельное выполнение команд.

Каждая команда выполняется процессором за несколько внутренних тактов, поэтому удобно организовать конвейер обработки команд, чтобы ускорить выполнение программы.

- Технология изготовления процессора.

Чем меньше размеры элементов процессора, тем больше его быстродействие. Для этого увеличивают плотность элементов в микросхеме, при этом уменьшая толщину проводников. Площадь кристалла уменьшилась при переходе от технологии 0.19 мкм на технологию 0.13 мкм.

5.2. Принципы работы центральных процессоров

Центральный процессор компьютера состоит из:

- блока управления процессором,
- регистров команд и данных,
- арифметико-логических устройств (выполняют арифметические и логические операции)
- блока операций с действительными числами, то есть с числами с плавающей точкой (FPU),
- буферной памяти (кэша) первого уровня (отдельно для команд и данных),
- буферной памяти (кэша) второго уровня для хранения промежуточных результатов вычислений,
- кэша третьего уровня (в большинстве современных процессоров),
- интерфейса системной шины.

Алгоритм работы центрального процессора компьютера можно представить, как последовательность следующих действий:

- блок управления процессором берет из оперативной памяти, в которую загружена программа, определенные значения (данные) и команды которые необходимо выполнить (инструкции). Эти данные загружаются в кэш-память процессора;
- из буферной памяти процессора (кэша) инструкции и полученные данные записываются в регистры. Инструкции помещаются в регистры команд, а значения в регистры данных;
- арифметико-логическое устройство считывает инструкции и данные из соответствующих регистров процессора и выполняет эти команды над полученными числами;
- результаты снова записываются в регистры и, если вычисления закончены, в буферную память процессора. Регистров у

процессора немного, поэтому он вынужден хранить промежуточные результаты в кэш-памяти различного уровня.

- новые данные и команды, необходимые для расчетов, загружаются в кэш верхнего уровня (из третьего во второй, из второго в первый), а неиспользуемые данные наоборот в кэш нижнего уровня;
- если цикл вычислений закончен, результат записывается в оперативную память компьютера для высвобождения места в буферной памяти процессора для новых вычислений. То же самой происходит при переполнении данными кэш-памяти: неиспользуемые данные перемещаются в кэш нижнего уровня или в оперативную память.

Последовательность этих операций образует операционный поток процессора.

Параллелизм на уровне команд

Разработчики компьютеров стремятся к тому, чтобы повысить производительность своих машин. Один из способов заставить процессоры работать быстрее - повышение их тактовой частоты, однако при этом существуют некоторые технологические ограничения, связанные с конкретным историческим периодом. Поэтому большинство разработчиков для повышения производительности при данной тактовой частоте процессора используют параллелизм (выполнение двух или более операций одновременно).

Существует две основные формы параллелизма: параллелизм на уровне команд и параллелизм на уровне процессоров. В первом случае параллелизм реализуется за счет запуска большого количества команд каждую секунду. Во втором случае над одним заданием работают одновременно несколько процессоров. Каждый подход имеет свои преимущества. В этом разделе мы рассмотрим параллелизм на уровне команд, а в следующем - параллелизм на уровне процессоров.

Конвейеры

Уже много лет известно, что главным препятствием высокой скорости выполнения команд является необходимость их вызова из памяти. Для разрешения этой проблемы можно вызывать команды из памяти заранее и хранить в специальном наборе регистров. Эта идея использовалась еще в 1959 году при разработке компьютера Stretch компании IBM, а набор регистров был назван буфером выборки с упреждением. Таким образом, когда требовалась определенная

команда, она вызывалась прямо из буфера, а обращения к памяти не происходило.

В действительности при выборке с упреждением команда обрабатывается за два шага: сначала происходит вызов команды, а затем ее выполнение. Еще больше продвинула эту стратегию идея конвейера. При использовании конвейера команда обрабатывается уже не за два, а за большее количество шагов, каждый из которых реализуется определенным аппаратным компонентом, причем все эти компоненты могут работать параллельно.

На рис. 5.1, а изображен конвейер из пяти блоков, которые называются ступенями. Первая ступень (блок С1) вызывает команду из памяти и помещает ее в буфер, где она хранится до тех пор, пока не потребуется. Вторая ступень (блок С2) декодирует эту команду, определяя ее тип и тип ее операндов. Третья ступень (блок СЗ) определяет местонахождение операндов и вызывает их из регистров или из памяти. Четвертая ступень (блок С4) выполняет команду, обычно проводя операнды через тракт данных. И наконец, блок С5 записывает результат обратно в нужный регистр.

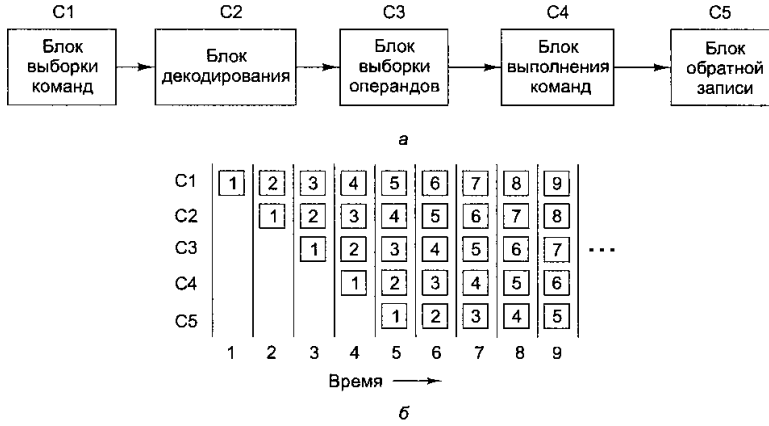


Рис 5.1. Пятиступенчатый конвейер (а);
состояние каждой ступени в зависимости от количества пройденных циклов (б). Показано 9 циклов

На рис. 5.1 б) представлено функционирование конвейера во времени. Во время цикла 1 блок Cl обрабатывает команду 1, вызывая ее из памяти. Во время цикла 2 блок С2 декодирует команду 1, в то время как блок Cl вызывает из памяти команду 2. Во время цикла 3

блок С3 вызывает операнды для команды 1, блок С2 декодирует команду 2, а блок Cl вызывает команду 3. Во время цикла 4 блок С4 выполняет команду 1, С3 вызывает операнды для команды 2, С2 декодирует команду 3, а Cl вызывает команду 4. Наконец, во время цикла 5 блок С5 записывает результат выполнения команды 1 обратно в регистр, тогда как другие ступени конвейера обрабатывают следующие команды.

Чтобы лучше понять принципы работы конвейера, представим себе кондитерскую фабрику, на которой выпечка тортов и их упаковка для отправки выполняются раздельно. Предположим, что в отделе отправки находится длинный конвейер, вдоль которого располагаются 5 рабочих (или ступеней обработки). Каждые 10 секунд (это время цикла) первый рабочий ставит пустую коробку для торта на ленту конвейера. Эта коробка отправляется ко второму рабочему, который кладет в нее торт. После этого коробка с тортом доставляется третьему рабочему, который закрывает и запечатывает ее. Затем она поступает к четвертому рабочему, который ставит на ней штамп. Наконец, пятый рабочий снимает коробку с конвейерной ленты и помещает ее в большой контейнер для отправки в супермаркет. Примерно таким же образом действует компьютерный конвейер: каждая команда (в случае с кондитерской фабрикой — торт) перед окончательным выполнением проходит несколько ступеней обработки.

Возвратимся к нашему конвейеру на рис. 5.1. Предположим, что время цикла у этой машины - 2 нс. Тогда для того, чтобы одна команда прошла через весь конвейер, требуется 10 нс. На первый взгляд может показаться, что такой компьютер будет выполнять 100 млн команд в секунду, в действительности же скорость его работы гораздо выше. В течение каждого цикла (2 нс) завершается выполнение одной новой команды, поэтому машина выполняет не 100, а 500 млн команд в секунду. Конвейеры позволяют добиться компромисса между временем запаздывания (время выполнения одной команды) и пропускной способностью процессора (количество команд, выполняемых процессором в секунду).

Суперскалярные архитектуры

Один конвейер — хорошо, а два — еще лучше. Одна из возможных схем процессора с двумя конвейерами показана на рис. 5.2. В ее основе лежит конвейер, изображенный на рис. 5.1. Здесь общий блок выборки команд вызывает из памяти сразу по две команды и помещает каждую из них в один из конвейеров. Каждый конвейер содержит АЛУ для параллельных операций. Чтобы выполняться

параллельно, две команды не должны конфликтовать из-за ресурсов (например, регистров), и ни одна из них не должна зависеть от результата выполнения другой. Как и в случае с одним конвейером, либо компилятор должен гарантировать отсутствие нештатных ситуаций (когда, например, аппаратура не обеспечивает проверку команд на несовместимость и при обработке таких команд выдает некорректный результат), либо за счет дополнительной аппаратуры конфликты должны выявляться и устраняться непосредственно в ходе выполнения команд.

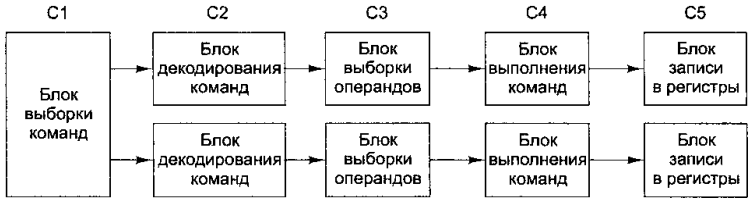


Рис 5.2. Сдвоенный пятиступенчатый конвейер с общим блоком выборки команд

Сначала конвейеры (как сдвоенные, так и обычные) использовались только в RISC-компьютерах. У процессора 386 и его предшественников их не было. Конвейеры в процессорах компании Intel появились, только начиная с модели 486. Процессор 486 имел один пятиступенчатый конвейер, а Pentium - два таких конвейера. Похожая схема изображена на рис. 5.2, но разделение функций между второй и третьей ступенями (они назывались декодер 1 и декодер 2) было немного другим. Главный конвейер (u-конвейер) мог выполнять произвольные команды. Второй конвейер (v-конвейер) мог выполнять только простые команды с целыми числами, а также одну простую команду с плавающей точкой (FXCH).

Имеются сложные правила определения, является ли пара команд совместимой в отношении возможности параллельного выполнения. Если команды, входящие в пару, были сложными или несовместимыми, выполнялась только одна из них (в u-конвейере). Оставшаяся вторая команда составляла затем пару со следующей командой. Команды всегда выполнялись по порядку. Таким образом, процессор Pentium содержал особые компиляторы, которые объединяли совместимые команды в пары и могли порождать программы, выполняющиеся быстрее, чем в предыдущих версиях. Необходимо отметить, что параллельное функционирование

отдельных блоков процессора имело место и в предыдущем микропроцессоре (386). Этот механизм стал прообразом 5-ступенчатого конвейера.

Переход к четырем конвейерам возможен, но требует громоздкого аппаратного обеспечения. Вместо этого используется другой подход. Основная идея — один конвейер с большим количеством функциональных блоков, как показано на рис. 5.3. В 1987 году для обозначения этого подхода был введен термин суперскалярная архитектура. Однако подобная идея нашла воплощение еще тридцатью годами ранее в компьютере CDC 6600. Этот компьютер вызывал команду из памяти каждые 100 нс и помещал ее в один из 10 функциональных блоков для параллельного выполнения. Пока команды выполнялись, центральный процессор вызывал следующую команду.

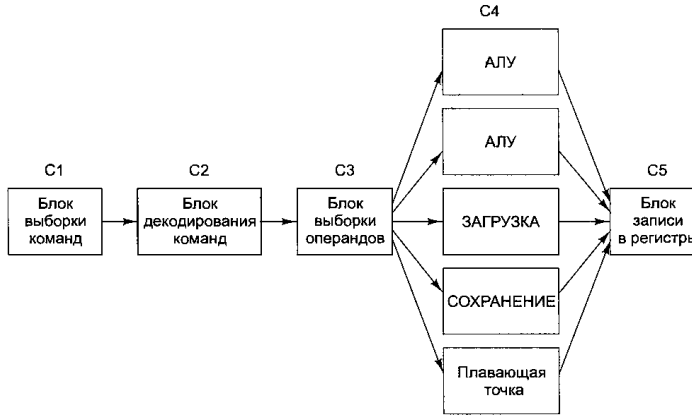


Рис 5.3. Суперскалярный процессор с пятью функциональными блоками

Со временем значение понятия "суперскалярный" несколько изменилось. Теперь суперскалярными называют процессоры, способные запускать несколько команд (зачастую от четырех до шести) за один тактовый цикл. Естественно, чтобы передавать все эти команды, в суперскалярном процессоре должно быть несколько функциональных блоков. Поскольку в процессорах этого типа, как правило, предусматривается один конвейер, его устройство обычно соответствует рис. 5.3.

В свете такой терминологической динамики на сегодняшний день можно утверждать, что компьютер 6600 не был суперскалярным с

технической точки зрения: ведь за один тактовый цикл в нем запускалось не больше одной команды. Однако при этом был достигнут аналогичный результат: команды запускались быстрее, чем выполнялись. На самом деле разница в производительности между ЦП с циклом в 100 не, передающим за этот период по одной команде четырем функциональным блокам, и ЦП с циклом в 400 нс, запускающим за это время четыре команды, трудноуловима. В обоих процессорах соблюдается принцип превышения скорости запуска над скоростью управления; при этом рабочая нагрузка распределяется между несколькими функциональными блоками.

Отметим, что на выходе ступени 3 команды появляются значительно быстрее, чем ступень 4 способна их обрабатывать. Если бы на выходе ступени 3 команды появлялись каждые 10 нс, а все функциональные блоки делали свою работу также за 10 нс, то на ступени 4 всегда функционировал бы только один блок, что сделало бы саму идею конвейера бессмысленной. В действительности большинству функциональных блоков ступени 4 (точнее, обоим блокам доступа к памяти и блоку выполнения операций с плавающей точкой) для обработки команды требуется значительно больше времени, чем занимает один цикл. Как видно из рис. 5.3, на ступени 4 может быть несколько АЛУ.

Параллелизм на уровне процессоров

Спрос на компьютеры, работающие с все более и более высокой скоростью, не прекращается. Быстродействие процессоров растет, но у них постоянно возникают проблемы со скоростью передачи информации, поскольку скорость распространения электромагнитных волн в медных проводах и света в оптико-волоконных кабелях по-прежнему остается равной 20 см/нс, независимо от того, насколько умны инженеры компании Intel. Кроме того, чем быстрее работает процессор, тем сильнее он нагревается, поэтому возникает задача защиты его от перегрева.

Параллелизм на уровне команд в определенной степени помогает, но конвейеры и суперскалярная архитектура обычно повышают скорость работы всего лишь в 5-10 раз. Чтобы увеличить производительность в 50, 100 и более раз, нужно создавать компьютеры с несколькими процессорами. Ознакомимся с устройством таких компьютеров.

Матричные процессоры

Многие задачи в физических и технических науках предполагают использование массивов или других упорядоченных структур. Часто одни и те же вычисления могут производиться над разными наборами данных в одно и то же время. Упорядоченность и структурированность программ, предназначенных для выполнения такого рода вычислений, очень удобны в плане ускорения вычислений за счет параллельной обработки команд. Существует две схемы ускоренного выполнения больших научных программ. Хотя обе схемы во многих отношениях схожи, одна из них предполагает расширение единственного процессора, другая — добавление параллельного вычислителя.

Матричный процессор (array processor) состоит из большого числа сходных процессоров, которые выполняют одну и ту же последовательность команд применительно к разным наборам данных. Первым в мире таким процессором был ILLIAC IV (Университет Иллинойс). Схематически он изображен на рис. 5.4.

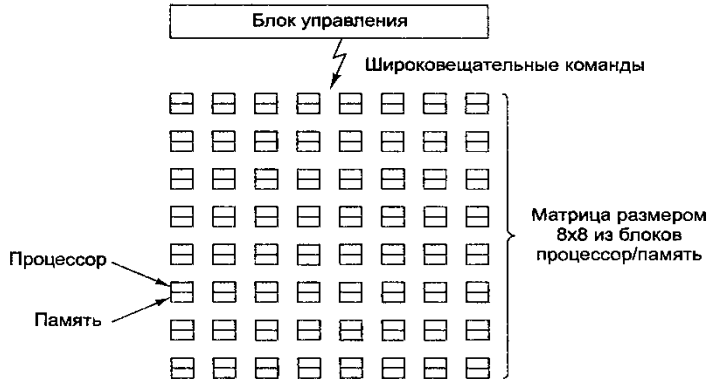


Рис 5.4. Матричный процессор ILLIAC IV

Первоначально предполагалось сконструировать машину, состоящую из четырех квадрантов, каждый из которых содержал бы матрицу размером 8 х 8 из блоков процессор/память. Для каждого квадранта имелся один блок контроля. Он рассылал команды, которые выполнялись всеми процессорами одновременно, при этом каждый процессор использовал собственные данные из собственной памяти (загрузка данных происходила при инициализации). Это решение, значительно отличающееся от стандартной фон-неймановской

машины, иногда называют архитектурой SIMD. Из-за очень высокой стоимости был построен только один такой квадрант, но он мог выполнять 50 млн операций с плавающей точкой в секунду. Если бы при создании машины использовалось четыре квадранта, она могла бы выполнять 1 млрд операций с плавающей точкой в секунду, и вычислительные возможности такой машины в два раза превышали бы возможности компьютеров всего мира.

С точки зрения программиста векторный процессор (vector processor) очень похож на матричный. Как и матричный, он чрезвычайно эффективен при выполнении последовательности операций над парами элементов данных. Однако в отличие от матричного процессора, все операции сложения выполняются в одном блоке суммирования, который имеет конвейерную структуру. Компания Cray Research, основателем которой был Сеймур Крей, выпустила множество моделей векторных процессоров, начиная с модели Cray-1 (1974).

Оба типа процессоров работают с массивами данных. Оба они выполняют одни и те же команды, которые, например, попарно складывают элементы двух векторов. Однако если у матричного процессора столько же суммирующих устройств, сколько элементов в массиве, векторный процессор содержит векторный регистр, состоящий из набора условных регистров. Эти регистры загружаются из памяти единственной командой, которая фактически делает это последовательно. Команда сложения попарно складывает элементы двух таких векторов, загружая их из двух векторных регистров в суммирующее устройство с конвейерной структурой. В результате из суммирующего устройства выходит другой вектор, который либо помещается в векторный регистр, либо сразу используется в качестве операнда при выполнении другой операции с векторами.

Матричные процессоры в настоящее время не выпускаются, но принцип, на котором они основаны, по-прежнему актуален. Аналогичная идея применяется в наборах MMX- и SSE-команд современных процессоров, и она успешно решает задачу ускоренного выполнения мультимедийных программ. В этом отношении компьютер ILLIAC IV можно считать одним из прародителей современных процессоров.

Мультипроцессоры

Элементы матричного процессора связаны между собой, поскольку их работу контролирует единый блок управления. Система из нескольких параллельных процессоров, имеющих общую память,

называется мультипроцессором. Поскольку каждый процессор может записы вать информацию в любую часть памяти и считывать информацию из любой части памяти, чтобы не допустить каких-либо нестыковок, их работа должна согласовываться программным обеспечением. В ситуации, когда два или несколько процессоров имеют возможность тесного взаимодействия, а именно так происходит в случае с мультипроцессорами, эти процессоры называют сильно связанными (tightly coupled).

Возможны разные способы воплощения этой идеи. Самый простой из них — соединение единственной шиной нескольких процессоров и общей памяти. Схема такого мультипроцессора показана на рис. 5.5, а. Подобные системы производят многие компании.

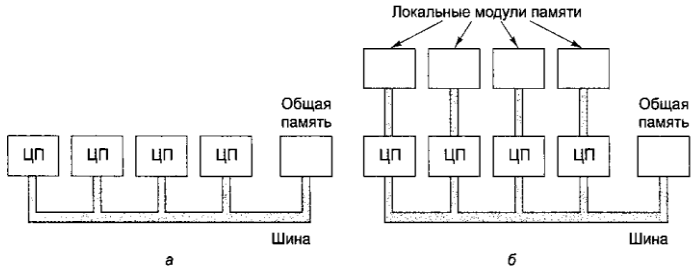


Рис 5.5. Мультипроцессор с единственной шиной и общей памятью (а); мультипроцессор с собственной локальной памятью для каждого процессора (б)

Нетрудно понять, что при наличии большого числа быстродействующих процессоров, которые постоянно пытаются получить доступ к памяти через одну и ту же шину, будут возникать конфликты. Чтобы разрешить эту проблему и повысить производительность компьютера, разработаны различные схемы. Одна из них изображена на рис. 5.5, б. В таком компьютере каждый процессор имеет собственную локальную память, недоступную для других процессоров. Эта память используется для тех программ и данных, которые не нужно разделять между несколькими процессорами. При доступе к локальной памяти основная шина не используется, и, таким образом, объем передаваемой по ней информации становится меньше. Возможны и другие варианты решения проблемы (например, использование кэш-памяти).

Мультипроцессоры имеют преимущество перед другими видами параллельных компьютеров, поскольку с единой общей памятью очень

легко работать. Например, представим, что программа ищет раковые клетки на сделанном через микроскоп снимке ткани. Фотография в цифровом виде может храниться в общей памяти, при этом каждый процессор будет обследовать какую-нибудь определенную область фотографии. Поскольку каждый процессор имеет доступ к общей памяти, обследование клетки, расположенной сразу в нескольких областях, не представляет трудностей.

Мультикомпьютеры

Мультипроцессоры с небольшим числом процессоров (< 256) разрабатывать достаточно просто, а вот создание больших мультипроцессоров представляет определенные трудности. Сложность заключается в том, чтобы связать все процессоры с общей памятью. Поэтому многие разработчики просто отказались от идеи разделения памяти и стали создавать системы без общей памяти, состоящие из большого числа взаимосвязанных компьютеров, у каждого из которых имеется собственная память. Такие системы называются мультикомпьютерами. В них процессоры являются слабо связанными, в противоположность сильно связанным процессорам в мультипроцессорных системах.

Процессоры мультикомпьютера отправляют друг другу сообщения (это несколько похоже на электронную почту, но гораздо быстрее). Каждый компьютер не обязательно соединять со всеми другими, поэтому обычно в качестве топологий используются двух- и трехмерные решетки, а также деревья и кольца. Хотя на пути до места назначения сообщения проходят через один или несколько промежуточных компьютеров, время передачи занимает всего несколько микросекунд. Уже работают мультикомпьютеры, содержащие до 10 000 процессоров.

Поскольку мультипроцессоры легче программировать, а мультикомпьютеры — конструировать, появилась идея создания гибридных систем, сочетающих в себе достоинства обеих топологий. Такие компьютеры представляют иллюзию общей памяти, при этом в действительности она не существует.

Выборка и исполнение команды

В начале каждого цикла процессор выбирает из памяти команду. Обычно адрес ячейки, из которой нужно извлечь очередную команду, хранится в программном счетчике (PC). Если не указано иное, после извлечения каждой команды процессор увеличивает значение программного счетчика на единицу. Таким образом, команды

выполняются в порядке возрастания номеров ячеек памяти, в которых они хранятся. Рассмотрим, например, упрощенный компьютер, в котором каждая команда занимает одно 16-битовое слово памяти. Предположим, что значение программного счетчика установлено равным 300. Это значит, что следующая команда, которую должен извлечь процессор, находится в 300-й ячейке. При успешном завершении цикла команды процессор перейдет к извлечению команд из ячеек 301, 302, 303 и т.д. Однако, эта последовательность может быть изменена.

Извлеченные команды загружаются в регистр команд. Команда состоит из последовательности битов, указывающих процессору, какие именно действия он должен выполнить. Процессор интерпретирует команду и выполняет требуемые действия. Все действия можно разбить на четыре категории:

1) процессор — память. Данные передаются из процессора в память или обратно;

2) процессор — устройства ввода-вывода. Данные из процессора поступают на периферийное устройство через устройство ввода-вывода. Возможен и обратный процесс;

3) обработка данных. Процессор выполняет с данными различные арифметические или логические операции;

4) управление. Команда может задавать изменение последовательности выполнения команд. Например, если процессор извлекает из ячейки 149 команду, которая указывает, что следующей по очереди должна быть исполнена команда из ячейки 182, то процессор устанавливает значение программного счетчика равным 182. Таким образом, в следующем цикле выборки команда извлекается не из ячейки 150, а из ячейки 182.

Для выполнения команды может потребоваться последовательность, состоящая из комбинации вышеперечисленных действий.

5.3. Понятие архитектуры системы команд. Классификация архитектур

Системой команд вычислительной машины называют полный перечень команд, которые способна выполнять данная ЭВМ. В свою очередь, под архитектурой системы команд (АСК) принято определять те средства вычислительной машины, которые видны и доступны программисту. АСК можно рассматривать как линию согласования нужд разработчиков программного обеспечения с возможностями

создателей аппаратуры вычислительной машины. Таким образом, АСК служит интерфейсом между программной и аппаратной частями компьютера (см. рис. 5.6).

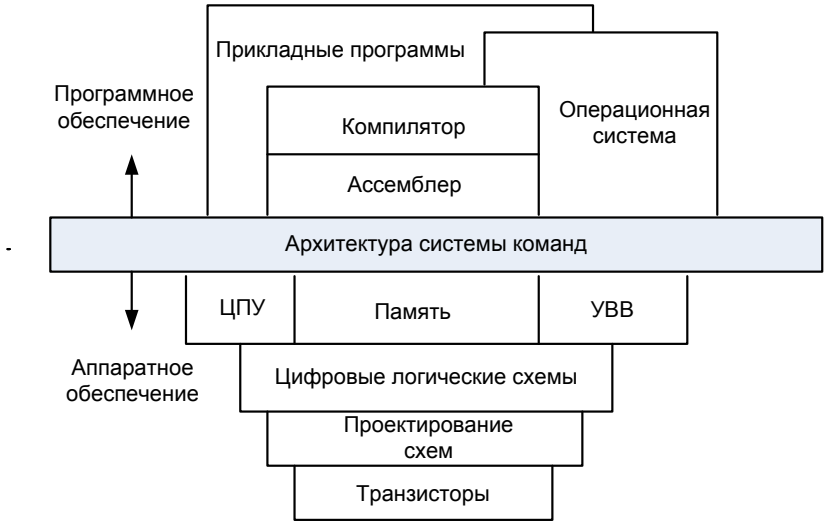


Рис 5.6. Основные компоненты компьютера

В конечном итоге цель разработчиков программного обеспечения и создателей аппаратуры — реализация вычислений наиболее эффективным образом, то есть за минимальное время, и здесь важнейшую роль играет правильный выбор архитектуры системы команд.

В упрощенной трактовке время выполнения программы $T_{выч}$ можно определить через число команд в программе $N_{ком}$, среднее количество тактов процессора CPI, приходящихся на одну команду и длительность тактового периода $\tau_{тп}$:

$$T_{выч} = N_{ком} * CPI * \tau_{тп}.$$

Каждая из составляющих данного выражения зависит от одних аспектов архитектуры системы команд и, в свою очередь, влияет на другие (см. рисунок 5.7), что свидетельствует о необходимости ответственного подхода к выбору АСК.

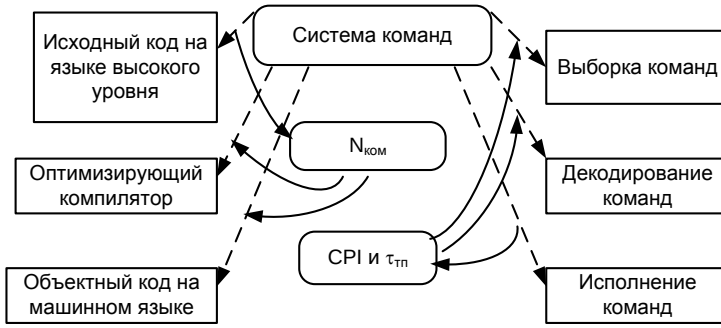


Рис 5.7. АСК и эффективность вычислений

В истории развития вычислительной техники отражаются изменения, происходившие во взглядах разработчиков на перспективность той или иной архитектуры системы команд. Сложившуюся на настоящий момент ситуацию в области АСК иллюстрирует рисунок 5.8.

Рис 5.8. Классификация АСК

Среди мотивов, чаще всего определяющих переход к новому типу АСК, остановимся на двух наиболее существенных. Первый — **место хранения операндов**, что влияет на количество и длину адресов, указываемых в адресной части команд обработки данных. Второй — это **состав операций**, выполняемых вычислительной машиной, и их сложность. Именно эти моменты взяты в качестве критериев для приведенной классификации архитектур системы команд. Важную роль при выборе АСК играет ответ на вопрос о том, где могут храниться операнды и каким образом к ним осуществляется доступ. С этих позиций различают следующие виды архитектур системы команд:

- стековую;
- аккумуляторную;
- регистровую;
- с выделенным доступом к памяти.

Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой будет общая длина команд.

5.4. CISC процессоры. RISC процессоры. MISC процессоры

CISC (англ. Complex instruction set computing, или англ. complex instruction set computer — компьютер с полным набором команд) — концепция проектирования процессоров, которая характеризуется следующим набором свойств:

- нефиксированное значение длины команды;
- арифметические действия кодируются в одной команде;
- небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Типичными представителями являются процессоры на основе команд x86 (исключая современные Intel Pentium 4, Pentium D, Core, AMD Athlon, Phenom, которые являются гибридными) и процессоры Motorola MC680x0.

Наиболее распространённая архитектура современных настольных, серверных и мобильных процессоров построена по архитектуре Intel x86 (или x86-64 в случае 64-разрядных процессоров). Формально, все x86-процессоры являлись CISC-процессорами, однако новые процессоры, начиная с Intel Pentium Pro, являются CISC-процессорами с RISC-ядром. Они непосредственно перед исполнением

преобразуют CISC-инструкции процессоров x86 в более простой набор внутренних инструкций RISC.

В микропроцессор встраивается аппаратный транслятор, превращающий команды x86 в команды внутреннего RISC-процессора. При этом одна команда x86 может порождать несколько RISC-команд (в случае процессоров типа P6 — до четырёх RISC-команд в большинстве случаев). Исполнение команд происходит на суперскалярном конвейере одновременно по несколько штук.

Это потребовалось для увеличения скорости обработки CISC-команд, так как известно, что любой CISC-процессор уступает RISC-процессорам по количеству выполняемых операций в секунду. В итоге, такой подход и позволил поднять производительность CPU.

Недостатки CISC архитектуры:

- высокая стоимость аппаратной части;
- сложности с распараллеливанием вычислений.

Методика построения системы команд CISC противоположна другой методике - RISC. Различие этих концепций состоит в методах программирования, а не в реальной архитектуре процессора. Практически все современные процессоры эмулируют наборы команд как RISC, так и CISC типа.

В рабочих станциях, серверах среднего звена и персональных компьютерах используются процессоры с CISC. Наиболее распространенная архитектура команд процессоров мобильных устройств - SOC и мэйнфреймов - RISC. В микроконтроллерах различных устройств RISC используется в подавляющем большинстве случаев.

RISC (англ. restricted (reduced) instruction set computer — компьютер с сокращённым набором команд) — архитектура процессора, в которой быстродействие увеличивается за счёт упрощения инструкций, чтобы их декодирование было более простым, а время выполнения — меньшим. Первые RISC-процессоры даже не имели инструкций умножения и деления. Это также облегчает повышение тактовой частоты и делает более эффективной суперскалярность (распараллеливание инструкций между несколькими исполнительными блоками).

Наборы инструкций в более ранних архитектурах, для облегчения ручного написания программ на языках ассемблеров или прямо в машинных кодах, а также для упрощения реализации компиляторов, выполняли как можно больше работы. Нередко в наборы включались инструкции для прямой поддержки конструкций языков высокого уровня. Другая особенность этих наборов —

большинство инструкций, как правило, допускали все возможные методы адресации (т. н. «ортогональность системы команд (англ.)») — к примеру, и операнды, и результат в арифметических операциях доступны не только в регистрах, но и через непосредственную адресацию, и прямо в памяти. Позднее такие архитектуры были названы CISC (англ. Complex instruction set computer).

Однако многие компиляторы не задействовали все возможности таких наборов инструкций, а на сложные методы адресации уходит много времени из-за дополнительных обращений к медленной памяти. Было показано, что такие функции лучше исполнять последовательностью более простых инструкций, если при этом процессор упрощается и в нём остаётся место для большего числа регистров, за счёт которых можно сократить количество обращений к памяти. В первых архитектурах, причисляемых к RISC, большинство инструкций для упрощения декодирования имеют одинаковую длину и похожую структуру, арифметические операции работают только с регистрами, а работа с памятью идёт через отдельные инструкции загрузки (load) и сохранения (store). Эти свойства и позволили лучше сбалансировать этапы конвейеризации, сделав конвейеры в RISC значительно более эффективными и позволив поднять тактовую частоту.

Количество инструкций

Нередко слова «сокращённый набор команд» понимаются как минимизация количества инструкций в системе команд. В действительности, инструкций у многих RISC-процессоров больше, чем у CISC-процессоров. Некоторые RISC-процессоры вроде транспьютеров фирмы INMOS (англ.) имеют наборы команд не меньше, чем, например, у CISC-процессоров IBM System/370; и наоборот — CISC-процессор DEC PDP-8 имеет только 8 основных и несколько расширенных инструкций.

На самом деле, термин «сокращённый» в названии описывает тот факт, что сокращён объём (и время) работы, выполняемый каждой отдельной инструкцией — как максимум один цикл доступа к памяти, — тогда как сложные инструкции CISC-процессоров могут требовать сотен циклов доступа к памяти для своего выполнения.

Некоторые архитектуры, специально разработанные для минимизации количества инструкций, сильно отличаются от классических RISC-архитектур и получили другие названия: Minimal instruction set computer (MISC), Zero instruction set computer

(Компьютер с нулевым набором команд), Ultimate RISC (также называемый OISC), Transport triggered architecture (TTA) и т. п.

Характерные особенности RISC-процессоров:

- Фиксированная длина машинных инструкций (например, 32 бита) и простой формат команды.
- Специализированные команды для операций с памятью — чтения или записи. Операции вида Read-Modify-Write («прочитать-изменить-записать») отсутствуют. Любые операции «изменить» выполняются только над содержимым регистров (т. н. архитектура load-and-store).
- Большое количество регистров общего назначения (32 и более).
- Отсутствие поддержки операций вида «изменить» над укороченными типами данных — байт, 16-битное слово. Так, например, система команд DEC Alpha содержала только операции над 64-битными словами, и требовала разработки и последующего вызова процедур для выполнения операций над байтами, 16- и 32-битными словами.
- Отсутствие микропрограмм внутри самого процессора. То, что в CISC-процессоре исполняется микропрограммами, в RISC-процессоре исполняется как обыкновенный (хотя и помещённый в специальное хранилище) машинный код, не отличающийся принципиально от кода ядра ОС и приложений. Так, например, обработка отказов страниц в DEC Alpha и интерпретация таблиц страниц содержалась в так называемом PALcode (Privileged Architecture Library), помещённом в ПЗУ. Заменой PALCode можно было превратить процессор Alpha из 64-битного в 32-битный, а также изменить порядок байтов в слове и формат входов таблиц страниц виртуальной памяти.

MISC (англ. minimal instruction set computer — «компьютер с минимальным набором команд») — вид процессорной архитектуры.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно большое слово (связку, bound). Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных. Кроме этого MISC использует стековую модель вычислительного устройства и основные команды работы со стеком языка Forth.

Процессоры с MISC, как и процессоры RISC, характеризуются небольшим числом чаще всего встречающихся команд. Вместе с этим, принцип «очень длинных командных слов» (VLIW) обеспечивает

выполнение группы непротиворечивых команд за один цикл работы процессора. Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных. Таким образом архитектура MISC объединила вместе суперскалярную и VLIW-концепции. Компоненты процессора просты и работают на высоких частотах.

5.5. Теговая архитектура. Разрядно-модульная архитектура

В различных архитектурах типы данных определяются программой, иначе говоря, типы данных определяются кодом операции, что приводит к необходимости использовать разные команды для обработки различных типов данных. Каждый тип данных предусматривает наличие специального подмножества команд.

Принцип теговой архитектуры заключается в самоопределение данных, то есть каждая ячейка памяти имеет специальное дополнительное поле (тег), которое служит для описания атрибутов его содержимого. Это позволяет получить инвариантность команд к типам используемых данных. Сокращается число и длина команд. Определение типа данных откладывается на момент выборки этих данных, а не на момент выборки команды. Вполне может быть, что возникают исключительные ситуации, связанные с несоответствием типов командам, как правило, это является ошибкой программирования.

Расширенный принцип тегирования

Заключается в задании не только типа хранимых данных, но и длины операнда, занятости ячейки и т.д. Используются дополнительные флаги:

- Бит занятости. Определяет, свободна ячейка или занята.
- Бит захвата. Необходимость выполнения прерывания при доступе к ячейке.
- Биты защиты — определяет, какие обращения допустимы к этой «порции» данных (только чтение, только запись, чтение/запись).
- Длина данных.

Преимущества механизма тегирования:

- Компактная система команд.
- Строгий контроль типов.
- Более простые компиляторы. Меньший объем кода.
- Уменьшение интенсивности пересылки основная память — процессор, т.к. уменьшается код программы, загружающий интерфейс

на 50%, а увеличивается объем читаемых данных, загружающий интерфейс на 30%.

Недостатки теговой организации:

- Строгий контроль типов снижает выразительные способности языка.
- Уменьшается быстродействие процессора за счет откладывания привязки атрибутов данных на этап выполнения команды. В момент дешифрации команды сразу определяются типы данных, здесь же в момент дешифрации команды типы данных не определяются и поэтому то время, которое связано с подгрузкой операнда могло быть использовано для определения типов данных, что и происходит. Т.е. сокращено время дешифрации команды, но доступ к памяти медленный и снижается быстродействие процессора.
- Дополнительные расходы основной памяти в области данных для хранения полей тега.

Разрядно-модульная архитектура.

Большинство известных процессоров имеют фиксированную разрядность выполнения операций: 8,16,32 или 64 разряда. При необходимости изменение разрядности обработки данных осуществляется программным путем, то есть за счет усложнения программы. В то же время выпускаются процессоры, состоящие из отдельных разрядных секций, и допускающие аппаратное наращивание разрядности обрабатываемых данных путем параллельного включения нужного количества таких секций. Аналогичным образом решается задача расширения адресного пространства микропрограммируемого процессора — применением требуемого числа секций устройства управления адресацией микрокоманд.

5.6. Вопросы и задания для самоконтроля

Список вопросов:

1. Что такое процессор? Опишите классификацию процессоров.
2. Перечислите основные характеристики процессоров.
3. Перечислите основные компоненты ЦП компьютера.
4. Опишите алгоритм работы ЦП.
5. Опишите принципы параллелизма на уровне команд и процессов.
6. Как реализована выборка и исполнение команд?
7. Матричные компьютеры и принцип их работы.
8. Что такое мультипроцессоры и мультикомпьютеры?
9. Что такое система команд? Что такое архитектура системы команд?
10. Опишите CISC процессоры. Назовите их достоинства и недостатки.
11. Опишите RISC процессоры. Перечислите их особенности и отличия от CISC процессоров?
12. Опишите MISC процессоры. В чём их отличие от RISC процессоров?
13. Опишите понятие «матричный процессор».
14. Что такое «процессор с конвейерной обработкой»? Пример типового конвейера команд.
15. Какие виды процессоров с конвейерной обработкой существуют?
16. Что такое теговая архитектура? Расширенная теговая архитектура? Недостатки теговой архитектуры.
17. Что входит в понятие «сигнальные процессоры»? Перечислите особенности этих процессоров.
18. Опишите свойства систолической структуры. Где она применяется?
19. Что такое суперскалярный процессор?
20. В чём отличия VLIW-машин от суперскалярных процессоров?

6. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПАМЯТИ В ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ И СИСТЕМАХ

6.1. Основная память в ВМ и ВС

Основная память - это устройство для хранения информации. Она состоит из оперативного запоминающего устройства (ОЗУ) и постоянного запоминающего устройства (ПЗУ).

Оперативное запоминающее устройство (ОЗУ)

ОЗУ — быстрая, полупроводниковая, энергозависимая память. В ОЗУ хранятся исполняемая в данный момент программа и данные, с которыми она непосредственно работает. Это значит, что когда вы запускаете какую-либо компьютерную программу, находящуюся на диске, она копируется в оперативную память, после чего процессор начинает выполнять команды, изложенные в этой программе. Часть ОЗУ, называемая "видеопамять", содержит данные, соответствующие текущему изображению на экране. При отключении питания содержимое ОЗУ стирается.

Быстродействие (скорость работы) компьютера напрямую зависит от величины его ОЗУ, которое в современных компьютерах может доходить до 128 Мбайт. В первых моделях компьютеров оперативная память составляла не более 1 Мбайт. Современные прикладные программы часто требуют для своего выполнения не менее 4 Мбайт ОЗУ; в противном случае они просто не запускаются.

ОЗУ - это память, используемая как для чтения, так и для записи информации. При отключении электропитания информация в ОЗУ исчезает (энергозависимость).

Постоянное запоминающее устройство (ПЗУ)

ПЗУ - быстрая, энергонезависимая память. ПЗУ - это память, предназначенная только для чтения. Информация заносится в нее один раз (обычно в заводских условиях) и сохраняется постоянно (при включенном и выключенном компьютере). В ПЗУ хранится информация, присутствие которой постоянно необходимо в компьютере.

В ПЗУ находятся:

- тестовые программы, проверяющие при каждом включении компьютера правильность работы его блоков;
- программы для управления основными периферийными устройствами -дисководом, монитором, клавиатурой;

- информация о том, где на диске расположена операционная система.

Основная память состоит из регистров. Регистр - это устройство для временного запоминания информации в оцифрованной (двоичной) форме. Запоминающим элементом в регистре является триггер - устройство, которое может находиться в одном из двух состояний, одно из которых соответствует запоминанию двоичного нуля, другое - запоминанию двоичной единицы. Триггер представляет собой крошечный конденсатор-батарейку, которую можно заряжать множество раз. Если такой конденсатор заряжен - он как бы запомнил значение "1", если заряд отсутствует - значение "0". Регистр содержит несколько связанных друг с другом триггеров. Число триггеров в регистре называется разрядностью компьютера. Производительность компьютера напрямую связана с разрядностью, которая бывает равной 8, 16, 32, 64, 128.

6.2. Классификация памяти по специфике использования (СОЗУ, ОЗУ, ПЗУ, ППЗУ)

По специфике использования в ЭВМ различают ЗУ с изменением информации в процессе работы ЭВМ, с редким изменением информации в процессе эксплуатации ЭВМ и без изменения информации в процессе эксплуатации ЭВМ. Информация систематически изменяется в процессе работы ЭВМ в ОЗУ, СОЗУ, ВЗУ, редко изменяется в процессе эксплуатации в ППЗУ и никогда не изменяется в ПЗУ.

СОЗУ обладает максимальным быстродействием (равным процессорному), небольшим объемом (10^5 — 10^7 байтов) и располагается, как правило, на кристалле процессорной БИС. Для обращения к СОЗУ не требуются магистральные (машинные) циклы. В СОЗУ размещаются наиболее часто используемые на данном участке программы данные, а иногда — и фрагменты программы.

Быстродействие ОЗУ может быть ниже процессорного (не более чем на порядок), а объем составляет 10^5 — 10^7 байтов. В ОЗУ располагаются подлежащие выполнению программы и обрабатываемые данные. Связь между процессором и ОЗУ осуществляется по системному или специализированному интерфейсу и требует для своего осуществления машинных циклов.

К постоянной памяти относят постоянное запоминающее устройство, ПЗУ (в англоязычной литературе - Read Only Memory, ROM, что дословно переводится как "память только для чтения"),

перепрограммируемое ПЗУ, ППЗУ (в англоязычной литературе — Programmable Read Only Memory, PROM), и флэш-память (flash memory). Название ПЗУ говорит само за себя. Информация в ПЗУ записывается на заводе-изготовителе микросхем памяти, и в дальнейшем изменить ее значение нельзя. В ПЗУ хранится критически важная для компьютера информация, которая не зависит от выбора операционной системы. Программируемое ПЗУ отличается от обычного тем, что информация на этой микросхеме может стираться специальными методами (например, лучами ультрафиолета), после чего пользователь может повторно записать на нее информацию. Эту информацию будет невозможно удалить до следующей операции стирания информации.

Микросхемы программируемых ПЗУ по принципу построения и функционирования аналогичны масочным ПЗУ, но имеют существенное отличие в том, что допускают программирование на месте своего применения пользователем. Операция программирования заключается в разрушении (пережигании) части плавких перемычек на поверхности кристалла импульсами тока, амплитудой 30 — 50 мА. Технические средства для выполнения этой операции достаточно просты и могут быть построены самим пользователем. Это обстоятельство в сочетании с низкой стоимостью и доступностью микросхем ППЗУ обусловило их широкое распространение в радиолюбительской практике.

6.3. Виды памяти: статическая и динамическая

Память динамического типа

Экономичный вид памяти. Для хранения разряда (бита или трита) используется схема, состоящая из одного конденсатора и одного транзистора (в некоторых вариантах два конденсатора). Такой вид памяти, во-первых, дешевле (один конденсатор и один транзистор на 1 бит дешевле нескольких транзисторов триггера), и, во-вторых, занимает меньшую площадь на кристалле (там, где в SRAM размещается один триггер, хранящий 1 бит, можно разместить несколько конденсаторов и транзисторов для хранения нескольких бит). Но DRAM имеет и недостатки. Во-первых, работает медленнее, поскольку если в SRAM изменение управляющего напряжения на входе триггера сразу очень быстро изменяет его состояние, то для того, чтобы изменить состояние конденсатора его нужно зарядить или разрядить. Перезаряд конденсатора гораздо более длителен (в 10 и более раз), чем переключение триггера, даже если ёмкость

конденсатора очень мала. Второй существенный недостаток — конденсаторы со временем разряжаются. Причём разряжаются они тем быстрее, чем меньше их электрическая ёмкость и больше ток утечки, в основном, утечка через ключ.

Именно из-за того, что заряд конденсатора динамически уменьшается во времени, память на конденсаторах получила своё название DRAM — динамическая память. Поэтому, дабы не потерять содержимое памяти, заряд конденсаторов периодически восстанавливается («регенерируется») через определённое время, называемое циклом регенерации (обычно 2 мс). Для регенерации в современных микросхемах достаточно выполнить циклограмму «чтения» по всем строкам запоминающей матрицы. Процедуру регенерации выполняет процессор или контроллер памяти. Так как для регенерации памяти периодически приостанавливается обращение к памяти, это снижает среднюю скорость обмена с этим видом ОЗУ.

Память статического типа

ОЗУ, которое не надо регенерировать (обычно схемотехнически выполненное в виде массива триггеров), называют статической памятью с произвольным доступом или просто статической памятью. Достоинство этого вида памяти — скорость. Поскольку триггеры являются соединением нескольких логических вентилей, а время задержки на вентиль очень мало, то и переключение состояния триггера происходит очень быстро. Данный вид памяти не лишён недостатков. Во-первых, группа транзисторов, входящих в состав триггера, обходится дороже, чем ячейка динамической памяти, даже если они изготавливаются групповым методом миллионами на одной кремниевой подложке. Кроме того, группа транзисторов занимает гораздо больше площади на кристалле, чем ячейка динамической памяти, поскольку триггер состоит минимум из 2 вентилей (шести-восьми транзисторов), а ячейка динамической памяти — только из одного транзистора и одного конденсатора. Используется для организации сверхбыстродействующего ОЗУ, обмен информацией с которым критичен для производительности системы.

6.4. DRAM

DRAM (Dynamic random access memory, Динамическая память с произвольным доступом) — тип энергозависимой полупроводниковой памяти с произвольным доступом; DRAM широко используемая в качестве оперативной памяти современных компьютеров, а также в

качестве постоянного хранилища информации в системах, требовательных к задержкам.

Физически DRAM состоит из ячеек, созданных в полупроводниковом материале, в каждой из которых можно хранить определённый объём данных, строку от 1 до 4 бит. Совокупность ячеек такой памяти образуют условный «прямоугольник», состоящий из определённого количества строк и столбцов. Один такой «прямоугольник» называется страницей, а совокупность страниц называется банком. Весь набор ячеек условно делится на несколько областей.

Как запоминающее устройство, DRAM представляет собой модуль памяти различных конструктивов, состоящий из электрической платы, на которой расположены микросхемы памяти и разъёма, необходимого для подключения модуля к материнской плате.

Физически DRAM-память представляет собой набор запоминающих ячеек, которые состоят из конденсаторов и транзисторов, расположенных внутри полупроводниковых микросхем памяти.

Принцип действия

При отсутствии подачи электроэнергии к памяти этого типа происходит разряд конденсаторов, и память опустошается (обнуляется). Для поддержания необходимого напряжения на обкладках конденсаторов ячеек и сохранения их содержимого, их необходимо периодически подзаряжать, прилагая к ним напряжения через коммутирующие транзисторные ключи. Такое динамическое поддержание заряда конденсатора является основополагающим принципом работы памяти типа DRAM. Конденсаторы заряжают в случае, когда в «ячейку» записывается единичный бит, и разряжают в случае, когда в «ячейку» необходимо записать нулевой бит.

Важным элементом памяти этого типа является чувствительный усилитель-компаратор (англ. sense amp), подключенный к каждому из столбцов «прямоугольника». Он, реагируя на слабый поток электронов, устремившихся через открытые транзисторы с обкладок конденсаторов, считывает всю строку целиком. Именно строка является минимальной порцией обмена с динамической памятью, поэтому обмен данными с отдельно взятой ячейкой невозможен.

Регенерация памяти

В отличие от быстрой, но дорогой статической памяти типа SRAM (англ. static random access memory), которая является

конструктивно более сложным и более дорогим типом памяти и используется в основном в кэш-памяти, медленная, но дешёвая память DRAM(динамическая память с произвольным доступом) изготавливается на основе конденсаторов небольшой ёмкости, которые быстро теряют заряд, поэтому информацию приходится обновлять через определённые промежутки времени во избежание потерь данных. Этот процесс называется регенерацией памяти. Он реализуется специальным контроллером, установленным на материнской плате или же на кристалле центрального процессора. На протяжении времени, называемого шагом регенерации, в DRAM перезаписывается целая строка ячеек, и через 8-64 мс обновляются все строки памяти.

Процесс регенерации памяти в классическом варианте существенно тормозит работу системы, поскольку в это время обмен данными с памятью невозможен. Регенерация, основанная на обычном переборе строк, в современных типах DRAM не применяется. Существует несколько более экономичных вариантов этого процесса — расширенный, пакетный, распределённый; наиболее экономичной является скрытая (теневая) регенерация.

Среди новых технологий регенерации — PASR (англ. Partial Array Self Refresh), применяемый компанией Samsung в чипах памяти SDRAM(синхронная динамическая память с произвольным доступом) с низким уровнем энергопотребления. Регенерация ячеек выполняется только в период ожидания в тех банках памяти, в которых имеются данные.

Параллельно с этой технологией реализуется метод TCSR (англ. Temperature Compensated Self Refresh), который предназначен для регулировки скорости процесса регенерации в зависимости от рабочей температуры.

6.5. Структура организации блока памяти (2D, 3D, 2.5D)

Адресные ЗУ наиболее широко используются в современных ЭВМ для построения самых разнообразных устройств памяти. В процессе эволюции ЭВМ принципы построения и аппаратная реализация данных ЗУ прошли очень большой путь развития от первых ЗУ на электромагнитных реле до современных БИСов памяти емкостью в сотни Мбайт, которые в качестве ЗЭ используют либо разнообразные триггерные схемы на биполярных полупроводниках, либо МОП-структуры. При этом тип используемых ЗЭ влияет на структуру ЗУ. Кроме того, структура ЗУ во многом определяется особенностями его применения в конкретных

устройствах ЭВМ. Все это привело к тому, что в процессе развития возникло весьма большое разнообразие структур ЗУ, которые различаются по способу организации, быстродействию, объему, аппаратурным затратам, стоимости.

Ранее отмечалось, что основной частью любой памяти является запоминающий массив (ЗМ), представляющий собой совокупность ЗЭ, соединенных определенным образом. ЗМ называют еще запоминающей матрицей. Каждый ЗЭ хранит бит информации и должен реализовывать следующие режимы работы:

- хранение состояния (0 или 1);
- выдачу сигнала состояния (считывание);
- запись информации (0 или 1).

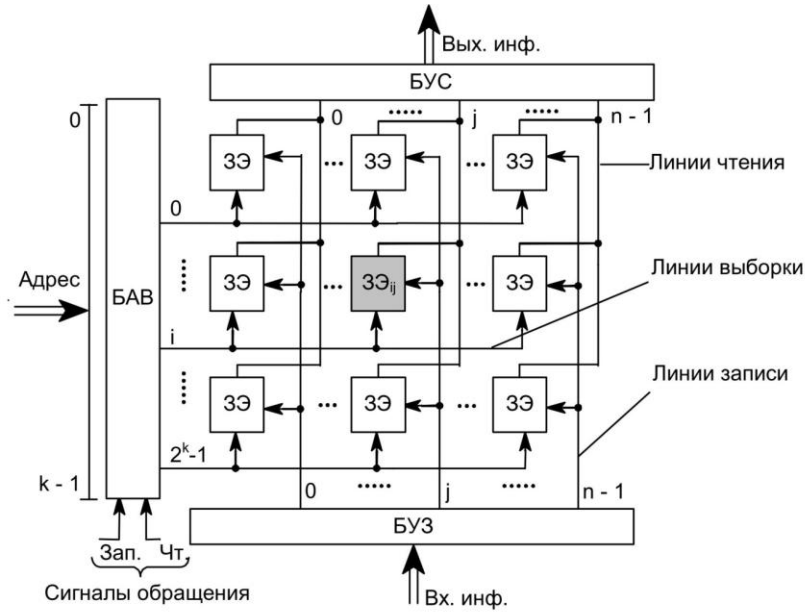
К ЗЭ должны поступать управляющие сигналы для задания режима работы, а также сигналы при записи. При считывании ЗЭ должен выдавать сигнал о своем состоянии, поэтому любой ЗМ имеет систему *адресных* и *разрядных* линий (проводников).

Адресные линии используются для выделения по адресу совокупности ЗЭ, которым устанавливается режим считывания или записи. Число ЗЭ, входящих в эту совокупность, равно ширине выборки. Иными словами, с помощью адресных линий происходит выбор необходимой ячейки памяти. Разрядные линии используются для записи или считывания информации в ЗЭ каждого разряда ячейки памяти.

Адресные и разрядные линии носят общее название *линий выборки*. В зависимости от числа таких линий, соединенных с одним ЗЭ, различают двух-, трехкоординатные ЗУ и т.д., называемые ЗУ типа 2D, 3D, 2.5D, 2D-M (от слова dimension — размерность), и их разнообразные модификации.

ЗУ типа 2D

Организация ЗУ типа 2D обеспечивает двухкоординатную выборку каждого ЗЭ ячейки памяти. Основу ЗУ составляет плоская матрица из ЗЭ, сгруппированных в 2^k ячеек по n разрядов. Обращение к ячейке задается k-разрядным адресом, что дает одну координату. Выделение разрядов производится разрядными линиями записи считывания, что дает вторую координату. Очень упрощенная структура ЗУ типа 2D представлена на рис. 6.1.



БАВ - блок адресной выборки (адресный формирователь);
БУС – блок усилителей считывания;
БУЗ – блок усилителя записи;
ЗЭ – запоминающий элемент

Рис. 6.1. Структура ЗУ типа 2D

Адрес из k разрядов поступает на блок адресной выборки БАВ (который называют также адресным формирователем), управляемый сигналами Чт. и Зап. Основу БАВ составляет дешифратор с 2k выходами, который при поступлении на его вход адреса формирует сигнал для выбора линии i. В зависимости от сигнала Чт. или Зап. БАВ в общем случае выдает сигнал, настраивающий ЗЭ i-й ячейки (i-й линии) на чтение либо на запись. Выделение разряда j в i-м слове (ЗУ серого цвета) производится второй координатной линией. При записи по линии j от БУЗ поступает сигнал, устанавливающий выбранный для записи ЗЭi,j в состояние 0 или 1. При считывании на БУС по линии j поступает сигнал о состоянии ЗЭi,j.

Следует иметь в виду, что ЗЭ должны допускать объединение выходов для работы на общую линию с передачей сигналов только от выбранного ЗЭ. Это свойство ЗЭ используется во всех современных ЗУ.

Таким образом, каждая адресная линия выборки ячейки памяти в общем случае передает три сигнала:

- Выборка при записи;
- Выборка при считывании;
- Отсутствие выборки.

Однако во многих современных ЗУ достаточно только двух сигналов — выборка и отсутствие выборки.

Каждая разрядная линия записи передает в ЗЭ записываемый бит информации, а разрядная линия считывания — считываемый из ЗЭ бит информации. Линии записи и считывания могут быть объединены в одну при использовании ЗЭ, допускающих соединение выхода со входом записи. В современных ЗУ широко используются совмещенные функции линий считывания и записи.

ЗУ типа 2D являются быстродействующими и достаточно удобными для реализации. Однако такие ЗУ неэкономичны по объему оборудования из-за наличия дешифратора на 2k выходов. В настоящее время структуры типа 2D используются, в основном, в ЗУ небольшой емкости (не более 1 К).

ЗУ ТИПА 3D

Для построения ЗУ больших объемов используют другую схему и другие типы ЗЭ, которые имеют не один, а два конъюнктивно связанных входа выборки. В этом случае адресная выборка осуществляется только при одновременном появлении двух сигналов. Использование таких ЗЭ позволяет строить ЗУ с трехкоординатным выделением ЗЭ. Итак, ЗУ типа 3D отличается от 2D тем, что к каждому ЗЭ подходят три линии выборки: две координатные и одна разрядная.

Запоминающий массив ЗУ типа 3D представляет собой пространственную матрицу, составленную из n плоских матриц. Каждая плоская матрица представляет собой ЗМ для запоминания j-х разрядов всех слов, т.е. запоминающие элементы для одноименных разрядов всех хранимых в ЗУ чисел сгруппированы в квадратную матрицу из 2k/2 рядов по 2k/2 ЗЭ в каждом. Это означает, что к записи или считыванию готов только тот элемент, для которого сигналы адресной выборки по координатам X и Y совпали. Для адресной выборки ЗЭ в плоской матрице необходимо задать две его координаты в ЗМ. ЗМ.

Структура матрицы j-го разряда в ЗУ типа 3D представлена на рис. 6.2. Код адреса i-й ячейки памяти разделяется на старшую и младшую части, каждая из которых поступает на свой адресный формирователь. Адресный формирователь БАВ1 выдает сигнал

выборки на линию i', а БАВ2 — на линию i". В результате в ЗМ оказывается выбранным ЗЭ, находящийся на пересечении этих линий (двух координат), т.е. адресуемый кодом i=i' / i" (ЗЭ серого цвета). Адресные формирователи управляются сигналами Чт и Зап и в зависимости от них выдают сигналы выборки для считывания или для записи. При считывании сигнал о состоянии выбранного ЗЭ поступает по j-линии считывания к БУС (третья координата ЗЭ при считывании). При записи в выбранный ЗЭ будет занесен 0 или 1 в зависимости от сигнала записи в j-й разряд, поступающего по j-й линии от БУЗ (третья координата ЗЭ при записи). Для полупроводниковых ЗУ характерно объединение в одну линию разрядных линий записи и считывания.

Для построения n-разрядной памяти используется n матриц рассмотренного вида. Адресные формирователи здесь могут быть общими для всех разрядных ЗМ.

Запоминающие устройства типа 3D более экономичны по оборудованию, чем ЗУ типа 2D. Действительно, сложность адресного формирователя с m входами пропорциональна $2m$, отсюда сложность двух адресных формирователей ЗУ типа 3D, пропорциональная $2 \cdot 2k$, значительно меньше сложности адресного формирователя ЗУ типа 2D, пропорциональной $2k$. Поэтому структура типа 3D позволяет строить ЗУ большего объема, чем структура 2D.

Запоминающие устройства типа 2.5D.

В ЗУ этого типа при считывании состояния j-го разряда i-й ячейки положение $3Э_{ij}$ в ЗМ определяется тремя координатами (две координаты для выборки и одна для выходного сигнала), а при записи в $3Э_{ij}$ - двумя координатами. Считывание при этом осуществляется так же, как и в ЗУ типа 3D, а запись сходна с записью в ЗУ типа 2D.

Запоминающий массив ЗУ типа 2,5D можно рассматривать как состоящий из отдельных ЗМ для каждого разряда памяти: $ЗМ_0$, $ЗМ_1$..., $ЗМ_j$,..., $ЗМ_{n-1}$. Структура одноразрядного ЗУ дана на рис. 6.3.

Код адреса i-й ячейки памяти, как и в ЗУ 3D, разделяется на две части: i' и i", каждая из которых отдельно дешифрируется. Адресный формирователь АдрФ выдает сигнал выборки на линию i', разрядно-адресный формирователь j-го разряда РАдрФ - на линию i". При считывании оба сигнала, являющиеся сигналами выборки для считывания, опрашивают ЗЭ, выходной сигнал которого поступает на УсСч разряда j. Работает ЗУ в этом случае так же, как и ЗУ типа 3D.

При записи АдрФ выдает сигнал выборки для записи, а РАдрФ выдает по линии i" сигнал записи 0 или 1 в зависимости от назначения входного информационного сигнала j-го разряда $ВхИнФ_j$.

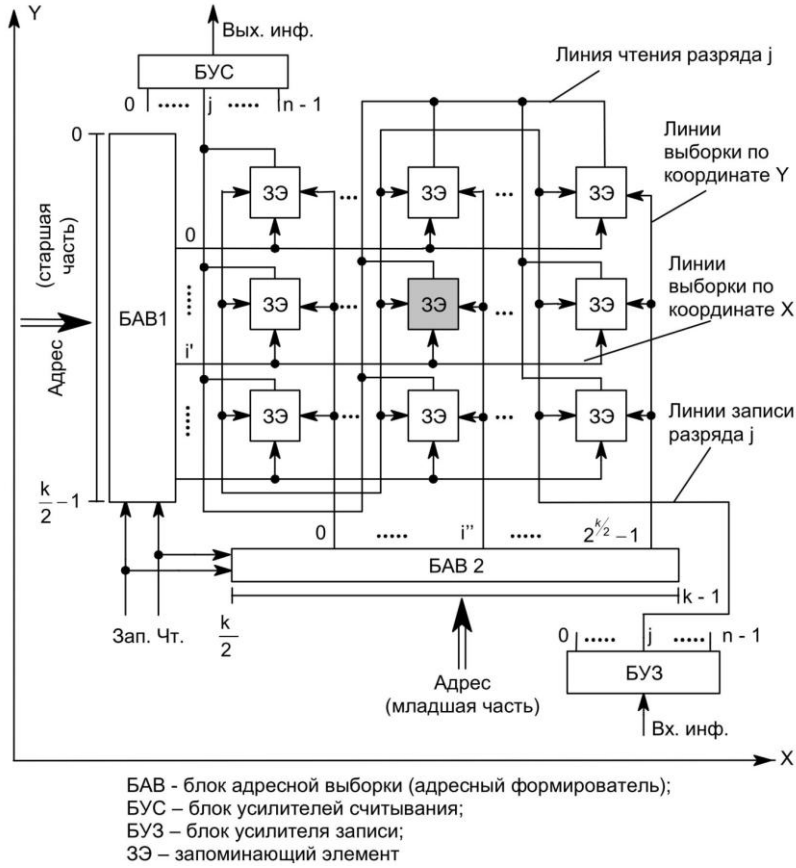


Рис. 6.2. Структура ЗУ типа 3D для j-го разряда

На остальных линиях РадрФ$_i$ не появляются сигналы записи, и состояния всех ЗЭ, кроме ЗЭ, лежащего на пересечении линий i' и i'', не меняются.

Из ЗМ отдельных разрядов формируется ЗМ всего ЗУ согласно схеме на рис. 4.7, б.

Наиболее экономичным по расходу оборудования ЗУ оказывается в том случае, если число выходных линий АдрФ и всех РАдрФ равно, т.е. если

$r = (k - r) \log_2 n$. (рис. 6.3, б).

Недостатком ЗУ типа 2,5 D является то, что сигналы на линиях РАдрФ должны иметь четыре значения: чтение, запись О, запись 1 и отсутствие записи (хранение). Для ЗЭ с разрушающим считыванием сигналы чтения и записи 0 совпадают и потребуются лишь три значения сигнала. В связи с этим ЗУ типа 2,5D используется для ЗЭ с разрушающим считыванием.

Для построения современных полупроводниковых ЗУ из ЗЭ с неразрушающим считыванием используется структура ЗУ с двухкоординатным выделением ЗЭ и мультиплексированием выходных сигналов при считывании.

6.6. Регенерация памяти. Различные методы регенерации (ROR, CBR, SR). SRAM

Запоминающие ячейки микросхем DRAM организованы в виде двумерной матрицы. Адрес строки и столбца передаётся по мультиплексированной шине адреса MA (Multiplexed Address) и стробируются по спаду импульсов RAS# (Row Access Strobe) и CAS# (Column Access Strobe). Временная диаграмма "классических" циклов записи и чтения приведена на рис. 6.3.

Поскольку обращения (запись и чтение) к различным ячейкам памяти обычно происходят в случайном порядке, то для поддержания сохранности данных применяется *регенерация* (Memory Refresh - "освежение памяти") - регулярный циклический перебор её ячеек (обращение к ним) с холостыми циклами. Регенерация в микросхеме происходит одновременно по всей строке матрицы при обращении к любой из её ячеек. Максимальный период обращения к каждой строке T_{RF} (refresh time) для гарантированного сохранения информации у современной памяти лежит в пределах 8-64 мс. В зависимости от объёма и организации матрицы для однократной регенерации всего объёма требуется 512, 1024, 2048 или 4096 циклов обращений. При *распределённой регенерации* (distributed refresh) одиночные циклы регенерации выполняются равномерно с периодом t_{RF} (рис. 6.4,а), который для стандартной памяти принимается равным 15.6 мкс. Период этих циклов называют "refresh rate", хотя такое название подошло бы больше к обратной величине - частоте циклов $f=1/ t_{RF}$. Для памяти с расширенной регенерацией (extended refresh) допустим период циклов до 125 мкс. Возможен также пакет *пакетной регенерации* (burst refresh), когда все циклы регенерации собираются в пакет (рис. 6.4,б), во время которого обращение к памяти по чтению и

записи блокируется. При количестве циклов 1024 эти пакеты будут периодически занимать шину памяти примерно на 130 мкс, что далеко не всегда допустимо. По этой причине практически всегда выполняется распределённая регенерация, хотя возможен и промежуточный вариант - пакетами по несколько (например, 4) циклов.

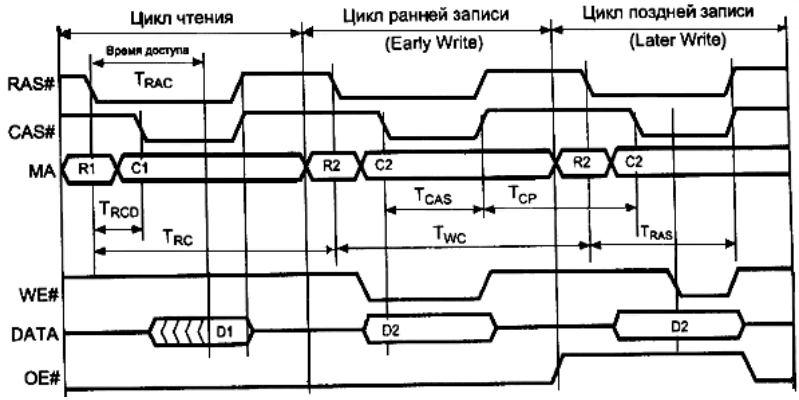


Рис.6.3. Временные диаграммы чтения и записи динамической памяти.

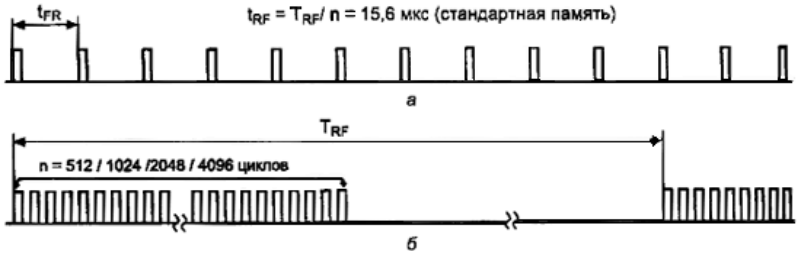


Рис.6.4. Методы регенерации динамической памяти

Циклы регенерации могут организовываться разными способами. Классическим является цикл без импульса CAS# (рис. 6.5а), сокращённо именуемый ROR - RAS Only refresh - регенерация только импульсом RAS#) . В этом случае адрес очередной регенерируемой строки выставляется контроллером памяти до спада RAS# очередного цикла регенерации, порядок перебора регенерируемых строк не важен.

Другой вариант - цикл CBR (CAS Before RAS), поддерживаемый практически всеми современными микросхемами памяти (рис. 6.5,б). В этом цикле регенерации спад импульса RAS#осуществляется при

низком уровне сигнала CAS# (в обычном цикле обращения такой ситуации не возникает). В этом случае микросхема выполняет регенерацию строки, адрес которой находится во внутреннем счётчике микросхемы, и в задачу контроллера входит только периодическое формирование таких циклов. Во время спада RAS# сигнал WE# должен находится в состоянии высокого уровня. Дополнительным преимуществом данного цикла является экономия потребляемой мощности за счёт неактивности внутренних адресных буферов.

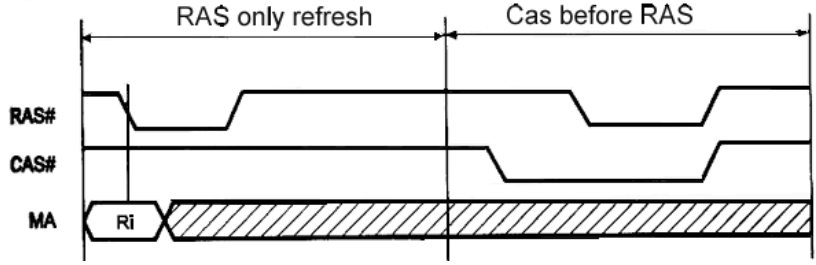


Рис. 6.5. Циклы регенерации динамической памяти

Микросхемы синхронной динамической памяти выполняют циклы CBR по команде Auto Refresh. А по команде Self Refresh или Sleep Mode они выполняют автономную регенерацию в энергосберегающем режиме. Такой возможностью обладают некоторые современные микросхемы, имеющие внутренний генератор. Вход в режим осуществляется как в цикл CBR, но сигналRAS# должен быть активен более 100 мкс. Информация в таком состоянии будет храниться сколь угодно долго при наличии питающего напряжения. Выход из этого "спящего" состояния осуществляется по подъёму сигналов RAS# и CAS#.

Цикл скрытой регенерации (hidden refresh) является разновидностью цикла CBR: здесь в конце полезного цикла чтения при записи сигнал CAS# удерживается на низком уровне, а RAS#поднимается и снова опускается, что и является указанием микросхеме на выполнение цикла регенерации по внутреннему счётчику (рис. 6.6). При этом слово "скрытность" не всегда означает экономию времени (затраты на регенерацию остаются теми же, что и в обычном CBR, хотя в принципе возможно предельное укорочение активной части импульса CAS# при чтении). Во время скрытой регенерации после цикла чтения выходные буферы сохраняют только что считанные данные (в обычном CBR выходные буферы находятся в высокоимпендансном состоянии).

Регенерация основной памяти в PC/XT осуществлялась каналом DMA-0. Сигнал Refr, вырабатываемый каждые 15,6 мкс по сигналу от первого канала таймер-счётчика 8253/9254 (порт 041h), вызывает холостой цикл обращения к памяти для регенерации очередной строки. В PC/XT контроллер регенерации усложнён. В современных компьютерах регенерацию основной памяти берёт на себя чипсет, и его задача - по возможности использовать для регенерации циклы шины, не занятые её абонентами (процессорами и активными контроллерами). Самые "ловкие " контроллеры регенерации (smart refresh) ставят запросы на регенерацию в очередь, которую в свободное для шины время, и только если запросов накапливается больше предельного количества, откладывается текущий цикл обмена по шине и цикл регенерации выполняется немедленно. Модули памяти в разных банках могут регенерироваться одновременно, но при использовании чередования (interleaving) для экономии времени целесообразно производить регенерацию одного банка во время полезного обращения к другому. Некоторые системные платы позволяют использовать режим пониженной частоты регенерации (slow refresh), однако его можно применять только с модулями памяти, допускающими режим Extended Refresh.

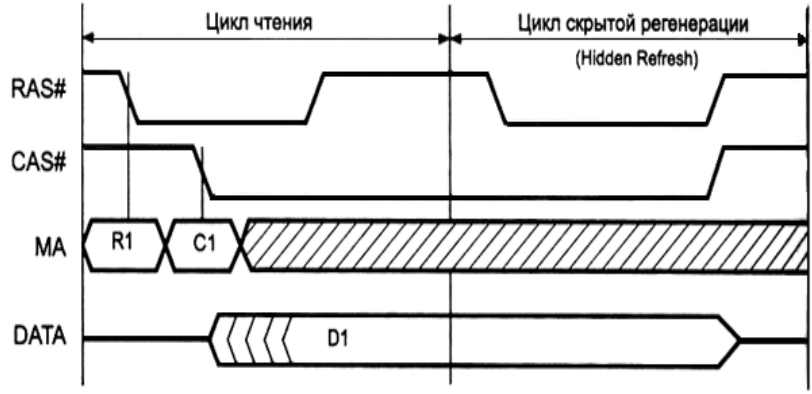


Рис.6.6. Скрытая регенерация (Hidden Refresh)

Динамическая память, используемая в видеобуферах графических адаптеров, специальных циклов регенерации, как правило, не требует, поскольку частота её чтения для регенерации изображения вполне достаточна для сохранения информации.

6.7. SRAM

Статическая оперативная память с произвольным доступом (SRAM, static random access memory) — полупроводниковая оперативная память, в которой каждый двоичный или троичный разряд хранится в схеме с положительной обратной связью, позволяющей поддерживать состояние без регенерации, необходимой в динамической памяти (DRAM). Тем не менее, сохранять данные без перезаписи SRAM может только пока есть питание, то есть SRAM остается энергозависимым типом памяти. Произвольный доступ (RAM — random access memory) — возможность выбирать для записи/чтения любой из битов (тритов) (чаще байтов (трайтов), зависит от особенностей конструкции), в отличие от памяти с последовательным доступом (SAM — sequential access memory).

Преимущества:

- Быстрый доступ. SRAM — это действительно память произвольного доступа, доступ к любой ячейке памяти в любой момент занимает одно и то же время.
- Простая схемотехника — SRAM не требуются сложные контроллеры.
- Возможны очень низкие частоты синхронизации, вплоть до полной остановки синхроимпульсов.

Недостатки:

- невысокая плотность записи (шесть-восемь элементов на бит, вместо двух у DRAM);
- дороговизна килобайта памяти.

Тем не менее, высокое энергопотребление не является принципиальной особенностью SRAM, оно обусловлено высокими скоростями обмена с данным видом внутренней памяти процессора. Энергия потребляется только в момент изменения информации в ячейке SRAM.

6.8. Классификация (ROM, PROM, EPROM, EEPROM, FLASH MEMORY)

В зависимости от возможности записи и перезаписи данных в памяти последняя подразделяется на:

- память (ЗУ) с записью-считыванием [read/write memory] - Тип памяти, дающей возможность пользователю помимо считывания данных производить их исходную запись, стирание и/или

обновление. К этому виду м.б. отнесены оперативная память (см. - ОЗУ , RAM, кэш-память) а также ППЗУ ;

- постоянная память , постоянное ЗУ , ПЗУ [Read Only Mamory (ROM)] - Тип памяти (ЗУ), предназначенный для хранения и считывания данных, которые никогда не изменяются. Запись данных на ПЗУ производится в процессе его изготовления, поэтому пользователем изменена быть не может. Наиболее распространены ПЗУ, выполненные на интегральных микросхемах (БИС , СБИС) и оптических (компакт) дисках (см. CD-ROM);
- программируемая постоянная память, программируемое ПЗУ, ППЗУ [PROM - Programmable Read-Only Memory] - Постоянная память или ПЗУ, в которых возможна запись или смена данных путем воздействия на носитель информации электрическими, магнитными и/или электромагнитными (в т.ч. ультрафиолетовыми или другими) полями под управлением специальной программы. Различают ППЗУ с однократной записью и стираемые ППЗУ [EPROM - Erasable PROM], в том числе:
  - электрически программируемое ПЗУ (ЭППЗУ) [EAROM - Electrically Alterable Read Only Memory];
  - электрически стираемое программируемое ПЗУ (ЭСПЗУ) [EEPROM - Electrically Erasable Programmable Read-Only Memory]. К стираемым ППЗУ относятся микросхемы флэш-памяти отличающиеся высокой скоростью доступа и возможностью быстрого стирания данных.
  - флеш-память (англ. flash memory) — разновидность полупроводниковой технологии электрически перепрограммируемой памяти (EEPROM). Это же слово используется в электронной схемотехнике для обозначения технологически законченных решений постоянных запоминающих устройств в виде микросхем на базе этой полупроводниковой технологии. В быту это словосочетание закрепилось за широким классом твердотельных устройств хранения информации. Благодаря компактности, дешевизне, механической прочности, большому объёму, скорости работы и низкому энергопотреблению, флеш-память широко используется в цифровых портативных устройствах и носителях информации. Серьёзным недостатком данной технологии является ограниченный срок эксплуатации

носителей, а также чувствительность к электростатическому разряду.

6.9. Аппаратный контроль корректности работы памяти. Контроль четности. ECC

Ошибки при хранении информации в оперативной памяти неизбежны. Они обычно классифицируются как аппаратные отказы и нерегулярные ошибки (сбои).

Если нормально функционирующая микросхема вследствие, например, физического повреждения начинает работать неправильно, то это называется аппаратным отказом. Чтобы устранить данный тип отказа, обычно требуется заменить некоторую часть аппаратных средств памяти, например неисправную микросхему, односторонний модуль памяти или двухсторонний модуль памяти.

Другой, более коварный тип отказа — нерегулярная ошибка (сбой). Это непостоянный отказ, который не происходит при повторении условий функционирования или через регулярные интервалы. (Такие отказы обычно “лечатся” выключением питания компьютера и последующим его включением.)

Приблизительно 20 лет назад сотрудники Intel установили, что причиной сбоев являются альфа-частицы. Поскольку альфа-частицы не могут проникнуть даже через тонкий лист бумаги, выяснилось, что их источником служит вещество, используемое в полупроводниках. При исследовании были обнаружены частицы тория и урана в пластмассовых и керамических корпусах микросхем, применявшихся в те годы. Изменив технологический процесс, производители памяти избавились от этих примесей.

В настоящее время производители памяти почти полностью устранили источники альфа-частиц. В связи с этим многие производители модулей памяти исключили из своей продукции поддержку проверки четности, несмотря на то, что от сбоев памяти не удалось избавиться полностью. Более поздние исследования показали, что альфа-частицы охватывают лишь малую долю причин сбоев памяти.

Сегодня самая главная причина нерегулярных ошибок — космические лучи. Поскольку они имеют очень большую проникающую способность, от них практически невозможно защититься с помощью экранирования. Этот тезис был подтвержден рядом исследований, проведенных компанией IBM под руководством доктора Дж.Ф. Зиглера.

Для повышения отказоустойчивости в современных компьютерах применяются такие методы как контроль четности и коды коррекции ошибок (ECC).

Системы без контроля четности вообще не обеспечивают отказоустойчивость. Единственная причина, по которой они используются, — их минимальная базовая стоимость. При этом, в отличие от других технологий, не требуется дополнительная оперативная память. Байт данных с контролем четности включает в себя 9, а не 8 бит, поэтому стоимость памяти с контролем четности выше примерно на 12,5%. Кроме того, контроллеры памяти, не требующие логических мостов для подсчета данных четности или ECC, обладают упрощенной внутренней архитектурой. Портативные системы, для которых вопрос минимального энергопотребления особенно важен, выигрывают от уменьшенного энергоснабжения памяти благодаря использованию меньшего количества микросхем DRAM. И наконец, шина данных памяти без контроля четности имеет меньшую разрядность, что выражается в сокращении количества буферов данных. Статистическая вероятность возникновения ошибок памяти в современных настольных компьютерах составляет примерно одну ошибку в несколько месяцев. При этом количество ошибок зависит от объема и типа используемой памяти.

Подобный уровень ошибок может оказаться приемлемым для обычных компьютеров, не используемых для работы с важными приложениями. В этом случае цена играет основную роль, а дополнительная стоимость модулей памяти с поддержкой контроля четности и кода ECC себя не оправдывает, поэтому легче смириться с нечастыми ошибками.

Контроль четности

Это один из введенных IBM стандартов, в соответствии с которым информация в банках памяти хранится фрагментами по 9 бит, причем восемь из них (составляющих один байт) предназначены собственно для данных, а девятый является битом четности. Использование девятого бита позволяет схемам управления памятью на аппаратном уровне контролировать целостность каждого байта данных. Если обнаруживается ошибка, работа компьютера останавливается, а на экран выводится сообщение о неисправности. Если вы работаете на компьютере под управлением Windows или OS/2, то при возникновении ошибки контроля четности сообщение, возможно, не появится, а просто произойдет блокировка системы.

После перезагрузки система BIOS должна идентифицировать ошибку и выдать соответствующее сообщение.

Первые ПК использовали память с контролем четности для регулировки точности осуществляемых операций. Начиная с 1994 года на рынке ПК стала развиваться тревожная тенденция. Большинство компаний начали предлагать компьютеры с памятью без контроля четности и вообще без каких бы то ни было средств определения или исправления ошибок. Применение модулей SIMM без контроля четности сокращало стоимость памяти на 10—15%. В свою очередь, память с контролем четности обходилась дороже за счет применения дополнительных битов четности. Технология контроля четности не позволяет исправлять системные ошибки, однако предоставляет пользователю компьютера возможность их обнаружить, что имеет следующие преимущества:

- контроль четности защищает от последствий неверных вычислений на базе некорректных данных
- контроль четности точно указывает на источник возникновения ошибок, помогая разобраться в проблеме и улучшая степень эксплуатационной надежности компьютера.

Для реализации поддержки памяти с контролем четности или без него не требуется особых усилий. В частности, внедрить поддержку контроля четности для системной платы не составит труда. Основные затраты внедрения связаны со стоимостью самих модулей памяти с контролем четности. Если покупатели готовы пойти на дополнительные затраты для повышения надежности заказываемых систем, производители компьютеров могут предоставить им такую возможность.

Далее рассматривается принцип выполнения контроля четности и ECC, который позволяет не только обнаруживать, но и автоматически корректировать ошибки памяти.

Схема проверки четности

При разработке стандарта контроля четности IBM определила, что значение бита четности задается таким, чтобы количество единиц во всех девяти разрядах (восемь разрядов данных и разряд четности) было нечетным. Другими словами, когда байт (8 бит) данных заносится в память, специальная схема контроля четности (микросхема, установленная на системной плате или на плате памяти) подсчитывает количество единиц в байте. Если оно четное, на выходе микросхемы формируется сигнал логической единицы, который сохраняется в соответствующем разряде памяти как девятый бит (бит

четности). Общее количество единиц во всех девяти разрядах при этом становится нечетным. Если же количество единиц в восьми разрядах исходных данных нечетное, то бит четности равен нулю, и сумма двоичных цифр в девяти разрядах также остается нечетной.

Рассмотрим конкретный пример (имейте в виду, что разряды в байте нумеруются с нуля, т.е. 0, 1, 2, ..., 7).

Разряд данных: 0 1 2 3 4 5 6 7 бит чётности

Значение бита: 1 0 1 1 0 0 1 1 0

В данном случае общее число единичных битов данных нечетное (5), поэтому бит четности должен быть равен нулю.

Рассмотрим еще один пример.

Разряд данных: 0 1 2 3 4 5 6 7 бит чётности

Значение бита: 0 0 1 1 0 0 1 1 1

В этом примере общее число единичных битов данных четное (4), поэтому бит четности должен быть равен единице, чтобы количество единиц во всех девяти разрядах было нечетным.

При считывании из памяти та же самая микросхема проверяет информацию на четность. Если в 9-разрядном байте число единиц четное, значит, при считывании или записи данных произошла ошибка. Определить, в каком разряде она произошла, невозможно (нельзя даже выяснить количество испорченных разрядов). Более того, если сбой произошел в трех разрядах (в нечетном их количестве), то ошибка будет зафиксирована; однако при двух ошибочных разрядах (или четном их количестве) сбой не регистрируется. Поскольку одновременная ошибка в нескольких разрядах одного байта крайне маловероятна, такая схема проверки была недорогой и при этом позволяла с большой вероятностью определять ошибки в памяти.

При обнаружении ошибки схема контроля четности на системной плате формирует немаскируемое прерывание (NMI) - системное предупреждение, которое программы не могут проигнорировать. Основная работа прекращается и инициируется специальная процедура, записанная в BIOS.

Еще несколько лет назад, когда память была дорогой, некоторые компании выпускали модули SIMM с фиктивными микросхемами проверки четности. Вместо того чтобы хранить биты четности для каждого байта памяти, эти микросхемы генерировали всегда корректный бит дополнения. Таким образом, когда система пыталась записать бит четности, он попросту отбрасывался, а при считывании байта всегда подставлялся “нужный” бит четности. В результате система всегда получала информацию о корректной работе памяти, хотя на самом деле все могло быть далеко не так.

Такие мошеннические действия были вызваны дороговизной микросхем памяти, и производители были готовы переплатить пару лишних долларов на генератор, чтобы не платить за более дорогую микросхему, хранящую биты четности. К сожалению, определить наличие в модуле памяти такого генератора было достаточно сложно. Поддельный генератор четности внешне отличался от обычных микросхем памяти и имел маркировку, отличную от других микросхем модуля. Большинство из генераторов имели логотип “GSM”, который указывал на изготовителя логического устройства проверки четности, часто отличавшегося от компании, выпустившей сам модуль памяти.

Единственным инструментом, позволявшим выявить модули с поддельным контролем четности, были аппаратные тестеры. Сейчас же цены на память упали, что устранило первопричину подобных махинаций.

Код коррекции ошибок

Коды коррекции ошибок (Error Correcting Code — ECC) позволяют не только обнаружить ошибку, но и исправить ее в одном разряде. Поэтому компьютер, в котором используются подобные коды, в случае ошибки в одном разряде может работать без прерывания, причем данные не будут искажены. Коды коррекции ошибок в большинстве ПК позволяют только обнаруживать, но не исправлять ошибки в двух разрядах. В то же время приблизительно 98% сбоев памяти вызвано именно ошибкой в одном разряде, т.е. она успешно исправляется с помощью данного типа кодов. Данный тип ECC получил название SEC/DED (эта аббревиатура расшифровывается как “одноразрядная коррекция, двухразрядное обнаружение ошибок”).

В кодах коррекции ошибок этого типа для каждых 32 бит требуется дополнительно семь контрольных разрядов при 4-байтовой и восемь — при 8-байтовой организации (64-разрядные процессоры Athlon/Pentium). Реализация кода коррекции ошибок при 4-байтовой организации, естественно, дороже обычной проверки четности, но при 8-байтовой организации их стоимости равны, поскольку требуют одного и того же количества дополнительных разрядов.

Для использования кодов коррекции ошибок необходим контроллер памяти, вычисляющий контрольные разряды при операции записи в память. При чтении из памяти такой контроллер сравнивает прочитанные и вычисленные значения контрольных разрядов и при необходимости исправляет испорченный бит (или биты). Стоимость дополнительных логических схем для реализации кода коррекции ошибок в контроллере памяти не очень высока, но это может

значительно снизить быстродействие памяти при операциях записи. Это происходит потому, что при операциях записи и чтения необходимо ожидать завершения вычисления контрольных разрядов. При записи части слова вначале следует прочитать полное слово, затем перезаписать изменяемые байты и только после этого — новые вычисленные контрольные разряды.

В большинстве случаев сбой памяти происходит в одном разряде, и потому такие ошибки успешно исправляются с помощью кода коррекции ошибок. Использование отказоустойчивой памяти обеспечивает высокую надежность компьютера. Память с кодом ECC предназначена для серверов, рабочих станций или приложений, в которых потенциальная стоимость ошибки вычислений значительно превышает дополнительные средства, вкладываемые в оборудование, а также временные затраты системы. Если данные имеют особое значение, и компьютеры применяются для решения важных задач, без памяти ECC не обойтись. По сути, ни один уважающий себя системный инженер не будет использовать сервер, даже самый неприхотливый, без памяти ECC.

Пользователи имеют выбор между системами без контроля четности, с контролем четности и с ECC, т.е. между желательным уровнем отказоустойчивости компьютера и степенью ценности используемых данных.

6.10. Логическая организация памяти

Аппаратный интерфейс процессора и памяти во всех компьютерах почти одинаков. Процессор выдает целевой физический адрес ячейки памяти на шину адреса, формирует сигнал считывания или записи, в операции записи помещает данные на шину данных, а память реагирует, возвращая считанные данные по шине данных или воспринимая новые данные с шины данных. Процессор, имеющий n линий в шине адреса, может обращаться к 2^n ячейкам памяти. Схема интерфейса действует линейным образом, т.е. каждая из 2^n возможных комбинаций соответствует только одной конкретной ячейке памяти. Если вы имеете 32 адресную шину, то возможный объем линейно адресуемой памяти равен 4 Гбайт, что вызывает ряд вопросов, т.к. реально объем оперативной памяти существенно меньше. Реализация потенциально имеющейся памяти — задача логической организации памяти. Есть еще одна непростая проблема — адресация и количество бит для ее представления. Рассмотрим команду ADD, которая требует

спецификации трех операндов: двух источников и одного пункта назначения.

Код операции	Адрес 1-го операнда	Адрес 2-го операнда	Адрес результата

Рис 6.7. Схема команды ADD

Если адреса памяти 32-битные, то нам понадобится помимо кода операции три 32-битных адреса, т.е. общее число разрядов, которое будет представлять эту команду и, следовательно, разрядность шины данных будет превышать 100 бит. Представьте себе это число — 2^{100} , а это ведь минимальные требования к объему адресного пространства памяти в этом случае. Значит необходимо найти способы уменьшения спецификации адресации.

Два метода уменьшения спецификации адресации очевидны.

Во-первых, если операнд используется неоднократно, а это наиболее распространенный случай, его можно переместить в регистр. Это даст не только уменьшение времени доступа, но и, главным образом, меньшее количество бит для определения операнда. Если имеется 32 регистра, любой из них можно определить, используя только 5 битов. Для нашего случая это потребует всего лишь 15 бит для определения всех трех операндов, а не 96.

Во-вторых, сокращение длины команды, что подразумевает определение одного или двух операндов неявным образом. Обычная трехадресная команда ADD использует следующую форму:

DESTINATION=SOURSE1+SOURSE2

Этот формат можно перевести в двухадресную команду и сократить до формы

REGISTER2=REGISTER2+SOURSE1

Этот прием сокращает число операндов с трех до двух, используя безадресный регистр-аккумулятор, мы можем прибавлять слово из памяти непосредственно к содержимому аккумулятора. Это уже одноадресные команды. Применение стековой организации памяти позволяет и вовсе обойтись безадресными командами. Все это имеет отношение к логической организации памяти.

Адресная, ассоциативная и стековая организация памяти

Запоминающее устройство, как правило, содержит множество одинаковых запоминающих элементов, образующих запоминающий массив. Массив разделен на отдельные ячейки; каждая из них

предназначена для хранения двоичного кода, число разрядов в котором определяется шириной выборки памяти (байт, машинное слово или несколько слов). Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве. По этому признаку различают адресную, ассоциативную и стековую память.

Адресная память

При адресной организации памяти размещение и поиск информации в запоминающем массиве основаны на использовании адреса хранения слова. Адресом служит номер ячейки массива, в которой это слово размещается. При записи (или считывании) слова в запоминающий массив, инициирующая эту операцию команда должна указывать адрес, по которому производится запись считывание).

Типичная структура адресной памяти содержит запоминающий массив из n-разрядных ячеек (обычно n равно 1, 4, 8 или 16) и его аппаратурное обрамление, включающее регистр адреса РгА, имеющий k разрядов (k больше или равно логарифму по основанию 2 от N), информационный регистр РгИ, блок адресной выборки БАВ, блок усилителей считывания БУС, блок разрядных усилителей-формирователей сигналов записи БУЗ и блок управления памятью БУП.

По коду адреса в регистре адреса блок адресной выборки формирует в соответствующей ячейке памяти сигналы, позволяющие произвести считывание или запись слова в ячейку. Проиллюстрируем адресную структуру памяти на примере ЭВМ с 16-разрядной шиной данных и 16-разрядными словами. Понятно, что при обмене данными с памятью по шине данных пересылаются двухбайтовые слова.

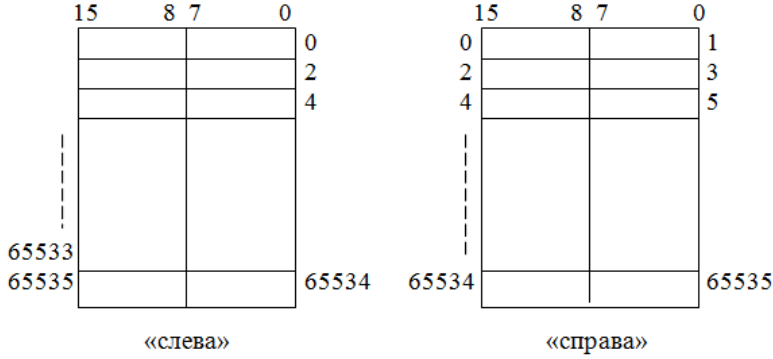


Рис. 6.8. Адресная память.

Такая структура организации памяти позволяет разрешить ряд проблем по хранению символов, закодированных в последовательных байтах (облегчается работа с текстами).

Ассоциативная память

В памяти этого типа поиск нужной информации производится не по адресу, а по ее содержанию (по ассоциативному признаку). При этом поиск по ассоциативному признаку (или последовательно по отдельным его разрядам) происходит параллельно во времени для всех ячеек запоминающего массива. Во многих случаях ассоциативный поиск позволяет существенно упростить и ускорить обработку данных. Это достигается за счет того, что в памяти этого типа операция считывания информации совмещена с выполнением ряда логических операций.

Память этого типа применяется в специализированных вычислительных машинах, машинах баз данных и при организации работы кэш-памяти. Если использовать адресную память или стек, то процедура поиска и вывода нужной информации сводится к последовательному считыванию содержимого отдельных ячеек памяти, выделению с помощью маски десяти его первых битов, сравнению выделенных битов с искомым кодом и выводом результата. Это очень нерационально. Ассоциативная память резко сокращает время поиска нужной информации. На рисунке 6.9 приведен пример организации подобной памяти.

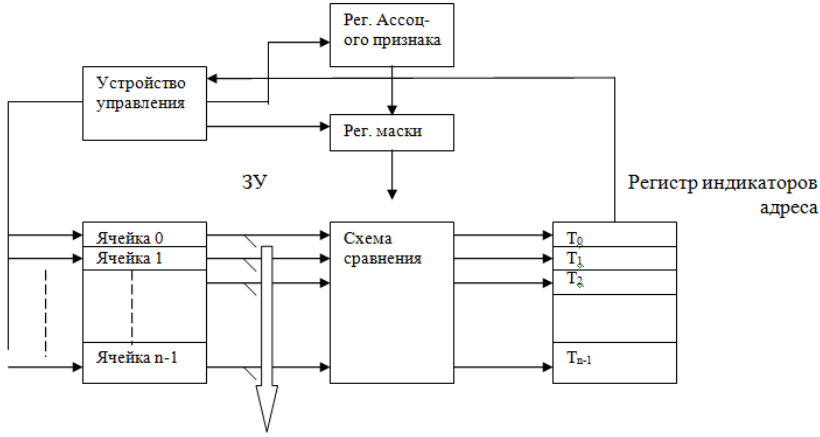


Рис. 6.9. Структура ассоциативной памяти

Рассмотрим алгоритм работы устройства. В регистр ассоциативного признака (РАП) УУ передает признак искомой информации (компаранд). Код может содержать произвольное число разрядов от 1 до n-1. Если код не полностью перекрывает разрядную сетку, то в этом случае ненужные разряды маскируются с помощью регистра маски (РМ). Перед началом поиска все разряды в регистре индикаторов адреса (РИА) устанавливаются в «1». После этого опрашиваются все первые разряды ячеек запоминающего устройства (ЗУ). Если содержимое первого разряда i- той ячейки не совпадает с маскированным содержимым 1-го разряда РАП, то соответствующий этой ячейке разряд регистра РИА сбрасывается в «0», если совпадает — на T_i остается «1». Затем операция повторяется с 2-ым, 3-им и т.д. разрядом. После поразрядного опроса и сравнения в «1» останутся те разряды РИА которые соответствуют ячейкам содержащим нужную информацию (совпадение с РАП). Эта информация может быть считана с помощью УУ. Время поиска зависит только от числа разрядов РАП и не зависит от числа ячеек ЗУ.

Стековая память

Стековая память состоит из ячеек, связанных друг с другом разрядными цепями передачи слов. Обмен информацией всегда выполняется только через верхнюю ячейку — *вершину стека.* При записи нового слова (команды, числа, символа) все ранее записанные слова сдвигаются на одну ячейку вниз, а новое слово помещается на вершину стека. Считывание возможно только с вершины стека и производится с удалением или без удаления считываемого слова. Такую память часто называют памятью типа LIFO (Last — In First — Out последним вошел, первым вышел). Аппаратная реализация стека сложна и обычно стек моделируют программно. При этом в качестве стека обычно используется часть адресной памяти.

Сегментная организация памяти.

Принцип адресации при сегментной организации памяти довольно прост. Представим себе девятиэтажную гостиницу. Каждому этажу присвоен номер от 0 до 9, т.е. на четвертом этаже будут комнаты 401, 402 … вплоть до 499. Здесь все просто и понятно. Теперь несколько модернизируем систему отсчета, введем понятие *сегмент.* Будем считать сегментом любую группу из 100 комнат одного этажа. Присвоим каждому сегменту порядковый номер из двух цифр, а не одной как было ранее. Чтобы получить любой номер комнаты на этом этаже в пределах от 0 до 99 введем понятие *смещение.*

Допустим, номер комнаты — 541 (линейный адрес). Если представить его в виде [сегмент]: [смещение], то можно сказать, что вы живете в комнате с адресом сегмента 54 (номер комнаты 540) и смещением 1 от начала сегмента, в результате мы получим линейный адрес комнаты — 541. Тот же результат мы можем получить и другими способами сегментной адресации:

Адрес сегмента	Смещение	Адрес
54	1	541
50	41	541
45	91	541

Очевидно, полный адрес состоит из суммы адреса сегмента (умноженного на 10, что реально осуществляется сдвигом на один разряд) и адреса смещения.

54	50	45
+ 1	+ 41	+ 91
541	541	541

Используя различные сегменты и смещения, можно получить один и тот же адрес. Необходимо иметь в виду, что линейный адрес не делится на две составляющие, как и номер комнаты в реальной гостинице. Чтобы лучше понять, как из двух 16 битовых слов получается 20 битовый адрес, еще раз покажем это на двоичном примере.

Пусть имеется 2 машинных слова:

ABCD H и 1234 H

(вспомним, что каждая 16-ричная цифра представляет 4 бита, например — $1111_2 = 15_{10} = F_{16}$, символ H — означает, что число 16-ричное, от англ. hexadecimal);

Берем первое число и приписываем 0 справа (если больше нравится - умножаем на 16), получим ABCD0, т.е. первоначальное число, но со смещением на один 16-ричный разряд. Это 20-ти разрядное число, но оно не может нам служить 20 битовым адресом, т.к. оно заканчивается на нуль и, следовательно может представлять только адреса которые тоже заканчиваются на нуль. Другими словами оно пригодно для адресации каждого 16-го байта или сегмента.

$$A*16^4 + B*16^3 + C*16^2 + D*16^1 + 0*16^0$$

При изменении A,B,C,D изменяется дискретно сегмент (параграф).

Чтобы получить окончательную схему сегментированной адресации возьмем другое 16-ти битовое число — 1234 H и сложим его с модифицированным первым:

```
    ABCD0
+    1234
----------
```

получим ACF04 , двадцати битовое число, которое может принимать любые значения от 0 до 1 048 577

Таким образом, ABCD0 — сегментная часть, которая указывает на ту область адреса, которая кратна 16 (сегмент, параграф. Второе число 1234 — смещение, которое указывает конкретное расположение байта внутри сегментного параграфа (это дает 65536 или 64 Кбайт адресов внутри сегмента).

Существует стандарт записи этих сегментированных адресов: ABCD:1234 (ассемблер), т.е. внутри сегмента применяется линейная адресация.

Например, на плате основного адаптера SCSI может быть установлено ПЗУ на 16 Кбайт, ячейкам которого выделен следующий диапазон адресов D4000 — D7FFF, при этом сегмент определяет 4 старших разряда, а смещение - 4 младших разряда. Поскольку эти составляющие перекрываются, конечный адрес можно получить различными способами:

```
D000:7FFF  =  D0000   (сегмент)
           +  7FFF    (смещение)
           -----------
              D7FFF   (полный адрес)
```

Отметим, что линейный адрес является пятиразрядным, хотя мы использовали для сегмента и смещения четырехразрядные числа (шестнадцатеричной системы счисления). Можно констатировать, что вычисление сегментированного адреса основано на так называемом сегментном сложении, позволяющем получить (например) 20-битовое двоичное число из двух 16 битовых.

Таким образом мы смогли адресоваться к каждой ячейке 16 Кбайт — ного блока из почти 1 Мбайт объема общей памяти, используя только 16 разрядную шину адреса, а не 20 разрядную, как это было бы в случае прямой адресации.

Основные характеристики памяти:

- Емкость памяти определяется максимальным количеством хранимой информации;
- Удельная плотность записи;
- Плотность записи;

- Быстродействие измеряется временем цикла обращения к памяти стирания прежней информации с магнитных носителей или процесс подготовки поверхности CD-RW).

Время обращения называется также временем цикла работы памяти или временем цикла доступа памяти.

6.11. Вопросы и задания для самоконтроля

1. Что такое основная память в ВМ и ВС?
2. Опишите классификация памяти по специфике использования.
3. В чём отличие статической памяти от динамической?.
4. Что такое DRAM.
5. Опишите структуру организации блока памяти 2D.
6. Опишите структуру организации блока памяти 3D.
7. Опишите структуру организации блока памяти 2.5D
8. Как происходит регенерация памяти и зачем она нужна? Назовите различные методы регенерации.
9. Что такое SRAM?
10. Опишите классификацию памяти в зависимости от возможности записи и перезаписи данных (ROM, PROM, EPROM, EEPROM, Flash Memory).
11. Аппаратный контроль корректности работы памяти.
12. Расскажите как работает контроль четности.
13. Расскажите как работает контроль ECC.
14. Логическая организация памяти. Основные характеристики памяти.
15. Опишите особенности сегментной организации памяти. Что такое «сегмент» и «смещение»? Продемонстрируйте вычисление адреса.
16. Каким должно быть значение бита контроля чётности для бита данных, восемь бит которого содержат 0 0 1 0 0 0 0 1?
17. Каким должно быть значение бита контроля чётности для бита данных, восемь бит которого содержат 0 0 1 0 0 0 1 1?
18. Каким должно быть значение бита контроля чётности для бита данных, восемь бит которого содержат 1 0 1 1 1 0 1 1?
19. Каким должно быть значение бита контроля чётности для бита данных, восемь бит которого содержат 1 1 1 1 1 1 1 1?
20. Вычислите адрес памяти, на который ссылается Segment:Offset, равные 0203:5BDO.
21. Вычислите адрес памяти, на который ссылается Segment:Offset, равные 0048:7780.

22. Вычислите адрес памяти, на который ссылается Segment:Offset, равные 0204:5BC0.
23. Вычислите адрес памяти, на который ссылается Segment:Offset, равные FFFF:FFFF. Это максимальное значение адреса, на который может ссылаться комбинация S:O из двух машинных слов.

7. ОРГАНИЗАЦИЯ ВВОДА/ВЫВОДА ИНФОРМАЦИИ В ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ И СИСТЕМАХ

7.1. Типы интерфейсов

В этой главе речь идет об основных интерфейсах ввода-вывода современных персональных компьютеров. Основными средствами коммуникации, используемыми в PC, являются последовательные и параллельные порты.

К последовательным портам чаще подключаются двунаправленные устройства, которые должны как передавать информацию в компьютер, так и принимать ее. Последовательные порты, параллельные порты, универсальная последовательная шина (Universal Serial Bus — USB), IEEE-1394 (i.Link или FireWire) — все это интерфейсы ввода вывода. SCSI и IDE — это тоже интерфейсы ввода-вывода.

Параллельные порты обычно используются для подключения принтеров и работают в однонаправленном режиме, хотя могут применяться и как двунаправленные. Существуют версии сетевых адаптеров, накопителей на магнитной ленте, сканеров, дисководов для гибких дисков и CD-ROM, которые также подключаются к параллельным портам. Задачи, традиционно выполняемые последовательными или параллельными портами, в настоящее время постепенно переходят к портам более новых типов, вроде USB или IEEE-1394. Тем не менее традиционные порты все еще остаются одними из самых важных интерфейсов ввода-вывода.

Последовательные порты

Асинхронный последовательный интерфейс — это основной тип интерфейса, с помощью которого осуществляется взаимодействие между компьютерами. Термин асинхронный означает, что при передаче данных не используются никакие синхронизирующие сигналы и отдельные символы могут передаваться с произвольными интервалами, как, например, при вводе данных с клавиатуры.

Термин последовательный означает, что передача данных осуществляется по одиночному проводнику, а биты при этом передаются последовательно, один за другим. Такой тип связи характерен для телефонной сети, в которой каждое направление обслуживает один проводник. Типичные системы включают в себя один или два последовательных порта, располагаемых обычно на

задней панели системного блока. В современных конструкциях системных плат для управления встроенными последовательными портами этого типа используется микросхема Super I/O, расположенная на системной плате, или высокоинтегрированная микросхема SouthBridge.

Параллельные порты

В параллельных портах для одновременной передачи байта информации используется восемь линий. Этот интерфейс отличается высоким быстродействием, часто применяется для подключения к компьютеру принтера, а также для соединения компьютеров. (Ведь при этом скорость передачи данных значительно выше, чем при соединении через последовательные порты: 4, а не 1 бит за раз). Существенным недостатком параллельного порта является то, что соединительные провода не могут быть слишком длинными. При большой длине соединительного кабеля в него приходится вводить промежуточные усилители сигналов, так как в противном случае возникает множество помех.

Стандарт IEEE 1284

Этот стандарт был окончательно утвержден в марте 1994 года. В нем определены физические характеристики параллельных портов (режимы передачи данных и т.д.). Кроме того, в стандарте IEEE 1284 описан характер изменения внешних сигналов, поступающих на многорежимные параллельные порты компьютера, т.е. на порты, которые могут работать в 4- и 8-разрядном режимах, а также в режимах EPP и ECP. Хотя IEEE 1284 был выпущен для стандартизации форм сигналов, с помощью которых компьютер “общается” с подключаемыми устройствами, в частности с принтером, этот стандарт интересен и для производителей периферийных устройств, подключаемых к параллельным портам (дисководов, сетевых адаптеров и др.).

Поскольку IEEE 1284 предназначен только для аппаратного обеспечения и не содержит требований к программному обеспечению, работающему с параллельными портами, вскоре был разработан стандарт, определяющий требования к такому программному обеспечению и направленный на устранение различий между микросхемами параллельных портов разных производителей. В нем, в частности, описана спецификация для поддержки режима EPP через BIOS. Стандартом IEEE 1284 предусмотрена более высокая пропускная способность соединения между компьютером и принтером

или двумя компьютерами. Для реализации этой возможности стандартный кабель принтера не подходит. Стандартом IEEE 1284 для принтера предусмотрена витая пара.

В стандарте IEEE 1284 определен также новый разъем. Разъем типа A определен как штыревой DB25, разъем типа B — как Centronics 36. Разъем типа C является разъемом высокой плотности. Такие разъемы (типа C) устанавливаются на принтерах Hewlett-Packard. Стандартные параллельные порты. В первом компьютере IBM PC существовал только один параллельный порт, предназначенный для передачи информации от компьютера к какому-либо устройству, например к принтеру. Однонаправленность параллельного порта первого PC вполне соответствовала его основному назначению — передаче данных на принтер. Однако во многих случаях желательно было иметь двунаправленный параллельный порт даже для принтера (чтобы можно было реализовать обратную связь, например для принтеров типа PostScript). С однонаправленным параллельным портом осуществить это было невозможно. Такой тип параллельных портов не предназначался для использования в качестве ввода, однако с помощью специальных схем (в которых четыре сигнальные линии могут быть представлены как 4-разрядное соединение) и однонаправленного параллельного порта можно обеспечить 8-разрядный вывод и 4-разрядный ввод. В настоящее время этот тип портов используется довольно редко, так как в компьютерах, выпущенных после 1993 года, как правило, устанавливаются параллельные порты наподобие 8-разрядного, EPP и ECP. Стандартный параллельный порт обеспечивает скорость передачи данных 50 Кбайт/с, но при использовании различных усовершенствований пропускную способность можно увеличить до 150 Кбайт/с.

USB и 1394 (i.Link) FireWire — новые интерфейсы ввода- вывода

В настоящее время для настольных и портативных компьютеров разработано два высокоскоростных устройства с последовательной шиной: USB (Universal Serial Bus — универсальная последовательная шина) и IEEE-1394, называемая также i.Link или FireWire. Эти высокоскоростные коммуникационные порты отличаются от стандартных параллельных и последовательных портов, установленных в большинстве современных компьютеров, более широкими возможностями. Преимущество новых портов состоит в том, что их можно использовать как альтернативу SCSI для высокоскоростных соединений с периферийными устройствами, а

также подсоединять к ним все типы внешних периферийных устройств (т.е. в данном случае предпринята попытка объединения устройств ввода-вывода).

Новым направлением в развитии высокоскоростных периферийных шин является использование последовательной архитектуры. Для передачи информации в параллельной архитектуре, где биты передаются одновременно, необходимы линии, имеющие 8, 16 и более проводов. Можно предположить, что за одно и то же время через параллельный канал передается больше данных, чем через последовательный, однако на самом деле увеличить пропускную способность последовательного соединения намного легче, чем параллельного. Параллельное соединение обладает рядом недостатков, одним из которых является фазовый сдвиг сигнала, из-за чего длина параллельных каналов, например SCSI, ограничена (не должна превышать 3 м). Проблема в том, что, хотя 8- и 16-разрядные данные одновременно пересылаются передатчиком, из-за задержек одни биты прибывают в приемник раньше других. Следовательно, чем длиннее кабель, тем больше время задержки между первым и последним прибывшими битами на приемном конце.

Последовательная шина позволяет единовременно передавать 1 бит данных. Отсутствие задержек при передаче данных позволяет значительно увеличить тактовую частоту. Напри мер, максимальная скорость передачи данных параллельного порта EPP/ECP достигает 2 Мбайт/с, в то время как порты IEEE-1394 (в которых используется высокоскоростная последовательная технология) поддерживают скорость передачи данных, равную 400 Мбит/с (около 50 Мбайт/с), т.е. в 25 раз выше.

Еще одно преимущество последовательного способа передачи данных — возможность использования только одно- или двухпроводного канала, поэтому помехи, возникающие при передаче, очень малы, чего нельзя сказать о параллельном соединении. Стоимость параллельных кабелей довольно высока, поскольку провода, предназначенные для параллельной передачи, не только используются в большом количестве, но и специальным образом укладываются, чтобы предотвратить возникновение помех, а это весьма трудно. Кабели для последовательной передачи данных, напротив, очень дешевые, так как состоят из нескольких проводов и требования к их экранированию намного ниже, чем у используемых для параллельных соединений. Именно поэтому, а также учитывая требования внешнего периферийного интерфейса Plug and Play и необходимость устранения физического нагромождения портов в

портативных компьютерах, были разработаны эти две высокоскоростные последовательные шины, используемые уже сегодня. Несмотря на то что шина IEEE 1394 была изначально предназначена для узкоспециализированного использования (например, с видеокамерами стандарта DV), в настоящий момент она применяется и с другими устройствами наподобие профессиональных сканеров и внешних жестких дисков.

Универсальная последовательная шина USB

В USB реализована возможность подключения большого количества периферийных устройств к компьютеру. При подключении устройств к USB не нужно устанавливать платы в разъемы системной платы и реконфигурировать систему; кроме того, экономно используются такие важные системные ресурсы, как IRQ (запросы прерывания). При подключении периферийного оборудования к персональным компьютерам, оснащенным шиной USB, его настройка происходит автоматически, сразу после физического подключения, без перезагрузки или установки. Основным инициатором разработки стандарта USB выступила Intel. Начиная с набора микросхем системной логики Triton II (82430HX), в котором стандарт USB был воплощен в микросхеме PIIX3 South Bridge, компания Intel поддерживает этот стандарт во всех своих на борах микросхем системной логики. Совместно с Intel над созданием универсальной последовательной шины работало еще семь компаний, среди которых Compaq, Digital, IBM, Microsoft, NEC и Northern Telecom. Ими был создан USB Implement Forum (USB-IF), целью которого является развитие, поддержка и распространение архитектуры USB.

Первая версия USB анонсирована в январе 1996 года, а версия 1.1 — в сентябре 1998. В этой спецификации более подробно описаны концентраторы и другие устройства. Большинство USB-устройств должны быть совместимы со спецификацией 1.1, даже если они выпущены до ее официального опубликования. В появившейся относительно недавно спецификации USB 2.0 скорость передачи данных в 40 раз выше, чем в оригинальной USB 1.0, кроме того, обеспечивается полная обратная совместимость устройств. Платы расширения PCI (для настольных систем) и платы PC Card Cardbus совместимых портативных компьютеров позволяют модернизировать компьютеры ранних версий, не имеющие встроенных разъемов USB. Большинство плат расширения поддерживают спецификацию USB 2.0,

IEEE-1394 (FireWire или i.Link)

В конце 1995 года отдел стандартов Института инженеров по электротехнике и электронике (Institute of Electrical and Electronic Engineers — IEEE) опубликовал стандарт IEEE- 1394 (или сокращенно 1394). Эти цифры — порядковый номер нового стандарта, который явился результатом обширных исследований в области мультимедийных устройств. Его основное преимущество заключается в высокой скорости передачи данных. На сегодняшний день скорость передачи, поддерживаемая этим стандартом, достигает 400 Мбит/с. Текущая версия стандарта 1394 получила на данный момент название 1394a (иногда ее называют по году опубликования стандарта — 1394a-2000). Стандарт 1394a предназначен для решения проблем, связанных с совместимостью и многофункциональностью, которые существовали в оригинальной версии стандарта 1394. В этом стандарте используются те же разъемы и поддерживаются те же скорости передач, что и в оригинальном стандарте 1394. Предлагаемый стандарт 1394b, как ожидается, будет поддерживать скорость передачи данных, равную 1 600 Мбит/с; скорости передач будущих версий этого стандарта смогут достичь 3 200 Мбит/с. Стандарт 1394b будет поддерживать более высокие скорости, чем существующие в настоящее время стандарты 1394/1394a. Это связано с внедрением новых сетевых технологий, в частности стеклянного и пластикового волоконно-оптических кабелей и кабеля UTP 5-й категории, а также с увеличением возможного расстояния между устройствами, использующими кабельное соединение 5-й категории, и улучшением принципа передачи сигналов. Стандарт 1394b будет обратно совместим с устройствами 1394a.

FireWire — это высокоскоростная локальная последовательная шина, способная передавать данные со скоростью 100, 200 и 400 Мбит/с (12,5, 25 и 50 Мбайт/с), а при работе с некоторыми типами файлов — до 1 Гбит/с. Эта шина использует простой 6-проводной кабель, со стоящий из двух различных пар линий, предназначенных для передачи тактовых импульсов и информации, а также двух линий питания. Как и USB, IEEE-1394 полностью поддерживает технологию Plug-and-Play, включая возможность горячего подключения (установка и извлечение компонентов без отключения питания системы). По структуре шина 1394 не так сложна, как параллельная шина SCSI, и устройства, подключаемые к ней, могут потреблять от нее ток до 1,5 А. По производительности шина IEEE-1394 превосходит Ultra-Wide SCSI, стоит гораздо меньше, а подсоединить устройства к ней намного проще. На рис. 17.10 показаны компоненты разъемов шины 1394. Шина 1394 построена на разветвляющейся топологии и позволяет

использовать до 63 узлов в цепочке и подсоединять при этом к каждому узлу до 16 устройств. Если этого недостаточно, то можно дополнительно подключить до 1 023 шинных перемычек, которые могут соединять более 64 000 узлов! Кроме того, шина 1394 может поддерживать устройства, построенные на одной шине, но работающие на разных скоростях передачи данных, как и SCSI. Большинство адаптеров 1394 имеют три узла, каждый из которых поддерживает 16 устройств. Подключить к компьютеру через шину 1394 можно практически все устройства, которые могут работать со SCSI. Сюда входят все виды дисковых накопителей, включая жесткие, оптические, CD- и DVD-ROM. К шине 1394 могут подключаться цифровые видеокамеры, устройства с записью на магнитную ленту и многие другие высокоскоростные периферийные устройства. Вероятно, очень скоро шина 1394 начнет широко использоваться как в настольных, так и в портативных компьютерах, а с течением времени заменит все другие высокоскоростные шины. В настоящее время наборы микросхем системной логики, поддерживающие шину 1394, уже предлагаются производителями. Появились адаптеры PCI, позволяющие добавить поддержку 1394 в существующие компьютеры. В настоящее время шина 1394 получила наиболее широкое распространение в области цифровых видеоустройств (камеры, видеомагнитофоны и т.д.). Подобные устройства выпускают компании Sony, Panasonic, Sharp, Matsushita и др. Кроме цифровых видеоустройств стали появляться устройства обработки видеоданных. Например, компании Adaptec и Texas Instruments выпускают адаптеры PCI, поддерживающие IEEE-1394. После появления USB 2.0 скорости передачи данных IEEE-1394 и USB практически одинаковы (скорость передачи данных IEEE-1394 и USB 1.1 отличаются в 16 раз). Поэтому обсуждать преимущества или недостатки сравниваемых технологий по этому параметру нецелесообразно.

Для подключения периферии USB необходимо узловое устройство (чаще всего это компьютер), в то время как устройства IEEE-1394 можно подключать напрямую. Именно поэтому технология IEEE-1394 получила наибольшее распространение в цифровых видеоустройствах.

7.2. Основные принципы организации ввода/вывода

Как известно, ввод-вывод считается одной из самых сложных областей проектирования операционных систем, в которой сложно применить общий подход и в которой изобилуют частные методы. В действительности, источником сложности является огромное число

устройств ввода-вывода разнообразной природы, которые должна поддерживать операционная система. При этом перед создателями операционной системы встает очень непростая задача — не только обеспечить эффективное управление устройствами ввода-вывода, но и создать удобный и эффективный виртуальный интерфейс устройств ввода-вывода, позволяющий прикладным программистам просто считывать или сохранять данные, не обращая внимание на специфику устройств и проблемы распределения устройств между выполняющимися задачами. Система ввода-вывода, способная объединить в одной модели широкий спектр устройств, должна быть универсальной. Она должна учитывать потребности существующих устройств, от простой мыши до клавиатур, принтеров, графических дисплеев, дисковых накопителей, компакт-дисков и даже сетей. С другой стороны, необходимо обеспечить доступ к устройствам ввода-вывода для множества параллельно выполняющихся задач, причем так, чтобы они как можно меньше мешали друг другу.

Поэтому самым главным является следующий принцип: любые операции по управлению вводом-выводом объявляются привилегированными и могут выполняться только кодом самой операционной системы. Для обеспечения этого принципа в большинстве процессоров даже вводятся режимы пользователя и супервизора. Последний так же называют привилегированным режимом, или режимом ядра. Как правило, в режиме супервизора выполнение команд ввода-вывода разрешено, а в пользовательском режиме — запрещено. Обращение к командам ввода-вывода в пользовательском режиме вызывает исключение, и управление через механизм прерываний передается коду операционной системы. Хотя возможны и более сложные схемы, в которых в ряде случаев пользовательским программам может быть разрешено непосредственное выполнение команд ввода-вывода.

Еще раз подчеркнем, что мы, прежде всего, говорим о мультипрограммных операционных системах, для которых существует проблема разделения ресурсов, и одним из основных видов ресурсов являются устройства ввода-вывода и соответствующее программное обеспечение, с помощью которого осуществляется обмен данными между внешними устройствами и оперативной памятью. Помимо разделяемых устройств ввода-вывода (эти устройства допускают разделение посредством механизма доступа) существуют неразделяемые устройства. Примерами разделяемого устройства могут служить накопитель на магнитных дисках, устройство чтения компакт-

дисков. Это устройства с прямым доступом. Примеры неразделяемых устройств — принтер, накопитель на магнитных лентах. Это устройства с последовательным доступом. Операционные системы должны управлять и теми, и другими, предоставляя возможность параллельно выполняющимся задачам их использовать.

Можно назвать три основные причины, по которым нельзя разрешать каждой отдельной пользовательской программе обращаться к внешним устройствам непосредственно.

- Необходимость разрешать возможные конфликты в доступе к устройствам ввода-вывода. Например, пусть две параллельно выполняющиеся программы пытаются вывести на печать результаты своей работы. Если не предусмотреть внешнего управления устройством печати, то в результате мы можем получить абсолютно нечитаемый текст, так как каждая программа будет время от времени выводить свои данные, перемежающиеся с данными от другой программы. Либо можно взять ситуацию, когда для одной программы необходимо прочитать данные с одного сектора магнитного диска, а для другой записать результаты в другой сектор того же накопителя. Если операции ввода-вывода не будут отслеживаться каким-то третьим (внешним) процессом-арбитром, то после позиционирования магнитной головки для первой задачи может тут же прийти команда позиционирования головки для второй задачи, и обе операции ввода-вывода не смогут выполниться корректно.
- Желание увеличить эффективность использования ресурсов ввода-вывода. Например, у накопителя на магнитных дисках время подвода головки чтения/записи к необходимой дорожке и время обращения к определенному сектору могут значительно (до тысячи раз) превышать время пересылки данных. В результате, если задачи по очереди обращаются к цилиндрам, далеко отстоящим друг от друга, то полезная работа, выполняемая накопителем, может быть существенно снижена.
- Необходимость избавить программы ввода-вывода от ошибок. Ошибки в программах ввода-вывода могут привести к краху всех вычислительных процессов, ибо часть операций ввода-вывода требуются самой операционной системе. В ряде операционных систем системный ввод-вывод имеет существенно более высокие привилегии, чем ввод-вывод задач пользователя. Поэтому системный код, управляющий операциями ввода-вывода, очень тщательно отлаживается и оптимизируется для повышения

надежности вычислений и эффективности использования оборудования.

Итак, управление вводом-выводом осуществляется компонентом операционной системы, который часто называют супервизором ввода-вывода. Перечислим основные задачи, возлагаемые на супервизор.

1. Модуль супервизора операционной системы, иногда называемый супервизором задач, получает запросы от прикладных задач на выполнение тех или иных операций, в том числе на ввод-вывод. Эти запросы проверяются на корректность и, если они соответствуют спецификациям и не содержат ошибок, то обрабатываются дальше. В противном случае пользователю (задаче) выдается соответствующее диагностическое сообщение о недействительности (некорректности) запроса.
2. Супервизор ввода-вывода получает запросы на ввод-вывод от супервизора задач или от программных модулей самой операционной системы.
3. Супервизор ввода-вывода вызывает соответствующие распределители каналов и контроллеров, планирует ввод-вывод (определяет очередность предоставления устройств ввода-вывода задачам, затребовавшим эти устройства). Запрос на ввод-вывод либо тут же выполняется, либо ставится в очередь на выполнение.
4. Супервизор ввода-вывода инициирует операции ввода-вывода (передает управление соответствующим драйверам) и в случае управления вводом-выводом с использованием прерываний предоставляет процессор диспетчеру задач с тем, чтобы передать его первой задаче, стоящей в очереди на выполнение.
5. При получении сигналов прерываний от устройств ввода-вывода супервизор идентифицирует эти сигналы и передает управление соответствующим программам обработки прерываний.
6. Супервизор ввода-вывода осуществляет передачу сообщений об ошибках, если таковые происходят в процессе управления операциями ввода-вывода.
7. Супервизор ввода-вывода посылает сообщения о завершении операции ввода-вывода запросившей эту операцию задаче и снимает ее с состояния ожидания ввода-вывода, если задача ожидала завершения операции.

В случае если устройство ввода-вывода является инициативным, управление со стороны супервизора ввода-вывода будет заключаться в активизации соответствующего вычислительного процесса (перевод его в состояние готовности к выполнению).

Таким образом, прикладные программы (а в общем случае - все обрабатывающие программы) не могут непосредственно связываться с устройствами ввода-вывода независимо от того, в каком режиме используются эти устройства (монопольно или совместно), но, установив соответствующие значения параметров в запросе на ввод-вывод, определяющие требуемую операцию и количество потребляемых ресурсов, обращаются к супервизору задач. Последний передает управление супервизору ввода-вывода, который и запускает необходимые логические и физические операции.

Упомянутый выше запрос на ввод-вывод должен удовлетворять требованиям API той операционной системы, в среде которой выполняется приложение. Параметры, которые указываются в запросах на ввод-вывод, передаются не только в вызывающих последовательностях, создаваемых по спецификациям API, но и как данные, хранящиеся в соответствующих системных таблицах. Все параметры, которые будут стоять в вызывающей последовательности, предоставляются компилятором и отражают требования программиста, а также постоянные сведения об операционной системе и архитектуре компьютера в целом. Переменные сведения о вычислительной системе (ее конфигурация, состав оборудования, состав и особенности системного программного обеспечения) содержатся в специальных системных таблицах. Процессору, каналам прямого доступа в память и контроллерам необходимо передавать конкретную двоичную информацию, с помощью которой и осуществляется управление оборудованием. Эта конкретная двоичная информация в виде кодов и данных часто готовится с помощью препроцессоров, но часть ее хранится в системных таблицах.

7.3. Специфика подсистем ввода/вывода

Эффективность использования вычислительных возможностей ЭВМ определяется не только возможностями ее процессора и характеристиками основной памяти, но также составом ее периферийных устройств, их техническими характеристиками и способами организации их совместной работы с ядром (процессор и основная память) компьютера.

При разработке подсистемы ввода/вывода должны быть решены следующие проблемы.

Должна быть обеспечена возможность реализации машины с переменной конфигурацией (то есть с переменным составом оборудования). В первую очередь, пользователь должен иметь

возможность легко дополнять машину новыми устройствами, изменять состав периферийных устройств в соответствии с назначением ЭВМ.

Должна реализовываться параллельная во времени работа процессора над программой и выполнение периферийными устройствами процедур ввода/вывода.

Для пользователя должно быть упрощено и стандартизировано программирование операций ввода/вывода, обеспечена независимость программирования ввода/вывода от особенностей того или иного периферийного устройства.

Должны быть обеспечены автоматическое распознавание различных ситуаций, возникающих в периферийных устройствах, и реакция ядра ЭВМ на эти ситуации (будь то готовность устройства, различные нарушения его работы или отсутствие носителей).

Наиболее актуально решение этих проблем для ЭВМ с большим количеством разнообразных устройств.

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.

Унифицированные (не зависящие от типа периферийных устройств) форматы данных, которыми периферийные устройства обмениваются с ядром ЭВМ, в том числе и унифицированный формат сообщения, которое периферийное устройство посылает в ядро о своем состоянии. Преобразование в индивидуальные форматы данных осуществляют контроллеры и адаптеры.

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 выбор 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.

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).

Использование множества периферийных устройств ПУ с относительно небольшим быстродействием приводит к тому, что основное время работы процессора занято обработкой операций ввода-вывода и ожиданием готовности внешних устройств к обмену. Это существенно снижает быстродействие ЭВМ. Эффективность работы ЭВМ можно повысить путем разделения функций управления центральным процессором и периферийными устройствами.

7.4. Шины. Мезонинные шины

Шины, как известно, используются для передачи данных от центрального процессора к другим устройствам персонального компьютера. Для того, чтобы согласовать передачу данных к отдельным компонентам, работающих на своей частоте, используется чипсет — набор контроллеров, конструктивно объединенных в Северный и Южный мосты. Северный мост отвечает за обмен информацией с оперативной памятью и видеосистемой, Южный — за функционирование других устройств, подключаемых через соответствующие разъемы — жесткие диски, оптические накопители, а также устройств, находящихся на материнской плате (встроенная аудиосистема, сетевое устройство и др.), и для внешних устройств — клавиатура, мышь и т.д.

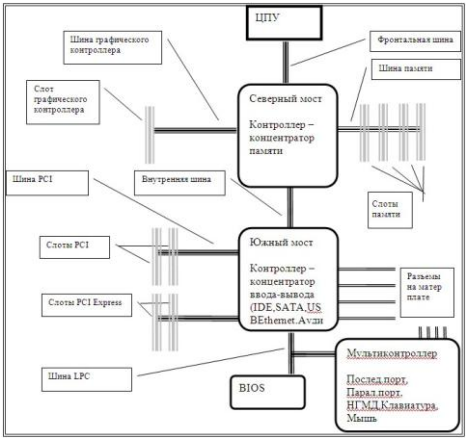


Рис. 7.1. Схема системной платы

Для связи процессора с мостами используется шина FSB (Front Side Bus) (наиболее часто используемые в настоящее время Hyper-Transport и SCI), северный мост (иногда называемый системным контроллером) позволяет функционировать наиболее производительным устройствам — видеоадаптеру с помощью шины PCI Express 16x и оперативной памяти через шину памяти. Южный мост обеспечивает работу менее скоростных устройств, подключаемых с помощью карт расширения (аудиокарты, сетевые карты, видеокарты и т.д.) через шины PCI и шину PCI Express, оптических дисководов и жестких дисков через шины ATA (ранее называемых IDE, сейчас имеют название PATA (Parallel ATA) и более современные шины SATA. Еще более медленные устройства подключены к южному мосту через шину LPC(шина с маленькой пропускной способностью) — микросхема BIOS, мультиконтроллер для связи с внешними устройствами через последовательные и параллельные порты — клавиатурой, мышью, принтером и др.

Мезонинная плата, мезонин — плата, вставляемая в основную плату (носитель) и располагающаяся параллельно плате-носителю. Носитель (ISA, PCI, VMEbus, CompactPCI) может иметь несколько слотов для размещения мезонин-модулей и, следовательно, допускает гибкую функциональную конфигурацию.

Сегодня большинство разработчиков промышленных и телекоммуникационных компьютерных систем предпочитают использовать готовые платы. Но многие идут по пути собственных разработок, объясняя этот выбор двумя причинами: или нет платы с теми функциями, которые нужны, или такая плата есть, но на ней есть лишние функции, которые не нужны и поэтому ее применение невыгодно. Действительно, несмотря на огромный парк готовых плат (ISA, PCI, VME, CompactPCI, PC/104, PC/104+), иногда не удается подобрать необходимую конфигурацию. Использование мезонинных технологий позволяет решить эту проблему.

Основными международными мезонинными стандартами, применяемыми сегодня, являются:

- IndustryPack (стандарт VITA 4-1995)
- PMC (PCI Mezzanine Card) (стандарт IEEE 1386.1)
- PC-MIP (стандарт VITA-29).
- XMC

7.5. История развития шин на ПРИМЕРЕ IBM PC (ISA, EISA, VLB, PCI, EXTENDED PCI)

ISA (от англ. Industry Standard Architecture, ISA bus) — 8- или 16-разрядная шина ввода-вывода IBM PC-совместимых компьютеров. Служит для подключения плат расширения стандарта ISA. Конструктивно выполняется в виде 62-х или 98-контактного разъёма на материнской плате.

Впервые шина ISA появилась на компьютерах IBM PC/XT в 1981 году. Это была 8-разрядная шина с частотой до 8 МГц и скоростью передачи данных до 4 Мбайт/с (передача каждого байта требовала минимум двух тактов шины). Разъём состоял из 62 контактов, из которых 8 использовалось для данных, 20 — для адреса, остальные — для управляющих сигналов, а также подачи напряжений питания (GND, +5 В, −5 В, +12 В и −12 В).

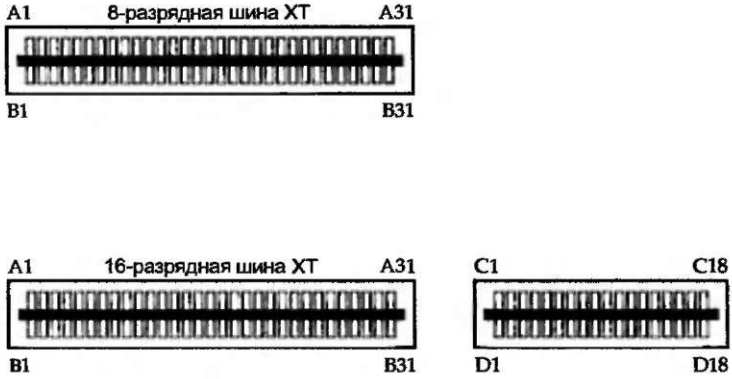


Рис. 7.2. Схема 8- и 16-разрядных разъемов шины.

В 1984 году шина была усовершенствована. Была удвоена разрядность данных (что повлекло удвоение пропускной способности) и добавлены четыре разряда адреса; кроме того, увеличилось число линий запросов прерываний и запросов прямого доступа к памяти (DMA). Кроме того, в 16-разрядной шине ISA любое подключенное к ней устройство могло выступать в роли задатчика, то есть инициировать операцию обмена данными (в 8-разрядной шине задатчиками были только процессор и контроллер DMA). Для подключения 16-разрядных устройств используются разъёмы, состоящие из двух частей: полностью совместимой с 8-разрядной шиной 62-контактной и новой 36-контактной.

EISA (англ. Extended Industry Standard Architecture) — шина для IBM-совместимых компьютеров. Была анонсирована в конце 1988 консорциумом из девяти основных производителей IBM-совместимых компьютеров (Compaq, Hewlett-Packard, Epson, NEC, Olivetti, AST Research, Tandy ruen, Wyse и Zenith) как ответ на введение фирмой IBM новой скоростной (по сравнению с устаревающей ISA), но проприетарной, шины MCA в компьютерах серии PS/2.

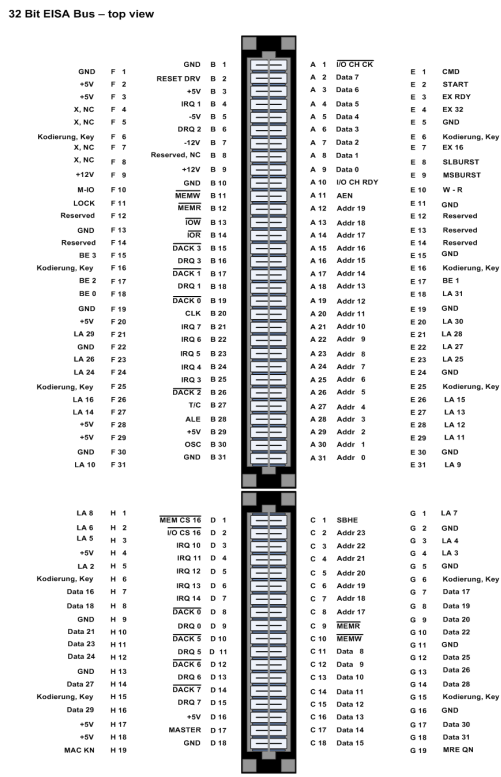


Рис. 7.3. Схема шины EISA.

EISA расширяет распространённую шину ISA до 32 разрядов и позволяет подключать к шине более одного ЦПУ. Адресное пространство, по сравнению с ISA, увеличено до 4 ГБ. Кроме того, шина поддерживает bus mastering. EISA является надмножеством ISA,

поэтому, в отличие от MCA, к ней можно подключать старые платы, предназначенные для работы с 8- и 16- разрядными версиями ISA: имеется как электрическая, так и механическая совместимость.

VESA local bus — VL-Bus или VLB — тип локальной шины, разработанный ассоциацией VESA для ПК. Шина VLB, по существу, является расширением внутренней шины МП Intel 80486 для связи с видеоадаптером и реже с контроллером HDD. Реальная скорость передачи данных по VLB — 80 Мбайт/с (теоретически достижимая — 132 Мбайт/с).

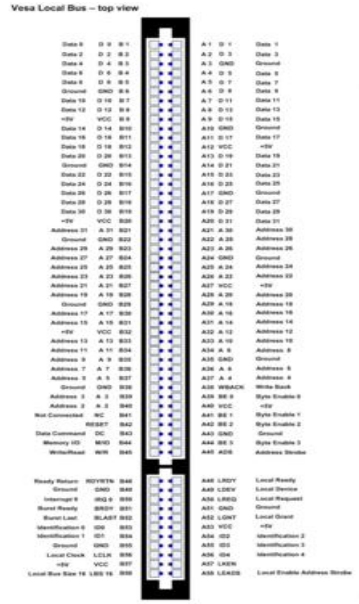


Рис. 7.4. Схема VLB шины.

Разработана в 1992 г. Ассоциацией стандартов видеооборудования (VESA — Video Electronics Standards Association), поэтому часто ее называют шиной VESA. Главной целью её разработки была дешёвая альтернатива шинам MicroChannel и EISA, пригодная для внедрения в массовые настольные компьютеры. С этой ролью шина VLB успешно справилась. Было выпущено большое количество плат контроллеров, использовавших эту шину, на основе выпущенных ранее микросхем, работавших до этого с шиной ISA. Даже при 16-битной архитектуре мог быть получен выигрыш от в 4 раза большей тактовой частоты

PCI (англ. Peripheral component interconnect) — шина ввода-вывода для подключения периферийных устройств к материнской плате компьютера.

Стандарт на шину PCI определяет:

- физические параметры (например, разъёмы и разводку сигнальных линий);
- электрические параметры (например, напряжения);
- логическую модель (например, типы циклов шины, адресацию на шине).

Развитием стандарта PCI занимается организация PCI Special Interest Group.

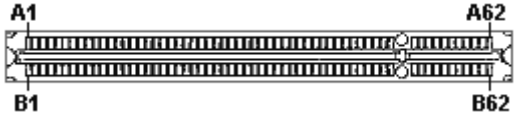


Рис. 7.5. Схема PCI.

PCI-устройства с точки зрения пользователя самонастраиваемы (Plug and Play). После старта компьютера системное программное обеспечение обследует конфигурационное пространство PCI каждого устройства, подключённого к шине, и распределяет ресурсы.

Каждое устройство может затребовать до шести диапазонов в адресном пространстве памяти PCI или в адресном пространстве ввода-вывода PCI.

Кроме того, устройства могут иметь ПЗУ, содержащее исполняемый код для процессоров x86 или PA-RISC, Open Firmware (системное ПО компьютеров на базе SPARC и PowerPC) или драйвер EFI.

Настройка прерываний осуществляется также системным программным обеспечением (в отличие от шины ISA, где настройка прерываний осуществлялась переключателями на карте). Запрос на прерывание на шине PCI передаётся с помощью изменения уровня сигнала на одной из линий IRQ, поэтому имеется возможность работы нескольких устройств с одной линией запроса прерывания; обычно системное ПО пытается выделить каждому устройству отдельное прерывание для увеличения производительности.

PCI Extended

PCI-X 1.0 — расширение шины PCI64 с добавлением двух новых частот работы, 100 и 133 МГц, а также механизма раздельных транзакций для улучшения производительности при одновременной работе нескольких устройств. Как правило, обратно совместима со всеми 3.3В и универсальными PCI-картами.

PCI-X карты обычно выполняются в 64-бит 3,3 В формате и имеют ограниченную обратную совместимость со слотами PCI64/66, а некоторые PCI-X карты — в универсальном формате и способны работать (хотя практической ценности это почти не имеет) в обычном PCI 2.2/2.3.

В сложных случаях для того, чтобы быть полностью уверенным в работоспособности комбинации из материнской платы и карты расширения, надо посмотреть таблицы совместимости (compatibility lists) производителей обоих устройств.

PCI-X 2.0 — дальнейшее расширение возможностей PCI-X 1.0; добавлены частоты 266 и 533 МГц, а также - коррекция ошибок чётности при передаче данных (ECC). Допускает расщепление на 4 независимых 16-битных шины, что применяется исключительно во встраиваемых и промышленных системах; сигнальное напряжение снижено до 1,5 В, но сохранена обратная совместимость разъёмов со всеми картами, использующими сигнальное напряжение 3,3 В.

7.6. Вопросы для самоконтроля

1. Перечислите основные типы интерфейсов.
2. Расскажите о стандарте IEEE 1284**.**
3. Дайте краткую характеристику интерфейса USB.
4. Какова специфика подсистем ввода/вывода?
5. Назовите задачи, выполняемые супервизором ввода/вывода?
6. Что такое шина, и какова её роль в системе ВМ?
7. Что такое Plug and play? Приведите несколько примеров.
8. Что такое мезонинные шины? В чём их предназначение?
9. Как осуществляется синхронизация на шине?
10. История развития и особенности шины IBM PC — ISA.
11. История развития и особенности шины IBM PC — EISA.
12. История развития и особенности шины IBM PC — VLB.
13. История развития и особенности шины IBM PC — PCI,
14. В чём отличие PCI-E от PCI?
15. В чём отличие PCI от USB?
16. В чём отличие bus master от bus slave?

8. АРХИТЕКТУРА КОМПЬЮТЕРНЫХ СЕТЕЙ

8.1. История развития сетей

*В **1957*** году, после запуска Советским Союзом первого искусственного спутника Земли, Министерство обороны США посчитало, что на случай войны Америке нужна надёжная система передачи информации. Агентство по перспективным оборонным научно-исследовательским разработкам США (Defense Advanced Research Projects Agency - агентство передовых оборонных исследовательских проектов) предложило разработать для этого компьютерную сеть. Разработка такой сети была поручена Калифорнийскому университету в Лос-Анджелесе, Стэнфордскому исследовательскому центру, Университету Юты и Университету штата Калифорния в Санта-Барбаре. Компьютерная сеть была названа ARPANET (англ. Advanced Research Projects Agency Network), и в 1969 году в рамках проекта сеть объединила четыре указанных научных учреждения. Все работы финансировались Министерством обороны США. Затем сеть ARPANET начала активно расти и развиваться, её начали использовать учёные из разных областей науки.

Первый сервер ARPANET был установлен ***2 сентября 1969 года*** в Калифорнийском университете (Лос-Анджелес). Компьютер Honeywell DP-516 имел 24 Кб оперативной памяти.

29 октября 1969 года в 21:00 между двумя первыми узлами сети ARPANET, находящимися на расстоянии в 640 км — в Калифорнийском университете Лос-Анджелеса (UCLA) и в Стэнфордском исследовательском институте (SRI) — провели сеанс связи. Чарли Клайн (Charley Kline) пытался выполнить удалённое подключение из Лос-Анджелеса к компьютеру в Стэнфорде. Успешную передачу каждого введённого символа его коллега Билл Дювалль (Bill Duvall) из Стэнфорда подтверждал по телефону.

В первый раз удалось отправить всего три символа «LOG», после чего сеть перестала функционировать. LOG должно было быть словом LOGIN (команда входа в систему). В рабочее состояние систему вернули уже к 22:30, и следующая попытка оказалась успешной. Именно эту дату можно считать днём рождения Интернета.

К ***1971*** году была разработана первая программа для отправки электронной почты по сети. Эта программа сразу стала очень популярна.

В ***1973*** году к сети были подключены через трансатлантический телефонный кабель первые иностранные организации из Великобритании и Норвегии, сеть стала международной.

В ***1970-х*** годах сеть в основном использовалась для пересылки электронной почты, тогда же появились первые списки почтовой рассылки, новостные группы и доски объявлений. Однако в то время сеть ещё не могла легко взаимодействовать с другими сетями, построенными на других технических стандартах. К концу 1970-х годов начали бурно развиваться протоколы передачи данных, которые были стандартизированы в 1982—1983 годах. Активную роль в разработке и стандартизации сетевых протоколов играл Джон Постел. 1 января 1983 года сеть ARPANET перешла с протокола NCP на TCP/IP, который успешно применяется до сих пор для объединения (или, как ещё говорят, «наслоения») сетей. Именно в 1983 году термин «Интернет» закрепился за сетью ARPANET.

В ***1984*** году была разработана система доменных имён (англ. Domain Name System, DNS).

В ***1984*** году у сети ARPANET появился серьёзный соперник: Национальный научный фонд США (NSF) основал обширную межуниверситетскую сеть NSFNet (англ. National Science Foundation Network), которая была составлена из более мелких сетей (включая известные тогда сети Usenet и Bitnet) и имела гораздо бóльшую пропускную способность, чем ARPANET. К этой сети за год подключились около 10 тыс. компьютеров, название «Интернет» начало плавно переходить к NSFNet.

В ***1988*** году был разработан протокол Internet Relay Chat (IRC – протокол общения), благодаря чему в Интернете стало возможно общение в реальном времени (чат).

В ***1989*** году в Европе, в стенах Европейского совета по ядерным исследованиям (ЦЕРН) родилась концепция Всемирной паутины. Её предложил знаменитый британский учёный Тим Бернерс-Ли, он же в течение двух лет разработал протокол HTTP, язык HTML и идентификаторы URI.

В ***1990*** году сеть ARPANET прекратила своё существование, полностью проиграв конкуренцию NSFNet. В том же году было зафиксировано первое подключение к Интернету по телефонной линии (т. н. «дозвóн», англ. dialup access).

В 1993 году появился знаменитый веб-браузер NCSA Mosaic. Всемирная паутина набирала популярность.

В ***1995*** году NSFNet вернулась к роли исследовательской сети, маршрутизацией всего трафика Интернета теперь занимались сетевые провайдеры, а не суперкомпьютеры Национального научного фонда.

В том же ***1995*** году Всемирная паутина стала основным поставщиком информации в Интернете, обогнав по трафику протокол пересылки файлов FTP. Был образован Консорциум Всемирной паутины (W3C). Можно сказать, что Всемирная паутина преобразила Интернет и создала его современный облик. С 1996 года Всемирная паутина почти полностью подменяет собой понятие «Интернет».

В ***1990-е*** годы Интернет объединил в себе большинство существовавших тогда сетей (хотя некоторые, как Фидонет, остались обособленными). Объединение выглядело привлекательным благодаря отсутствию единого руководства, а также благодаря открытости технических стандартов Интернета, что делало сети независимыми от бизнеса и конкретных компаний. К ***1997*** году в Интернете насчитывалось уже около 10 млн компьютеров, было зарегистрировано более 1 млн доменных имён. Интернет стал очень популярным средством для обмена информацией.

В настоящее время подключиться к Интернету можно через спутники связи, радио-каналы, кабельное телевидение, телефон, сотовую связь, специальные оптико-волоконные линии или электропровода. Всемирная сеть стала неотъемлемой частью жизни в развитых и развивающихся странах.

В течение пяти лет Интернет достиг аудитории свыше 50 миллионов пользователей. Другим средствам коммуникации требовалось гораздо больше времени для достижения такой популярности:

Информационная среда	Время, лет
Радио	38
Телевидение	13
Кабельное телевидение	10
Интернет	5

С 22 января 2010 года прямой доступ в Интернет получил экипаж Международной космической станции**.**

8.2. Пакетная обработка заданий

Двадцать лет назад компьютерное сообщество живо обсуждало альтернативу двух режимов эксплуатации вычислительных систем -

интерактивного доступа и пакетной обработки заданий. Появление ПК склонило мнение в сторону первого варианта, и графический интерфейс пользователя стал непременным компонентом всех современных вычислительных систем. Однако и пакетная обработка сохранилась в своей естественной нише - области интенсивных вычислений с характерными временами счета, измеряемыми в часах и сутках.

Эта область ассоциируется, в первую очередь, с суперкомпьютерами и Unix-системами, однако их базовые средства работают в рамках отдельной машины. С другой стороны, высокопроизводительные вычислительные системы могут быть построены из серийно выпускаемых компонентов путем их объединения сетевыми технологиями. Организация вычислений в такой распределенной среде компьютеров - это задача систем управления пакетной обработки (СУПО), расширяющих возможности локальных ОС.

Современные СУПО способны управлять различными комплексами: многопроцессорными системами SMP и MPP, кластерами на специализированных коммутаторах, конгломератом рабочих станций и серверов, соединенных стандартными сетевыми протоколами, и любой смесью указанных конфигураций.

Основные понятия СУПО

Базовый объект СУПО - пакетное задание (batch job), которое может выполняться без вмешательства пользователя и, следовательно, не требовать интерактивного входа в систему. Форма представления пакетного задания - скрипт командной оболочки Shell, который может включать команды для создания рабочей среды, компиляции, сборки программы и запуска ее на выполнение.

Система пакетной обработки принимает задания, вводимые пользователями с распределенных рабочих мест, и буферизует их в общем спуле ожидания (рис. 8.1). Отдельный процесс - диспетчеризация, выбирает из спула задания, находит для каждого подходящие исполнительные узлы, пересылает на них задание и запускает его. Диспетчеризация происходит периодически, и за один цикл не обязательно все задания из спула будут запущены - для некоторых может не найтись подходящих узлов.

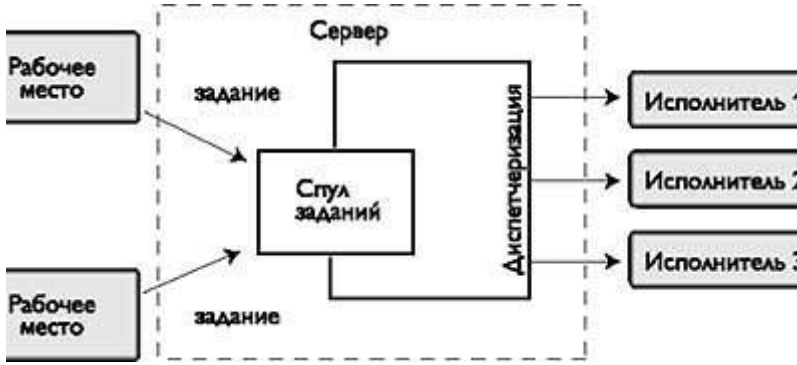


Рис. 8.1. Схема функционирования СУПО.

Наиболее существенные различия между СУПО заключены в механизмах диспетчеризации - управлении потоком заданий. Что же касается уровня обслуживания отдельных заданий, то здесь наблюдается более однородная картина. Задача СУПО - обеспечить надежную, контролируемую обработку задания в распределенной среде, включающей: машину (рабочее место) пользователя, с которой вводится задание; один или несколько исполнительных узлов, на котором задание обрабатывается; узел, содержащий файлы данных для задания; буферизующие и управляющие узлы самой СУПО.

В общем случае обработка заданий в распределенной среде предусматривает выполнение следующих функций:

- Централизованное управление заданиями
- Пост/пре доставка файлов на исполнительный узел
- Пересылка стандартных файлов на машину, с которой произведен запуск задания
- Мониторинг задания
- Установка зависимостей между заданиями
- Автоматическое восстановление задания после сбоя СУПО или исполнительного узла (рестарт, контрольные точки)
- Гарантированное выделение ресурсов, требующихся заданию при исполнении, и контроль за лимитами ресурсов.

8.3. Режим разделения времени

Коммутация на основе техники разделения частот разрабатывалась в расчете на передачу непрерывных сигналов, представляющих голос. При переходе к цифровой форме

представления голоса была разработана новая техника мультиплексирования, ориентирующаяся на дискретный характер передаваемых данных.

Эта техника носит название мультиплексирования с разделением времени (Time Division Multiplexing, TDM). Реже используется и другое ее название - техника синхронного режима передачи (Synchronous Transfer Mode, STM). Рисунок 8.2 поясняет принцип коммутации каналов на основе техники TDM.

Аппаратура TDM - сетей - мультиплексоры, коммутаторы, демультиплексоры - работает в режиме разделения времени, поочередно обслуживая в течение цикла своей работы все абонентские каналы. Цикл работы оборудования TDM равен 125 мкс, что соответствует периоду следования замеров голоса в цифровом абонентском канале. Это значит, что мультиплексор или коммутатор успевает вовремя обслужить любой абонентский канал и передать его очередной замер далее по сети. Каждому соединению выделяется один квант времени цикла работы аппаратуры, называемый также тайм - слотом. Длительность тайм - слота зависит от числа абонентских каналов, обслуживаемых мультиплексором TDM или коммутатором. Понятие и назначение экспертной системы (ЭС). Классификация ЭС. В повседневной жизни мы постоянно сталкиваемся с экспертами в самых различных областях человеческой деятельности- врачи, преподаватели, референты и т.д. Имея огромный багаж знаний, касающихся предметной области, они умеют точно сформулировать и правильно решить задачу.

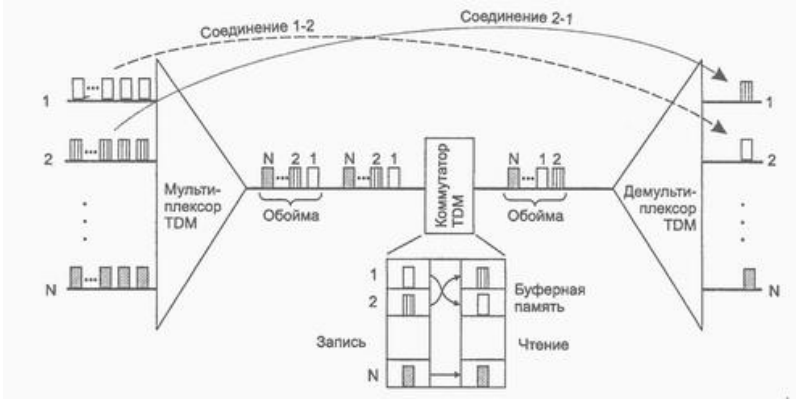


Рис. 8.2. Коммутация на основе разделения канала во времени.

Мультиплексор принимает информацию по N входным каналам от конечных абонентов, каждый из которых передает данные по абонентскому каналу со скоростью 64 Кбит/с - 1 байт каждые 125 мкс. В каждом цикле мультиплексор выполняет следующие действия:

- Прием от каждого канала очередного байта данных
- Составление из принятых байтов уплотненного кадра, называемого также обоймой
- Передача уплотненного кадра на выходной канал с битовой скоростью, равной Nx64 Кбит/с.

Порядок байт в обойме соответствует номеру входного канала, от которого этот байт получен. Количество обслуживаемых мультиплексором абонентских каналов зависит от его быстродействия. Например, мультиплексор T1, представляющий собой первый промышленный мультиплексор, работавший по технологии TDM, поддерживает 24 входных абонентских канала, создавая на выходе обоймы стандарта T1, передаваемые с битовой скоростью 1,544 Мбит/с.

Демультиплексор выполняет обратную задачу - он разбирает байты уплотненного кадра и распределяет их по своим нескольким выходным каналам, при этом он считает, что порядковый номер байта в обойме соответствует номеру выходного канала.

Коммутатор принимает уплотненный кадр по скоростному каналу от мультиплексора и записывает каждый байт из него в отдельную ячейку своей буферной памяти, причем в том порядке, в котором эти байты были упакованы в уплотненный кадр. Для выполнения операции коммутации байты извлекаются из буферной памяти не в порядке поступления, а в таком порядке, который соответствует поддерживаемым в сети соединениям абонентов. Так, например, если первый абонент левой части сети рис. 54 должен соединиться со вторым абонентом в правой части сети, то байт, записанный в первую ячейку буферной памяти, будет извлекаться из нее вторым. "Перемешивая" нужным образом байты в обойме, коммутатор обеспечивает соединение конечных абонентов в сети.

Однажды выделенный номер тайм - слота остается в распоряжении соединения "входной канал - выходной слот" в течение всего времени существования этого соединения, даже если передаваемый трафик является пульсирующим и не всегда требует захваченного количества тайм - слотов. Это означает, что соединение в сети TDM всегда обладает известной и фиксированной пропускной способностью, кратной 64 Кбит/с.

Работа оборудования TDM напоминает работу сетей с коммутацией пакетов, так как каждый байт данных можно считать некоторым элементарным пакетом. Однако, в отличие от пакета компьютерной сети, "пакет" сети TDM не имеет индивидуального адреса. Его адресом является порядковый номер в обойме или номер выделенного тайм - слота в мультиплексоре или коммутаторе. Сети, использующие технику TDM, требуют синхронной работы всего оборудования, что и определило второе название этой техники - синхронный режим передач (STM). Нарушение синхронности разрушает требуемую коммутацию абонентов, так как при этом теряется адресная информация. Поэтому перераспределение тайм - слотов между различными каналами в оборудовании TDM невозможно, даже если в каком-то цикле работы мультиплексора тайм-слот одного из каналов оказывается избыточным, так как на входе этого канала в этот момент нет данных для передачи (например, абонент телефонной сети молчит).

8.4. Сетевые архитектуры, сети ARCNET, TOKEN RING и ETHERNET

Сетевая архитектура - это комбинация стандартов, топологий и протоколов, необходимых для создания работоспособной сети. В соответствии со стандартными протоколами физического уровня выделяют три основные сетевые архитектуры: Ethernet (протокол 802,3) и Fast Ethernet (протокол 802,30); ArcNet (протокол 802,4); Token Ring (протокол 802.5). Рассмотрим каждую из сетевых архитектур более подробно.

Ethernet

Это самая популярная в настоящее время сетевая архитектура. Она использует:

- физические топологии "шина", "звезда" или "звезда - шина";
- логическую топологию "шина";
- узкополосную передачу данных со скоростями 10 и 100 Мбит/с;
- метод доступа - CSMA/CD.

Среда передачи является пассивной, т. е. получает питание от PC. Сеть прекратит работу из-за физического повреждения или неправильного подключения терминатора. Передает информацию кадрами, формат которых представлен на рис. 8.3 ниже.

Начало кадра (преамбула)	Адрес		Тип протокола	Данные	Циклический избыточный код для проверки ошибок
	Приемника	источника			

Рис. 8.3. Формат кадра в Ethernet.

Поле "Тип протокола" используется для идентификации протокола сетевого уровня (IPX и IP) - маршрутизируемый или нет. Спецификация Ethernet выполняет функции физического и канального уровня модели OSI. Различают несколько стандартов сетевых архитектур Ethernet:

- 0BaseT - на основе витой пары;
- 10Base2 - на тонком коаксиале;
- 10Base5 - на толстом коаксиале;
- 10BaseFL - на оптоволокне;
- 10BaseX - со скоростью передачи 100 Мбит/с, который включает в себя ряд спецификаций в зависимости от среды передачи.

Рассмотрим наиболее распространенные стандарты данной архитектуры, применяемые при построении ЛВС.

Стандарт 10BaseT

Физическая топология представляет собой "звезду" на основе витой пары, соединяющей все узлы сети с концентратором, используя две пары проводов: одну для передачи, другую - для приема (рис. ниже). Логически (т.е. по системе передачи сигналов) данная архитектура представляет собой "шину" как и все архитектуры Ethernet. Концентратор выступает как многопортовый репитер. Длина сегмента от 2,5 до 100 м. ЛВС стандарта 10BaseT может обслуживать до 1024 компьютеров.

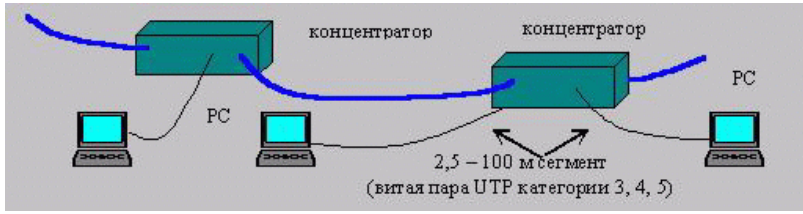


Рис. 8.4. Сеть стандарта 10BaseT.

Достоинством является возможность использования распределительных стоек и панелей коммутации, что позволяет легко перекоммутировать сеть или добавить новый узел без остановки работы сети. Новейшие концентраторы позволяют расширять топологию сети, соединив отдельные концентраторы между собой магистралью на основе коаксиального или оптоволоконного кабеля и получить топологию "звезда - шина".

Стандарт 10Base2

Сеть такого типа ориентирована на тонкий коаксиальный кабель с максимальной длиной сегмента 185 м и возможностью подключения к одному сегменту до 30 ЭВМ (рис. ниже).

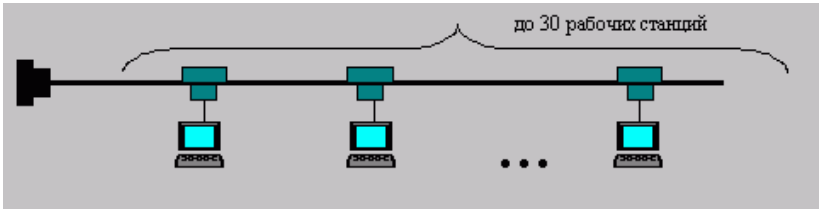


Рис. 8.5. Архитектура сети стандарта 10Base2.

Эта сетевая архитектура физически и логически представляет собой "шину". С использованием репитеров может быть увеличена общая протяженность сети введением дополнительных сегментов. Однако при этом необходимо учитывать правило **5-4-3**. Сеть на тонком коаксиале может состоять максимум из 5 сегментов кабеля, соединенных 4 репитерами. При этом только к 3 сегментам можно подключать рабочие станции. Два из пяти сегментов являются межрепитерными связями и служат только для увеличения длины сети (рис. ниже). Максимальное число компьютеров до 1024, а общая длина сети до 925м.

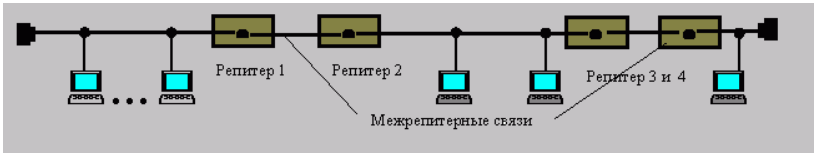


Рис. 8.6. Правило 5-4-3 для сети стандарта 10Base2.

Стандарт 10Base5

Сетевая архитектура на толстом Ethernet логически и физически представляет собой "шину" (рис. ниже). Магистральный сегмент (т. е. главный кабель, к которому подключаются трансиверы для связи с PC) имеет длину до 500 м и возможность подключения до 100 компьютеров. С использованием репитеров, которые также подключаются к магистральному сегменту через трансиверы, общая длина сети может составить 2500 м.

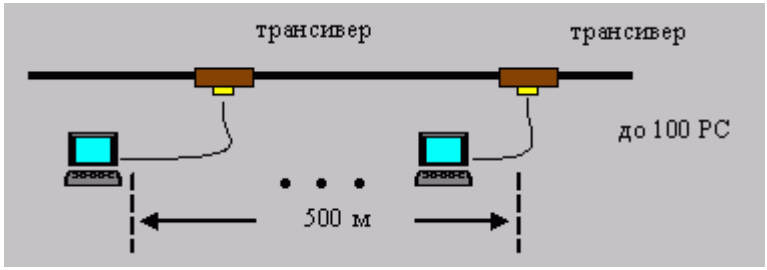


Рис. 8.7. Сеть стандарта 10Base5.

При расширении сети справедливо правило 5-4-3 и возможно комбинирование тонкого и толстого кабеля. В этом случае в качестве магистрали, способной передавать данные не большие расстояния, используется толстый кабель, а в качестве ответвляющих сегментов используют тонкий.

Стандарт 10BaseFL

Данная архитектура строится на оптоволоконном кабеле, доступ к которому со стороны компьютеров и репитеров осуществляется с помощью трансиверов (рис. ниже). На сегодняшний день в основном используются внешние трансиверы.

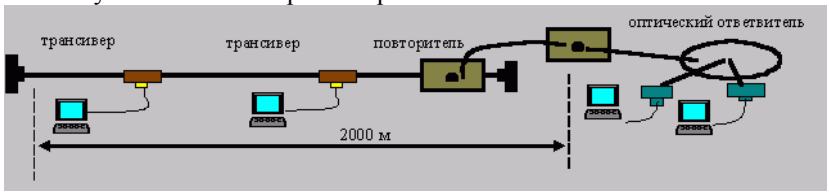


Рис. 8.8. Сеть стандарта 10BaseFL.

Особенность этих трансиверов в том, что их передатчики преобразуют электрические сигналы от ЭВМ в световые импульсы, а

приемники - световые в электрические. Популярность использования 10BaseFL обусловлена:

- высокой помехозащищенностью;
- возможностью прокладки кабеля между репитерами на большие расстояния, т. к. длина сегмента до 2 - 4 км;
- использование повторителей позволяющих реализовать "каскадные звезды" путем соединения оптических ответвителей.

На рынке предлагаются ответвители типа коаксиал - волокно и ответвители типа волокно - коаксиал.

Стандарт 100BaseX Ethernet

Этот стандарт, иногда называемый Fast Ethernet, является расширением существующей сетевой архитектуры Ethernet и соответствует протоколу физического уровня IEEE 802.30. Его особенностью является то, что он сохранил стандартный для Ethernet метод доступа CSMA/CD, от которого отходили разработчики других технологий повышенной скорости передачи в сети. Сохранение метода доступа означает, что имеющиеся в наличие драйверы для Ethernet будут работать без изменений.

Преимуществом этой технологии, появившейся в конце 1993 года, является то, что степень ее совместимости с Ethernet-сетями, позволяет интегрировать ее в эти сети с помощью двухскоростных сетевых адаптеров или мостов. Данная сетевая архитектура использует физическую топологию "звезда" или "звезда - шина" (подобно 10BaseT), где все кабели подключаются к концентратору (рис. 7.7). Различают три спецификации среды:

- 100BaseT4 (UTR категории 3, 4 или 5 с 4-мя парами);
- 100BaseTX (UTR или STP категории 5 с 2-мя парами);
- 100BaseFX (двужильный оптоволоконный кабель).

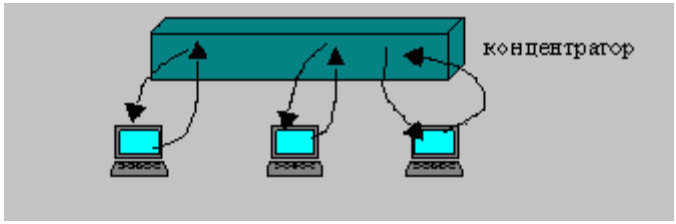


Рис. 8.9. Сеть стандарта 100BaseX Ethernet.

Для реализации этой технологии необходимо две пары проводов или двужильный оптокабель, чтобы организовать дуплексную

передачу сигналов по традиционной CSMA/CD, используя одну пару для передачи, а другую - для приема.

Сетевые архитектуры ArcNet и ArcNet Plus

Это простая, гибкая и недорогая сетевая архитектура, поддерживающая протокол физического уровня IEEE 802.4:

1. физическая топология - "звезда", "шина", "звезда - шина";
2. логическая топология - упорядоченное "кольцо";
3. широкополосная передача данных 2,5 Мбит/с и 20 Мбит/с (для ArcNet Plus);
4. метод доступа маркерный;
5. средой передачи может быть:

- коаксиальный кабель (длиной 600 м при "звезде" и 300 м при "шине");
- витая пара (максимальная длина 244 м - при "звезде" и "шине").

Компьютеры могут быть коаксиальным кабелем связаны в шину или в иных случаях подключены к концентраторам, которые могут быть: пассивными; активными; интеллектуальными. Пассивные концентраторы просто осуществляют коммутацию кабельных соединений сети. Активные - восстанавливают и ретранслируют сигнал. Интеллектуальные - обнаруживают изменения в сети и удаленно управляют работой сетевых устройств.

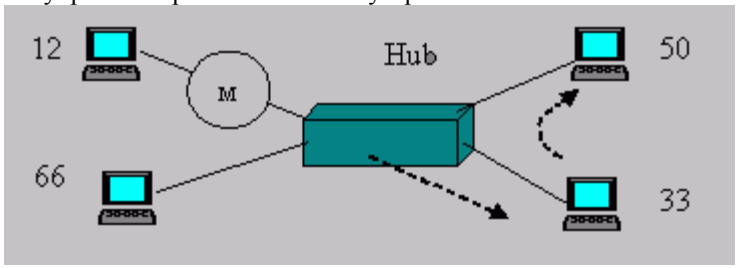


Рис. 8.10. Архитектура Arc Net.

Особенность маркерного доступа ArcNet (рис. выше) состоит в том, что:

- все компьютеры имеют свои сетевые адреса;
- маркер передается между компьютерами согласно их номерам;
- маркер двигается от компьютера с меньшим номером к компьютеру с более высоким номером, хотя тот может находиться на другом конце сети;
- приемник, получив маркер, добавляет к нему свой пакет, который, дойдя до адресата, освобождает маркер.

Формат пакета ArcNet Plus имеет вид, представленный на рис. ниже.

Рис. 8.11. Пакт передачи информации в ArcNet.

Общее количество узлов: 255 - ArcNet; 2047 - Arc Net Plus. ArcNet - это одна из самых старых сетевых архитектур, реализованная недавно фирмой DataPoint в более современную ArcNet Plus. Однако на смену этим архитектурам приходят более современные и производительные. Одной из таких архитектур является FDDI.

Token Ring (Маркерное кольцо)

Данная сетевая архитектура была разработана и внедрена фирмой IBM еще в 1984 г. как часть предложенного ею способа объединить в сеть весь ряд выпускаемых IBM компьютеров: персональные компьютеры; средние ЭВМ и мейнфреймы. Разрабатывая эту технологию, IBM ставила задачу обеспечить простоту монтажа кабеля - витой пары - соединяющего компьютер с сетью через розетку. Token Ring является реализацией протокола физического уровня IEEE 802.5:

- физическая топология - "звезда";
- логическая топология - "кольцо";
- узкополосный тип передачи;
- скорость передачи 4 и 16 Мбит/с;
- соединение неэкранированной и экранированной витой пары;
- метод доступа - маркерное кольцо.

Формат кадра имеет вид, представленный на рис. ниже.

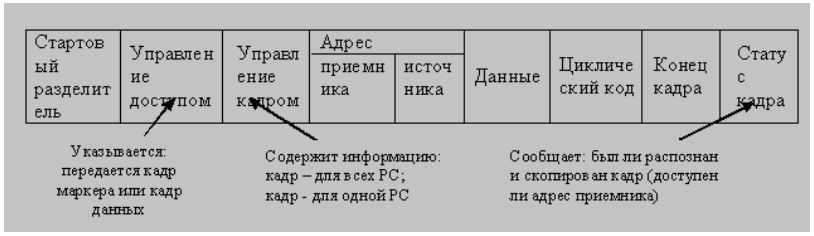


Рис. 8.12. Формат кадра, используемый в сетях Token Ring.

8.5. Коммутация пакетов и сообщений, сети массового обслуживания

Существует четыре принципиально различные схемы коммутации абонентов в сетях:

- Коммутация каналов (,circuit switching) — организация составного канала через несколько транзитных узлов из нескольких последовательно «соединённых» каналов на время передачи сообщения (оперативная коммутация) или на более длительный срок (постоянная/долговременная коммутация — время коммутации определяется административно).
- Коммутация сообщений (message switching) — разбиение информации на сообщения, которые передаются последовательно к ближайшему транзитному узлу, который, приняв сообщение, запоминает его и передаёт далее сам таким же образом. Получается нечто вроде конвейера.
- Коммутация пакетов (packet switching) — разбиение сообщения на «пакеты», которые передаются отдельно. Разница между сообщением и пакетом: размер пакета ограничен технически, сообщения — логически. При этом, если маршрут движения пакетов между узлами определён заранее, говорят о виртуальном канале (с установлением соединения). Пример: коммутация IP-пакетов. Если же для каждого пакета задача нахождения пути решается заново, говорят о датаграммном (без установления соединения) способе пакетной коммутации.
- Коммутация ячеек (cell switching) — совмещает в себе свойства сетей с коммутацией каналов и сетей с коммутацией пакетов, при коммутации ячеек пакеты всегда имеют фиксированный и относительно небольшой размер.

Система массового обслуживания (СМО) — система, которая производит обслуживание поступающих в неё требований. Обслуживание требований в СМО производится обслуживающими приборами. Классическая СМО содержит от одного до бесконечного числа приборов. В зависимости от наличия возможности ожидания поступающими требованиями начала обслуживания СМО подразделяются на:

- системы с потерями, в которых требования, не нашедшие в момент поступления ни одного свободного прибора, теряются;
- системы с ожиданием, в которых имеется накопитель бесконечной ёмкости для буферизации поступивших требований, при этом ожидающие требования образуют очередь;

- системы с накопителем конечной ёмкости (ожиданием и ограничениями), в которых длина очереди не может превышать ёмкости накопителя; при этом требование, поступающее в переполненную СМО (отсутствуют свободные места для ожидания), теряется.

Выбор требования из очереди на обслуживание производится с помощью так называемой дисциплины обслуживания. Их примерами являются FCFS/FIFO (пришедший первым обслуживается первым), LCFS/LIFO (пришедший последним обслуживается первым), random (англ.)(случайный выбор). В системах с ожиданием накопитель в общем случае может иметь сложную структуру.

8.6. Типичные конфигурации локальных сетей

Под топологией вычислительной сети понимается способ соединения ее отдельных компонентов (компьютеров, серверов, принтеров и т.д.). Различают три основные топологии:

- топология типа звезда;
- топология типа кольцо;
- топология типа общая шина.

При использовании топологии типа **звезда** информация между клиентами сети передается через единый центральный узел. В качестве центрального узла может выступать сервер или специальное устройство - **концентратор** (Hub).

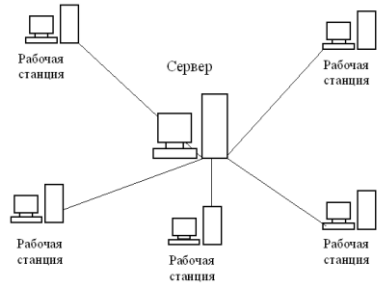


Рис. 8.13. Схема топологии типа звезда

Преимущества данной топологии состоят в следующем:

1. Высокое быстродействие сети, так как общая производительность сети зависит только от производительности центрального узла.

2. Отсутствие столкновения передаваемых данных, так как данные между рабочей станцией и сервером передаются по отдельному каналу, не затрагивая другие компьютеры.

Однако помимо достоинств у данной топологии есть и недостатки:

1. Низкая надежность, так как надежность всей сети определяется надежностью центрального узла. Если центральный компьютер выйдет из строя, то работа всей сети прекратится.
2. Высокие затраты на подключение компьютеров, так как к каждому новому абоненту необходимо ввести отдельную линию.

При топологии типа **кольцо** все компьютеры подключаются к линии, замкнутой в кольцо. Сигналы передаются по кольцу в одном направлении и проходят через каждый компьютер.

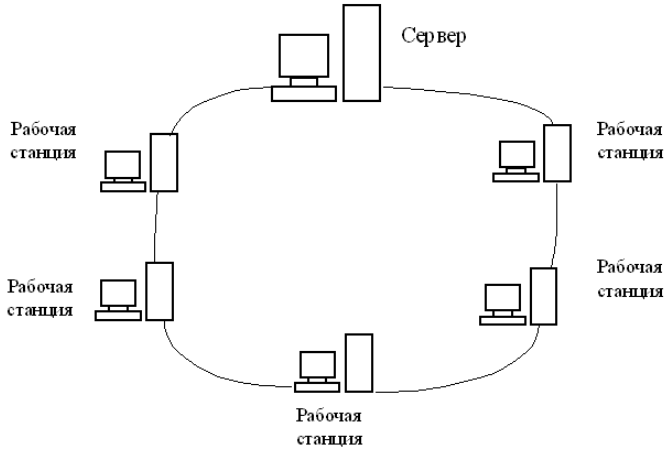


Рис. 8.14. Схема топологии типа кольцо.

Передача информации в такой сети происходит следующим образом. Маркер (специальный сигнал) последовательно, от одного компьютера к другому, передается до тех пор, пока его не получит тот, которому требуется передать данные. Получив маркер, компьютер создает так называемый "пакет", в который помещает адрес получателя и данные, а затем отправляет этот пакет по кольцу. Данные проходят через каждый компьютер, пока не окажутся у того, чей адрес совпадает с адресом получателя. После этого принимающий компьютер посылает источнику информации подтверждение факта получения данных.

Получив подтверждение, передающий компьютер создает новый маркер и возвращает его в сеть.

Преимущества топологии типа кольцо состоят в следующем:

1. Пересылка сообщений является очень эффективной, т.к. можно отправлять несколько сообщений друг за другом по кольцу. Т.е. компьютер, отправив первое сообщение, может отправлять за ним следующее сообщение, не дожидаясь, когда первое достигнет адресата.
2. Протяженность сети может быть значительной., т.е. компьютеры могут подключаться к друг к другу на значительных расстояниях, без использования специальных усилителей сигнала.

К недостаткам данной топологии относятся:

1. Низкая надежность сети, так как отказ любого компьютера влечет за собой отказ всей системы.
2. Для подключения нового клиента необходимо отключить работу сети.
3. При большом количестве клиентов скорость работы в сети замедляется, так как вся информация проходит через каждый компьютер, а их возможности ограничены.
4. Общая производительность сети определяется производительностью самого медленного компьютера.

При топологии типа **общая шина** все клиенты подключены к общему каналу передачи данных. При этом они могут непосредственно вступать в контакт с любым компьютером, имеющимся в сети.

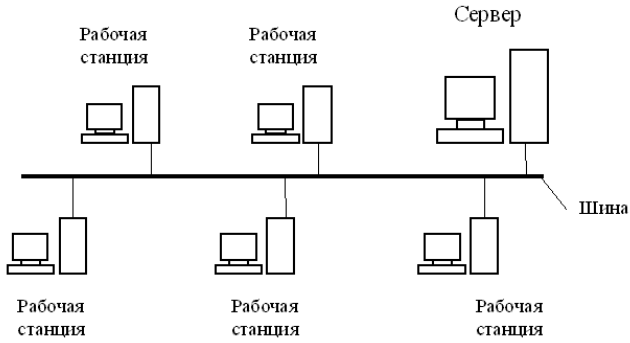


Рис. 8.15. Схема топологии типа общая шина

Передача информации в данной сети происходит следующим образом. Данные в виде электрических сигналов передаются всем компьютерам сети. Однако информацию принимает только тот

компьютер, адрес которого соответствует адресу получателя. Причем в каждый момент времени только один компьютер может вести передачу данных.

Преимущества топологии общая шина:

1. Вся информация находится в сети и доступна каждому компьютеру.
2. Рабочие станции можно подключать независимо друг от друга. Т.е. при подключении нового абонента нет необходимости останавливать передачу информации в сети.
3. Построение сетей на основе топологии общая шина обходится дешевле, так как отсутствуют затраты на прокладку дополнительных линий при подключении нового клиента.
4. Сеть обладает высокой надежностью, т.к. работоспособность сети не зависит от работоспособности отдельных компьютеров.

К недостаткам топологии типа общая шина относятся:

1. Низкая скорость передачи данных, т.к. вся информация циркулирует по одному каналу (шине).
2. Быстродействие сети зависит от числа подключенных компьютеров. Чем больше компьютеров подключено к сети, тем медленнее идет передача информации от одного компьютера к другому.
3. Для сетей, построенных на основе данной топологии, характерна низкая безопасность, так как информация на каждом компьютере может быть доступна с любого другого компьютера.

Самым распространенным типом сети с топологией **общая шина** является сеть стандарта Ethernet со скоростью передачи информации 10 - 100 Мбит/сек.

Рассмотрены основные топологии ЛВС. Однако на практике при создании ЛВС организации могут одновременно использоваться сочетание нескольких топологий. Например, компьютеры в одном отделе могут быть соединены по схеме звезда, а в другом отделе по схеме общая шина, и между этими отделами проложена линия для связи.

8.7. Коммуникационное оборудование локальных сетей: концентраторы, коммутаторы, маршрутизаторы

Сетевой адаптер (сетевая карта) - это устройство двунаправленного обмена данными между ПК и средой передачи данных вычислительной сети. Кроме организации обмена данными между ПК и вычислительной сетью, сетевой адаптер выполняет

буферизацию (временное хранение данных) и функцию сопряжения компьютера с сетевым кабелем. Сетевыми адаптерами реализуются функции физического уровня, а функции канального уровня семиуровневой модели ISO реализуются сетевыми адаптерами и их драйверами.

Адаптеры снабжены собственным процессором и памятью. Карты классифицируются по типу порта, через который они соединяются с компьютером: ISA, PCI, USB. Наиболее распространенные из них - это сетевые карты PCI. Карта, как правило, устанавливается в слот расширения PCI, расположенный на материнской плате ПК, и подключается к сетевому кабелю разъемами типа: RJ-45 или BNC.

Сетевые карты можно разделить на два типа:

- адаптеры для клиентских компьютеров;
- адаптеры для серверов.

В зависимости от применяемой технологии вычислительных сетей Ethernet, Fast Ethernet или Gigabit Ethernet, сетевые карты обеспечивают скорость передачи данных: 10, 100 или 1000 Мбит/с.

Промежуточное коммуникационное оборудование вычислительных сетей

В качестве промежуточного коммуникационного оборудования применяются: трансиверы (transceivers), повторители (repeaters), концентраторы (hubs), коммутаторы (switches), мосты (bridges), маршрутизаторы (routers) и шлюзы (gateways).

Промежуточное коммуникационное оборудования вычислительных сетей используется для усиления и преобразования сигналов, для объединения ПК в физические сегменты, для разделения вычислительных сетей на подсети (логические сегменты) с целью увеличения производительности сети, а также для объединения подсетей (сегментов) и сетей в единую вычислительную сеть.

Физическая структуризация вычислительных сетей объединяет ПК в общую среду передачи данных, т.е. образует физические сегменты сети, но при этом не изменяет направление потоков данных. Физические сегменты упрощают подключение к сети большого числа ПК.

Логическая структуризация разделяет общую среду передачи данных на логические сегменты и тем самым устраняет столкновения (коллизии) данных в вычислительных сетях. Логические сегменты или подсети могут работать автономно и по мере необходимости компьютеры из разных сегментов могут обмениваться данными между

собой. Протоколы управления в вычислительных сетях остаются теми же, какие применяются и в неразделяемых сетях.

Трансиверы и повторители обеспечивают усиление и преобразование сигналов в вычислительных сетях. Концентраторы и коммутаторы служат для объединения нескольких компьютеров в требуемую конфигурацию локальной вычислительной сети.

Концентраторы являются средством физической структуризации вычислительной сети, так как разбивают сеть на сегменты. Коммутаторы предназначены для логической структуризации вычислительной сети, так как разделяют общую среду передачи данных на логические сегменты и тем самым устраняют столкновения.

Для соединения подсетей (логических сегментов) и различных вычислительных сетей между собой в качестве межсетевого интерфейса применяются коммутаторы, мосты, маршрутизаторы и шлюзы.

Повторители — это аппаратные устройства, предназначенные для восстановления и усиления сигналов в вычислительных сетях с целью увеличения их длины.

Трансиверы или приемопередатчики — это аппаратные устройства, служащие для двунаправленной передачи между адаптером и сетевым кабелем или двумя сегментами кабеля. Основной функцией трансивера является усиление сигналов. Трансиверы применяются и в качестве конверторов для преобразование электрических сигналов в другие виды сигналов (оптические или радиосигналы) с целью использования других сред передачи информации.

Концентраторы — это аппаратные устройства множественного доступа, которые объединяют в одной точке отдельные физические отрезки кабеля, образуют общую среду передачи данных или физические сегменты сети.

Коммутаторы - это программно — аппаратные устройства, которые делят общую среду передачи данных на логические сегменты. Логический сегмент образуется путем объединения нескольких физических сегментов с помощью концентраторов. Каждый логический сегмент подключается к отдельному порту коммутатора.

Мосты — это программно — аппаратные устройства, которые обеспечивают соединение нескольких локальных сетей между собой или несколько частей одной и той же сети, работающих с разными протоколами. Мосты предназначены для логической структуризации сети или для соединения в основном идентичных сетей, имеющих

некоторые физические различия. Мост изолирует трафик одной части сети от трафика другой части, повышая общую производительность передачи данных.

Маршрутизаторы — это коммуникационное оборудование, которое обеспечивает выбор маршрута передачи данных между несколькими сетями, имеющими различную архитектуру или протоколы. Маршрутизаторы применяют только для связи однородных сетей и в разветвленных сетях, имеющих несколько параллельных маршрутов. Маршрутизаторами и программными модулями сетевой операционной системы реализуются функции сетевого уровня.

Шлюзы — это коммуникационное оборудование (например, компьютер), служащее для объединения разнородных сетей с различными протоколами обмена. Шлюзы полностью преобразовывают весь поток данных, включая коды, форматы, методы управления и т.д.

8.8. Вопросы и задания для самоконтроля

1. Назовите важные этапы история развития сетей.
2. В чём особенности систем с пакетной обработкой заданий?
3. Как работает режим разделения времени?
4. Перечислите основные сетевые архитектуры.
5. Опишите сетевую архитектуру ARCNet. Каковы её особенности?
6. Как реализована архитектура типа Token Ring?
7. Опишите сеть Ethernet и её основные характеристики.
8. Расскажите о коммутации пакетов и сообщений. Какие ещё существуют виды коммутации?
9. Что такое сети массового обслуживания?
10. Что такое сетевой адаптер? Каких видов бывают сетевые адаптеры?
11. Опишите типичные топологии локальных сетей.
12. Приведите примеры использования сетей различной топологии.
13. Назовите плюсы и минусы топологии типа звезда.
14. Назовите плюсы и минусы топологии типа кольцо.
15. Назовите плюсы и минусы топологии типа общая шина.
16. Какие виды коммуникационного оборудования локальных сетей существуют?
17. Опишите задачи, выполняемые трансиверами, повторителями, концентраторами.
18. Опишите задачи, выполняемые коммутаторами, маршрутизаторами, и их основные характеристики.
19. Опишите задачи, выполняемые мостами и шлюзами. В чём сходства и отличия этого оборудования?

СПИСОК ЛИТЕРАТУРЫ

1. Аляев Ю.А. Дискретная математика и математическая логика [текст]: справочник / Ю.А. Аляев, С.Ф. Тюрин — М.: Финансы и статистика, 2006. — 368 с
2. Баула В.Г. Архитектура ЭВМ и операционные среды [текст] : учебник для студентов / В. Г. Баула, А. Н. Томилин, Д. Ю. Волканов, Изд. Academia, 2011, 336с.
3. Башлы П. Н.Современные сетевые технологии [текст]: учебное пособие / П.Н. Башлы — М. Изд. Горячая Линия — Телеком, 2006. - 336с.
4. Богатырев В. А. Информационные системы и технологии [текст] : учебное пособие / В. А. Богатырев, Изд. Юрайт, 2016, 318с.
5. Боресков А.В. Основы работы с технологией CUDA [текст]: справочник / А. В. Боресков, А. А. Харламов — М.: ДМК Пресс, 2010. — 232с.: ил.
6. Вильям Столлингс Компьютерные сети, протоколы и технологии Интернета. [текст]: – СПб.: БХВ-Петербург, 2009. – 832 с.
7. Водяхо А. И. Архитектура информационных систем [текст]: учебник для студентов учреждений высшего профессионального образования / А. И. Водяхо, В. Дубнецкий, Изд. Academia, 2012, 288с.
8. Воеводин В. В.Параллельные вычисления [текст]: учеб. пособие / В. В. Воеводин, Вл. В. Воеводин. — СПб.: БХВ-Петербург, 2004. — 608 с.: ил.
9. Гребенюк Е. И. Технические средства информатизации [текст] : учебник для студентов учреждений среднего профессионального образования / Е. И. Гребенюк, Н. А. Гребенюк, Изд. Academia, 2013, 352с.
10. Жмакин А. П. Архитектура ЭВМ [текст]: пособие для студентов и преподавателей технических вузов / А. П. Жмакин Изд. БХВ-Петербург, 2010, 352с.
11. Иванов Н. А. Системное администрирование персонального компьютера [текст] : курс лекций для студентов / Н. А. Иванов, Изд. МГСУ, 2014, 168с.
12. Исаченко О.В. Программное обеспечение компьютерных сетей [текст]: учебное пособие предназначено для учащихся системы

среднего профессионального образования / О. В. Исаченко, Изд. Инфра-М, 2012, 120с.
13. Келим Ю. М. Вычислительная техника [текст]: учебник 10-е издание / Ю. М. Келим Изд. Академия, 2015, 368с.
14. Королёв Л. Н. Архитектура электронных вычислительных машин [текст] / Л.Н. Королёв — М. Научный мир, 2005, 272с., ил.
15. Кузьменко Н. Г. Компьютерные сети и сетевые технологии [текст]: учебное пособие / Н. Г. Кузьменко, Изд. Наука и техника, 2013, 368с.
16. Кулябов Д. С. Архитектура и принципы построения современных сетей и систем телекоммуникаций [текст]: учебное пособие. / Д. С. Кулябов, А. В. Королькова — М.: РУДН, 2008. — 281 с.: ил
17. Максимов Н. В [текст] : учебник для студентов 5-е издание / Н. В. Максимов, И. И. Попов, Т. П. Партыка, Изд. Форум, Инфра-М, 2016, 512с.
18. Максимов Н.В. Архитектура ЭВМ и вычислительных систем [текст]: учебник / Н.В. Максимов, Т.Л. Партыка, И.И. Попов — ФОРУМ:ИНФРА-М, 2005. — 512с.ил.
19. Мелехин В. Вычислительные системы и сети [текст]: учебное пособие / В. Ф. Мелехин, Е. Г. Павловский, Изд. Академия, 2013, 208с.
20. Мельников Д. А. Организация и обеспечение безопасности информационно-технологических сетей и систем [текст]: учебник для студентов государственных образовательных учреждений высшего профессионального образования / Д. А. Мельников, Изд. КДУ, 2015, 598с.
21. Новожилов О.П. Архитектура ЭВМ исистем. Учебное пособие для бакалавров. [текст] / О.П. Новожилов - М.: Изд. ЮРАЙТ, 2013. - 527 с
22. Новожилов О.П. Основы цифровой техники [текст]: учебное пособие / О.П. Новожилов. — М.: ИП РадиоСофт, 2004. — 528 с.: ил.
23. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. / В. Г. Олифер, Н. А. Олифер. — СПб.: Питер, 2010.- 312 с.
24. Пятибратов А. П. Вычислительные системы, сети и телекоммуникации [текст]: учебное пособие / А. П. Пятибратов, Л. П. Гудыно, А. А. Кириченко, Изд. КноРус, 2017, 372с.

25. Рыбальченко М. В. Архитектура информационных систем [текст]: учебное пособие для вузов / М. В. Рыбальченко, Изд. Юрайт, 2016, 92с.
26. Смирнова Е. В. Технологии современных сетей Ethernet. Методы коммутации и управления потоками данных [текст]: Е. В. Смернова, П. В. Козик – СПб.: БХВ-Петербург, 2012. – 272 с.: ил.
27. Сырецкий Г.А. Информатика. Фундаментальный курс. Том 1. Основы информационной и вычислительной техники [текст]: учебное пособие / Г.А. Сырецкий. — СПб.: БХВ-Петербург, 2005. — 832 с.: ил.
28. Таненбаум Э. Архитектура компьютера [текст]: учебное пособие / Э. Таненбаум, Т. Остин — 6-е изд. — СПб.: Питер, 2013. — 816с.: ил.
29. Трофимов В. В. Информационные технологии [текст]: учебник для студентов / В. В. Трофимов, О. П. Ильина, Изд. Юрайт, 2016, 392с.
30. Фуфаев Э. В. Пакеты прикладных программ [текст]: учебное пособие для студентов учреждений среднего профессионального образования / Э. В. Фуфаев, Л. И. Фуфаева, Изд. Academia, 2014, 352с.
31. Шелухин, О. И. Моделирование информационных систем [текст]: учебное пособие для студентов высших учебных заведений, обучающихся по специальностям "Сети и системы коммутации", "Многоканальные телекоммуникационные системы" / О. И. Шелухин .— Москва : Горячая линия-Телеком, 2011 .— 536 с.