





**THE  
McGRAW-HILL  
COMPUTER  
HANDBOOK**

**Editor in Chief  
Harry Helms**

**Overview by  
Adam Osborne**

**Foreword by  
Thomas C. Bartee**

**McGraw-Hill Book Company  
New York St. Louis San Francisco Auckland  
Bogotá Hamburg Johannesburg London Madrid  
Mexico Montreal New Delhi Panamá Paris  
São Paulo Singapore Sydney Tokyo Toronto  
1983**

# КОМПЬЮТЕРЫ

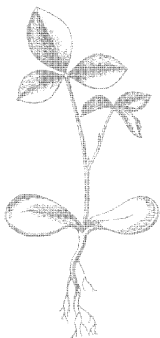
**СПРАВОЧНОЕ РУКОВОДСТВО**

**В ТРЕХ ТОМАХ**

**1**

**Под редакцией Г. Хелмса**

**Перевод с английского  
под редакцией  
д-ра техн. наук И. О. Атовяна**



**МОСКВА «МИР»  
1986**



ББК 32.97  
К 63  
УДК 681.3

Перевод выполнен Т. М. Дадашевым, В. М. Кисельниковым,  
В. С. Чепуром, В. С. Штаркманом

Хеллерман Г., Гамахер В., Вранежич З., Заки С., Кохави З.,  
Гивоне Д., Рессер Р., Голт Дж., Пиммел Р., Барти Т., Вране-  
сик З., Гизр К., Конрой Т.

**Компьютеры: Справочное руководство. В 3-х т. Т. 1. Пер.**  
К 63 с англ./Под ред. Г. Хелмса — М.: Мир, 1986. — 416 с., ил.

Справочное пособие посвящено вопросам построения и применения средств вычислительной техники. Авторы — видные специалисты США. В русском переводе книга выпускается в трех томах.

В томе I рассматриваются аппаратные средства ЭВМ. Приводятся основы машинной арифметики, булевой алгебры, цифровой схемотехники. Дается подробное описание основных компонент ЭВМ: АЛУ, ЗУ и устройств ввода-вывода. Особое внимание уделяется вопросам программного обеспечения и особенностям реализации режима разделения времени.

Для инженеров, имеющих дело с вычислительной техникой, и студентов соответствующих специальностей.

К 2405000000-139 173-86, ч. 1  
041(01)-86

ББК 32.97  
6Ф7

*Редакция литературы по информатике и электронике*

Copyright © 1983 McGraw-Hill, Inc.  
© перевод на русский язык,  
«Мир», 1986

# ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

Современные вычислительные машины представляют одно из самых значительных достижений человеческой мысли, влияние которого на развитие научно-технического прогресса трудно переоценить.

Области применения ЭВМ непрерывно расширяются. Этому в значительной степени способствует распространение персональных ЭВМ, и особенно микроЭВМ. В настоящее время даже не специалисты по вычислительной технике начинают понимать необходимость освоения принципов построения и применения ЭВМ. Однако до сих пор не было пособия, доступного широкому кругу научно-технических работников, которое охватывало бы широкий круг вопросов, связанных с технологической базой, структурой, средствами программирования и использованием ЭВМ.

Предлагаемое вниманию читателей справочное пособие в определенной степени восполняет этот пробел. Все главы с некоторыми изменениями заимствованы из различных книг соответствующей тематики. Однако благодаря тщательному подбору материала в справочнике удалось отразить современный уровень вычислительной техники. В русском переводе книга издается в трех томах.

Первый том посвящен аппаратным средствам ЭВМ. В простой и ясной форме излагаются основы машинной арифметики и алгебры логики. Рассматриваются комбинационные и последовательностные схемы, построенные на основе ТТЛ- и МОП-технологии. Типовые цифровые схемы сумматоров, счетчиков, регистров и дешифраторов, составляющих схемотехническую основу ЭВМ, описываются с единых позиций с помощью булевой алгебры. Значительное внимание уделяется упрощению булевых выражений и их схемной реализации. Достоинством книги является целостность и доступность изложения, не требующего какой-либо специальной подготовки. Книга будет понятна любому читателю со знанием математики, физики и электроники в пределах программы средней школы. Она окажется доступной

и полезной всем, кто по роду своей деятельности заинтересован в применении средств вычислительной техники, но не является специалистом в этой области.

К сожалению, в справочнике не нашли соответствующего освещения вопросы, связанные с построением устройств управления ЭВМ и канальных устройств, использующих унифицированные интерфейсы. Сведения об этих устройствах и о некоторых отечественных ЭВМ можно получить из работ, указанных в приведенном ниже списке литературы.

Перевод первого тома выполнили канд. техн. наук В. М. Кисельников (общие положения, предисловие, гл. 10), канд. физ.-мат. наук Т. М. Дадашев (гл. 1, 3, 5, 6, 8), В. С. Чепур (гл. 2, 7, 9), канд. физ.-мат. наук В. С. Штаркман (гл. 4).

### Литература

1. Малые ЭВМ и их применение. /Под общей редакцией Б. Н. Наумова./ — М.: Статистика, 1980.

2. Пржиялковский В. В., Ломов Ю. С. Технические и программные средства единой системы ЭВМ.— М.: Статистика, 1980.

3. Дроздов З. А., Комарницкий В. А., Пятибратов А. П. Электронные вычислительные машины единой системы.— М.: Машиностроение, 1981.

4. Каган Б. М. Электронные вычислительные машины и системы: Учебное пособие для вузов.— М.: Энергия, 1979.

Д-р техн. наук *И. О. Атовмян*

## ТЕНДЕНЦИИ В ПРОИЗВОДСТВЕ микроЭВМ

Стратегия разработки и производства персональной ЭВМ IBM PC, безусловно, отражает состояние производства микроЭВМ в 1982 г., а может быть и в 80-х годах вообще. Появление в 1982 г. персональной микроЭВМ IBM PC явилось заметным событием не только с точки зрения объема продажи и удельного веса в рыночном обороте. Наиболее существенным фактором оказалась избранная фирмой IBM рыночная стратегия. Именно эту стратегию вынуждены будут принять все другие фирмы — производители микроЭВМ, желающие выжить в конкурентной борьбе.

До появления персональной ЭВМ IBM PC все ведущие фирмы отрасли (такие, как Apple, Commodore и Radio Shack) стремились создать уникальные аппаратные средства, по возможности обеспечивающие выполнение программ, которые не могут быть реализованы на ЭВМ конкурирующих фирм. Более того, эти фирмы решительно противились попыткам конкурентов создавать однотипные машины.

Такова была линия поведения, характерная для старой промышленности мини-ЭВМ и разделяемая слишком многими производителями микроЭВМ. Ее несостоятельность стала очевидной задолго до вмешательства IBM. Причиной этого было появление CP/M, операционной системы, не типичной ни для одной из трех наиболее распространенных персональных ЭВМ (Apple, Commodore и Radio Shack) и, несмотря на это, не только выжившей, но и победившей в конкурентной борьбе. Система CP/M была принята огромным числом второстепенных производителей микроЭВМ. Кроме того, она была поставлена на многих микроЭВМ фирм Radio Shack и Apple, хотя это и было связано с дополнительными расходами на использование программного обеспечения сторонней разработки, а от фирмы Apple потребовало и введения дополнительных аппаратных средств.

Не явилось ли это ответом на пожелания потребителей, стремящихся во что бы то ни стало преодолеть препоны

изготовителей и предлагающих использовать СР/М? Действительно, это было так, и именно поэтому реальные стандарты промышленно выпускаемого программного обеспечения отвечают в настоящее время самым взыскательным требованиям пользователей.

Мнение пользователей не было оставлено без внимания фирмой IBM. Вообще фирма обычно реагирует на любую информацию, если она представляет хотя бы какую-либо ценность; именно поэтому IBM и добилась за время своего существования столь значительных успехов.

Таким образом, когда фирма выпустила свою персональную ЭВМ, она сразу же предложила ввести новый стандарт. Любой производитель средств вычислительной техники мог при желании выпускать изделия, совместимые с персональной ЭВМ фирмы IBM. Всяческую поддержку в принятии стандарта, предлагаемого фирмой IBM, получили фирмы, занимающиеся сбытом программного обеспечения. Первый независимый журнал по микроЭВМ, предназначенный для освещения вопросов применения IBM PC, в течение шести месяцев после выхода первого номера стал одним из наиболее популярных изданий отрасли. Люди покупали журнал и затем заказывали последующие номера в непредсказуемых количествах.

Существует уже несколько моделей микроЭВМ, созданных другими фирмами и совместимых с IBM PC. По-видимому, уже большее число фирм, занимающихся программным обеспечением, отдает предпочтение стандарту фирмы IBM по сравнению со всеми другими типовыми программными средствами, кроме, быть может, операционной системы СР/М.

Я уверен, что в ближайшее время для микроЭВМ будут существовать лишь два стандарта на программное обеспечение: система СР/М, реализуемая при построении 8-разрядных микроЭВМ на основе микропроцессора Z80, и система IBM (MDOS и СР/М 86), реализуемая в 16-разрядных микроЭВМ на основе микропроцессоров 8086 и 8088. Фирмам, не поддерживающим эти стандарты, предстоит тяжелая и изнурительная борьба.

Можно справедливо отметить, что 8-разрядные микропроцессоры уходят в прошлое, а микропроцессоры 8086 и 8088 относятся к числу наименее мощных 16-разрядных микропроцессоров. Однако если эти микропроцессоры справляются с решением возлагаемых на них задач, то пользователь не будет принимать во внимание никакие теоретические рассуждения о их несовершенстве. И если даже реально существующие стандарты не отражают наибольшие достижения в соответствующих областях, то для пользователя это не важно, пока он не ощущает их недостатки.



Разница в производстве микроЭВМ и мини-ЭВМ заключается в том, что микроЭВМ быстро становятся предметами широкого потребления. Проблема сбыта массовому покупателю потребительских товаров для руководителей промышленности микроЭВМ является более сложной, чем проблема, стоявшая перед производителями мини-ЭВМ при общении с относительно небольшим контингентом потребителей, формирование которого в основном происходило в начале 70-х годов.

Многие функционирующие в настоящее время производители микроЭВМ не прислушались к мнению пользователей. Но именно эта информация определяет как ничто другое, кто выживает в процессе неизбежных резких изменений рыночной конъюнктуры.

*А. Осборн,*

президент фирмы  
Osborne Computer, 1983

# ПРЕДИСЛОВИЕ

За время, прошедшее с 50-х годов, цифровая ЭВМ превратилась из «волшебного», но при этом дорогого, уникального и перегретого нагромождения электронных ламп, проводов и магнитных сердечников в небольшую по размерам машину, состоящую из сотен тысяч крошечных полупроводниковых приборов, которые упакованы в небольшие пластмассовые корпуса.

В результате этого превращения ЭВМ стали применяться повсюду. Они управляют работой кассовых аппаратов и продовольственных магазинов, следят за работой автомобильных систем зажигания и ведут учет семейного бюджета. Более того, бурный прогресс полупроводниковой микроэлектроники, представляющей собой базу вычислительной техники, свидетельствует о том, что сегодняшний уровень как самих ЭВМ, так и областей их применения является лишь слабым подобием того, что наступит в ближайшем будущем.

Предлагаемый вниманию читателя справочник по ЭВМ охватывает соответствующую тематику, начиная с технологических основ цифровых ЭВМ. В справочнике последовательно освещаются вопросы аппаратного и программного обеспечения ЭВМ и столь разнообразные темы, как искусственный интеллект, робототехника и распознавание речи. Достаточно подробно рассматриваются микропроцессоры, представляющие собой одно из последних достижений в области аппаратных средств ЭВМ.

ЭВМ начинают затрагивать жизнь каждого человека. Если вы заболели, то, попав в больницу, окажетесь в окружении ЭВМ (в современной больнице вычислительных машин может быть больше, чем больных, так как разработчики медицинской аппаратуры широко используют микропроцессоры). В школах ЭВМ многие годы применялись для ведения учебной документации, а теперь они используются при изучении многих учебных дисциплин, не имеющих прямого отношения к вычислительной технике. Даже в начальной школе ЭВМ внедряются для изучения курсов элементарной математики и физики. Широкое распространение получили игры, построенные на основе микропроцессоров. Микропроцессоры встраиваются в кухонные плиты для приготовления пищи, посудомоечные машины и системы поддержания заданной температуры.

Столь широкое применение вычислительных средств ставит ряд философских вопросов, относящихся к перспективам на будущее. Уже сейчас ЭВМ могут достаточно четко произносить простые словосочетания и предложения, в результате чего банковские ЭВМ, возможно, смогут сообщать кассиру состояние вашего счета в устной форме по телефону, а в некоторых современных автомобилях также в речевой форме водитель получит информацию о состоянии агрегатов автомобиля. ЭВМ способны и воспринимать устную речь, однако им приходится выполнять большую работу по расшифровке услышанного, если форма общения жестко не установлена. Эта область деятельности ЭВМ является объектом интенсивных исследований, и ей также уделено место в данной книге.

Упомянутая здесь проблема распознавания речи является частью более широкой проблемы, называемой распознаванием образов. Если ЭВМ смогут хорошо распознавать образы, они будут способны анализировать рентгенограммы и отпечатки пальцев, а также выполнять многие другие полезные функции (сортировкой почты они занимаются уже сейчас). Следует отметить, что человеческий мозг прекрасно справляется с распознаванием образов даже при наличии нерелевантной информации (шумов), и исследования в этой области, направленные на приближение соответствующих возможностей ЭВМ к способностям человека, представляются весьма перспективными. Если ЭВМ смогут достаточно качественно распознавать речь и отвечать на нее в словесной форме, то, по-видимому, станет возможным вводить в них в этой форме программы и данные. Это позволит в буквальном смысле слова говорить ЭВМ, что она должна делать, и выслушивать ее мнение по этому поводу при условии, конечно, что выдаваемые ей указания четкие, не содержат противоречий и т. д.

Устное общение с ЭВМ позволит упростить ее программирование, однако остается нерешенной проблема, на каком именно языке следует с ней общаться. Многие предлагают для этих целей английский язык, но он не обладает точностью и однозначностью, необходимыми с точки зрения ЭВМ и реализуемых в ней трансляторов. В этой области, разумеется, очень многое предстоит сделать, и устойчивая тенденция перехода от машинно-ориентированных языков ассемблера к современным языкам программирования высокого уровня свидетельствует о необходимости и значимости проводимых в ней работ. Ряд разделов справочника содержит материал, достаточно отражающий состояние и этой области и направления ее дальнейшего развития.

Робототехника также представляет собой перспективную область применения ЭВМ. На промышленных предприятиях используется сейчас множество робототехнических устройств;

неожиданные и удивительные виды роботов начинают заполнять и научно-исследовательские лаборатории. Возможность и целесообразность применения роботов в качестве слуг, официантов, билетных кассиров и в других ролях уже нашли отражение в продукции кино и телевидения; существует множество хирургических и точных производственных операций, которые могут и будут выполняться роботами, управляемыми ЭВМ (так как во многих случаях роботы справляются с этими действиями лучше, чем люди).

Мы часто жалуемся, что другие люди не понимают нас; в настоящее время ЭВМ не понимают нас. В течение некоторого периода нам придется довольствоваться такими машинами, которые просто следуют нашим указаниям. Однако память ЭВМ начинает конкурировать с человеческой памятью. В значительной степени благодаря мастерству разработчиков запоминающих устройств объем памяти больших ЭВМ сравним сейчас с объемом нашей собственной памяти; в то же время своеобразная организация мозга определяет его значительные преимущества с точки зрения творческой деятельности. В эту область начинает проникать искусственный интеллект. Следует отметить, что в некоторых областях, ранее считавшихся доступными лишь для «человеческой» мысли, ЭВМ выступают весьма успешно. Например, в таких стройных математических системах, как евклидова геометрия, ЭВМ функционируют гораздо успешнее, чем можно было ожидать; в процессе недавно проведенных испытаний ЭВМ понадобились считанные минуты, чтобы доказать все теоремы, входящие в программу средней школы.

Я считаю, что, для того чтобы уверенно общаться с ЭВМ, необходимы некоторые знания как аппаратного, так и программного аспектов их организации. Существует тенденция придавать ориентированному на пользователя персональным ЭВМ более «дружеские» качества, чем присущи им на самом деле, чтобы увеличить их сбыт. Тем самым снижается гибкость ЭВМ и им придается элемент загадочности, который должен и может быть устранен за счет хотя бы поверхностного знакомства с реальными принципами действия вычислительных машин. Книга, подобная этой, может быть весьма полезна пользователю в его стремлении избавиться от впечатлений загадочности. В то же время справочник такого рода может способствовать расширению областей применения ЭВМ и служить источником информации по вычислительной технике.

*Томас К. Бартц,*

Гарвардский университет, 1983 г.

# ИСТОРИЯ СОЗДАНИЯ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И ОСНОВНЫЕ ПОНЯТИЯ

*Г. Хеллерман*

## 1.1. ВВЕДЕНИЕ

В данной книге описывается современная цифровая вычислительная система общего назначения, представляющая собой наиболее гибкое и сложное творение человеческого разума. Ее гибкость проявляется в обеспечении возможности решения весьма широкого круга задач, который ограничен лишь способностью человека указать определенные инструкции для решения задачи. Подобные инструкции в форме точной и специфической последовательности операторов, подробно определяющих процедуру решения задачи, задаются **программой**. Вычислительная система предназначена для надежного и быстрого выполнения программ. Скорость современной машины, оцениваемая по скоростям выполнения арифметических операций, таких как сложение, вычитание и сравнение, изменяется в пределах от 100 000 команд в 1 с до 10 000 000 команд в 1 с в зависимости от ее размера и стоимости. Большая современная вычислительная машина может обработать всего лишь за несколько часов больше информации, чем это было сделано человечеством до наступления века электроники в 50-х годах нашего столетия. Неудивительно, что это необыкновенное усиление способности человека обрабатывать информацию ускоряет новую научно-техническую революцию.

Большинство людей, по-видимому, считают термины «вычислительная машина» и «вычислительная система» синонимами и связывают их с физическим оборудованием, как, например, центральным процессором, консолью, лентами, дисками, устройством считывания с перфокарт и печатающими устройствами, которые привлекают внимание при посещении машинного зала. Хотя эти устройства важны, они составляют лишь «верхушку айсберга». На начальном этапе использования современной вычислительной системы мы имеем дело не с самой машиной, а с совокупностями правил, называемых **языками програм-**

---

Adapted from Digital Computer Systems Principles, 2d ed., by Herbert Hellerman, Copyright © 1973, Used by permission of McGraw-Hill, Inc. All rights reserved.



мирования, на которых указываются действия, которые должна выполнять ЭВМ. Важное значение языка программирования подчеркивается тем фактом, что сама вычислительная машина может рассматриваться как аппаратный интерпретатор некоторого конкретного языка, который называется **машинным языком**. Для обеспечения эффективной работы машины разработаны машинные языки, использование которых представляет известные трудности для человека. Большинство пользователей не чувствуют этих неудобств благодаря наличию одного или нескольких языков, созданных для улучшения связи человека с машиной. Гибкость вычислительной машины проявляется в том, что она может исполнять программы-трансляторы (в общем случае они называются **компиляторами** или **интерпретаторами**) для преобразования программ с языков, ориентированных на пользователей, в программы на машинном языке.

Таким образом, вычислительная система состоит не только из вычислительной машины, представляющей собой комплекс оборудования, но и из программ, включая те, которые транслируют программы пользователей с любого из имеющихся языков в программы на машинном языке. Значительная часть этой книги посвящена подробному анализу теории и практики двух важных тем, связанных с вычислительными системами: оборудованию (аппаратуре) и программированию (программному обеспечению).

## 1.2. ИСТОРИЧЕСКИЙ ОБЗОР

Механические средства для счета и вычисления были известны еще в далеком прошлом. Одним из древних средств подобного рода являются счеты, которые сохранились до наших дней в качестве простого практического приспособления для вычислений во многих частях света, в особенности на Востоке. (Разновидность счетов — абак — использовалась, видимо, древними египтянами и была знакома китайцам еще в VI в. до нашей эры.) Для опытного человека счеты могут быть хорошим вспомогательным средством для вычислений. Существуют различные виды счетов; они различаются обозначением, принятым для представления чисел и расположения перемещаемых костяшек или сходных с ними других простых объектов, соответствующих каждому разряду числа. Передвигая костяшки, можно задавать числа, складывать и вычитать их для получения желаемого результата. Операции умножения и деления выполняются с помощью последовательностей операций сложения и вычитания.

Хотя в ранних устройствах большее внимание уделялось необходимости механизации арифметических операций, хранение

промежуточных результатов было также важным. Большинство устройств, подобных счетам, хранили лишь простой текущий результат. В качестве памяти другого типа использовался любой писчий материал, например глиняные дощечки и позднее бумага. Пока скорость операций была небольшой и память использовалась медленно, отсутствовала заинтересованность в поиске путей механизации управления последовательностями операций. Однако предвестники такого управления появились в областях, далеких от вычислительной техники. Например, в ткацком станке Жаккарда образца 1801 г. использовались перфорированные (пробитые) карты для управления командами для переплетения нитей в ткани.

Чарльз Бэббидж (1792—1871 гг.) был, вероятно, первым, кто сформулировал идею создания универсальной вычислительной машины. Следует отметить, что первым побудительным мотивом Бэббиджа в этом направлении была невысокая надежность вычислений, производимых вручную, а не их низкая скорость. В частности, он обнаружил ряд ошибок в некоторых астрономических таблицах. При установлении причин ошибок Бэббидж пришел к убеждению, что безошибочные таблицы могут создаваться только машиной, которая получает описание необходимых вычислений, выданное человеком, а затем производит вычисления и выдает на печать их результаты без вмешательства человека. Бэббидж детально разработал проект аналитической машины, которая должна была стать первой универсальной вычислительной машиной. Из-за отсутствия достаточной финансовой поддержки он не смог завершить ее реализацию.

По мере развития промышленности возрастала необходимость в механизации вычислений. После переписи населения в США в 1890 г. стало ясно, что без создания новых процессов обработки данных обработка результатов одной переписи может не завершиться до начала следующей переписи. Доктор Герман Холлерит применил перфорированные карты и простые машины для обработки данных переписи в 1890 г. С тех пор машины с перфорированными картами получили широкое распространение в деловой и административной сферах.

Первая треть двадцатого века ознаменовалась последовательным развитием и внедрением многих вычислительных устройств. Весьма значительный вклад в эту область внес математик Алан Тьюринг, который в 1937 г. опубликовал работу с ясным и глубоким описанием универсальной схемы вычислений. Его результаты были представлены в терминах гипотетической «машины» с удивительно простой структурой, которая, как это было им отмечено, обладала всеми необходимыми признаками универсальной вычислительной машины. Хотя машина Тьюринга была лишь теоретическим построением и никогда серьезно

не рассматривалась как экономически приемлемая машина (она работала бы недопустимо медленно), она привлекла внимание ряда талантливых исследователей к вопросу о возможности создания универсальной вычислительной машины.

Вторая мировая война дала серьезный толчок к усовершенствованию вычислительных устройств и технологии их производства. В 1944 г. Говард Айкен и группа исследователей из IBM построили электрическую вычислительную машину Harvard Mark I на релейных логических элементах. В 1946 г. Дж.П. Эккерт и Дж.В. Мочли разработали электронную вычислительную машину (ЭВМ) ENIAC на электронных лампах. Обе эти ЭВМ предназначались для выполнения научных расчетов. Появление технологии ЭВМ первого поколения связывается с массовым производством машины UNIVAC I в 1951 г. Термин «первое поколение» ассоциируется с использованием электронных ламп в качестве основных компонент логической схемы, которая включает также широкий набор запоминающих устройств, таких, как ртутные линии задержки, запоминающие электронно-лучевые трубки, магнитные барабаны и магнитные сердечники и т. д.

Второе поколение оборудования характеризуется заменой электронной лампы как основной компоненты на транзисторы (изобретенные в 1948 г.). По сравнению с электронной лампой полупроводниковый транзистор является более эффективным отчасти потому, что не требует энергии для нагрева источника электронов. Транзистор по сравнению с электронной лампой имеет практически неограниченное время жизни и надежность и для его производства требуются существенно меньшие затраты. Оборудование второго поколения, которое появилось в 1960 г., относится к периоду широкого внедрения и использования ЭВМ общего назначения. Третье и четвертое поколения технологии ЭВМ (относящиеся примерно к 1964 и 1970 гг.) отличаются возрастающим применением методов интегрального изготовления с целью производства большей части ЭВМ в едином автоматическом непрерывном процессе без использования ручного труда.

Успехи в разработке оборудования сопровождались достижениями в программировании, которые трудно изложить в сжатой форме. Первой важной разработкой, обычно приписываемой Г. Хопперу, является символический машинный язык, который освобождает программиста от многих чрезвычайно утомительных действий, порождающих ошибки. Другой вехой оказался первый язык высокого уровня Фортран, появившийся приблизительно в 1955 г. и получивший широкое распространение. Этот язык включает многие элементы алгебраических обозначений, такие, как индексированные переменные и математические

выражения произвольной длины. Поскольку Фортран разработан фирмой IBM, выпускавшей наибольшее число машин, он быстро получил широкое распространение и, претерпев ряд модификаций, даже в настоящее время остается широко используемым языком.

Для удовлетворения потребностей решения различных классов задач с помощью ЭВМ были предложены другие языки. Среди них самыми важными являются Кобол, предназначенный для обработки экономических данных, Алгол, первый язык, получивший признание во всем мире, и особенно среди математиков и ученых, ПЛ/1, разработанный фирмой IBM и введенный в 1965 г. в качестве единого языка, способного удовлетворить требованиям решения научных и экономических задач, а также задач системного программирования.

Наряду с введением и усовершенствованием машинных языков соответствующее развитие получила технология программирования, т. е. методы построения компиляторов и интерпретирующих трансляторов, а также других вспомогательных средств для программистов. Крайне важная идея создания **операционной системы**, привлекавшая многих разработчиков, состояла в построении набора программ, предназначенных для управления и распределения всех ресурсов системы в ответ на запросы пользователя в соответствии с определенными критериями эффективности. Примерно с 1966 г. почти все средние и большие вычислительные машины работали под управлением операционных систем. Обычно задания предоставлялись пользователями в виде колод перфорированных карт в машинный зал или с помощью терминалов **дистанционного ввода заданий** (ДВЗ), т. е. устройств считывания карт и печатающих устройств, связанных телефонными линиями с ЭВМ. В любом случае после получения задания операционная система принимает почти все решения по планированию выполнения работ. Большая ЭВМ могла выполнить несколько сотен или даже тысяч заданий за 24-часовой рабочий день под управлением лишь одного или двух профессиональных операторов, находящихся в машинном зале.

В 60-е годы был значительно ускорен симбиоз ЭВМ с телефонной системой (**телеобработка**). В большинстве случаев он включал использование дистанционного ввода задания и программы специального назначения, как, например, в системах резервирования авиабилетов. Существенный успех был достигнут также в вопросах повышения общности и согласования работы ЭВМ общего назначения с отдельными пользователями посредством систем с **разделением времени**. Соответствующая операционная система чередует запросы нескольких пользователей, которые могут находиться на значительных расстояниях.

Пользователи соединяются с системой по телефонным линиям, используя такие устройства, как телетайп или терминал с печатающим устройством. Поскольку скорость ЭВМ существенно превышает скорость «обдумывания» человека, одна система может вполне нормально обслуживать от 50 до 100 (или еще большее число) пользователей, каждый из которых считает себя обладателем собственной ЭВМ. Система с разделением времени, приближающая людей к ЭВМ, по-видимому, обладает очень важными потенциальными возможностями усиления творческих способностей человека.

### 1.3. КЛАССИФИКАЦИЯ АВТОМАТИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

В широком смысле автоматические вычислительные машины могут быть классифицированы как аналоговые и цифровые (рис. 1.1). В аналоговых вычислительных машинах используется аналогия между значениями, принимаемыми некоторой физической величиной, такой, как скорость вращения вала, расстояние или электрическое напряжение, и переменной в рассматриваемой задаче. В принципе цифровые вычислительные

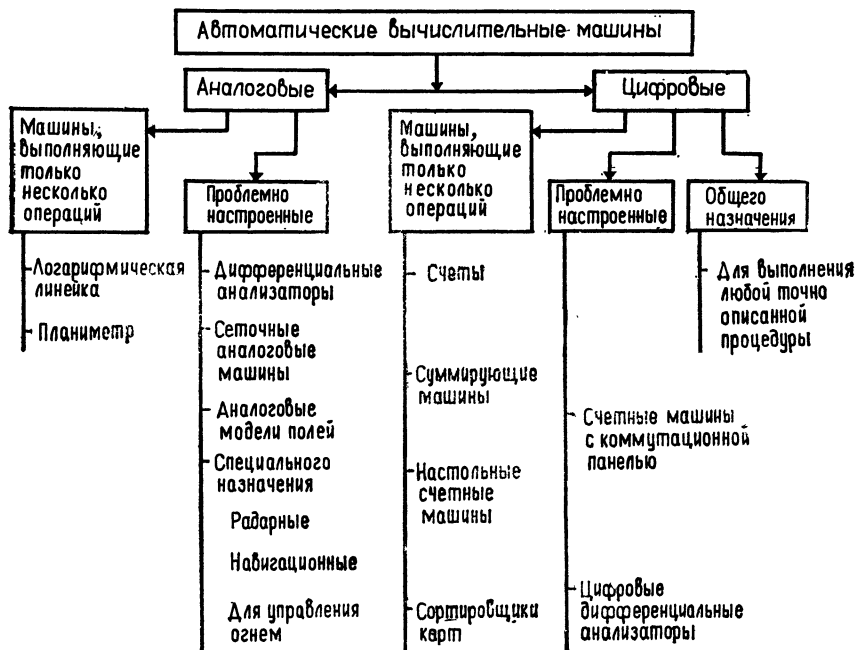


Рис. 1.1. Классификация вычислительных машин.



машины непосредственно оперируют с числами. Все вычислительные машины в некотором смысле обладают аналоговым свойством, так как необходимо использовать физическое представление для такой абстракции, каким является число. В цифровой вычислительной машине эта аналогия минимальна, тогда как в аналоговой вычислительной машине она применяется весьма широко.

Классы как аналоговых, так и цифровых вычислительных машин содержат подкласс довольно простых машин для механического выполнения лишь определенных простых операций. Например, логарифмическая линейка является аналоговой вычислительной машиной, в которой числа представляются посредством длин отрезков на логарифмической шкале. Умножение, деление, нахождение корней чисел и другие операции выполняются путем сложения и вычитания длин отрезков. Среди примеров цифровых машин, предназначенных для выполнения одной или нескольких операций, можно отметить суммирующие устройства и настольные счетные машины.

К другому классу машин относятся более сложные машины по сравнению с машинами, выполняющими лишь несколько операций. Эти машины можно назвать **проблемно настроенными**. Помимо арифметических операций в них предусмотрена запись процедуры соединения операций в последовательность, необходимую для решения задачи. Спецификация процедуры может быть встроена в управляющее устройство машины, как это имеет место в некоторых специализированных машинах, или же набираться на коммутационной панели для задания нужной последовательности операций. Главная идея заключается в том, что процедура решения задачи вводится посредством определенной операции, и после этого весь ход решения задачи выполняется автоматически.

Электронный дифференциальный анализатор, появившийся в конце 40-х годов, представляет собой аналоговую вычислительную машину наиболее общего вида. Он строится из нескольких тщательно спроектированных прецизионных схем (интеграторов, суммирующих усилителей, прецизионных потенциометров и конденсаторов), каждая из которых способна выполнять одну операцию. Обычно задача вводится в машину с помощью коммутационной панели. Поскольку, как правило, средства для внутреннего хранения результатов отсутствуют, выходные данные пересылаются прямо в графопостроитель. Точность, которая ограничивается дрейфом и шумами, обычно не превышает 1/1000 полной шкалы. По сравнению с вычислительными машинами общего назначения аналоговые вычислительные машины имеют такие недостатки, как неспособность решать задачи общего типа, низкая точность результатов, трудность выпол-

нения сложных операций (включая умножение и деление с высокой скоростью), отсутствие возможности эффективного хранения больших объемов информации, непосредственная связь количества используемого оборудования с размерностью задачи. Однако аналоговая вычислительная машина может обеспечить высокую скорость решения тех задач, для которых она предназначена, в частности математических или имитационных задач, включающих дифференциальные уравнения, и в случае необходимости с меньшими затратами, чем цифровая вычислительная машина. Высокая скорость аналоговой вычислительной машины является результатом значительного параллелизма в ее работе, т. е. компоненты машины работают одновременно над отдельными частями общей задачи.

Наиболее важный теоретический вопрос для проблемно настроенной машины состоит в выделении класса задач, которые можно решать на этой машине. С практической точки зрения вопрос редко ставится в этой форме, так как машины с коммутационными панелями обычно проектируются для конкретных типов задач. Тем не менее вопрос о максимальных логических возможностях машины, т. е. о классе задач, допускающих решения, является фундаментальным. В 1937 г. Тьюринг внес в него ясность, введя понятие весьма простой гипотетической «машины» (с тех пор называемой универсальной машиной Тьюринга) и доказав, что действительно любая процедура решения задач может быть представлена в виде процедуры для этой машины. Отсюда следует, что любая машина, которая может моделировать универсальную машину Тьюринга, также обладает присущим ей свойством общности. Такие машины называются **универсальными (общего назначения)**. Большинство ЭВМ для практических целей являются машинами общего назначения. Они отличаются друг от друга по стоимости, скорости, надежности, емкости памяти и степени сложности связей с остальными устройствами или людьми, но не по предельным логическим возможностям.

## 1.4. СТРУКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Вычислительную систему лучше всего следует рассматривать как набор ресурсов, доступных ее пользователям посредством программ, написанных в соответствии с правилами системных языков программирования. Эти ресурсы относятся к двум основным классам, обладающим широким набором компонент.

### 1. Оборудование (аппаратные средства)

а. **Запоминающие устройства.** Предназначены для хранения как программ, так и данных.

**б. Обрабатывающая логика.** Предназначена для осуществления арифметической и логической обработки информации.

**в. Управляющая логика.** Предназначена для пересылки информации и упорядочения событий.

**г. Преобразователи.** Устройства для преобразования информации из одной физической формы в другую, например печатающее устройство, которое преобразует электрические сигналы в печатные символы на бумаге.

## **2. Программы (программное обеспечение)**

**а. Прикладные программы.** Программы, разработанные для применения пользователями ЭВМ, например научные программы, программы для расчета платежных ведомостей и инвентаризации, фактически определяют основную часть работы вычислительных машин.

**б. Системные программы.** Программы, относящиеся к средствам, при помощи которых система обеспечивает пользователей определенными удобствами и управляет своими ресурсами, например трансляторы языков и программы операционных систем.

## **1.5. ПРИНЦИПЫ ОРГАНИЗАЦИИ АППАРАТУРЫ**

Теперь воспользуемся термином «вычислительная машина» для обозначения только аппаратной части вычислительной системы общего назначения.

Все вычислительные машины обладают некоторыми общими характеристиками. Изложение этих характеристик будет неформальным — наша цель состоит в таком их описании, чтобы можно было интуитивно понять сущность строения машины и основы ее универсальности.

С точки зрения пользователя машина оперирует двумя основными типами информации: 1) операндами или данными, 2) командами, каждая из которых определяет одну арифметическую или управляющую операцию (например, СЛОЖЕНИЕ, ВЫЧИТАНИЕ), и одним или несколькими операндами, являющимися объектами этой операции.

В машине как команды, так и данные представляются в виде целых чисел в двоичной системе счисления или в какой-либо форме двоичного кода. Выбор этого представления обусловлен тем, что для него «атомом» информации служит характеризующий двумя состояниями (обозначаемыми как 0 и 1) сигнал, для которого требуются лишь простейшие и наиболее надежно работающие электронные устройства. Хотя при машинной обработке информации требуется двоичное представление команд и данных, большинство пользователей могут применять обычные десятичные числа и буквенные представления имен операций и

данных. Программы-трансляторы (обычно предоставляемые фирмами-изготовителями вычислительных машин), которые выполняют машину, преобразуют эти удобные представления в двоичную форму, используемую непосредственно в ЭВМ. Другими словами, двоичное представление информации в машине является важным для машинных вычислений, но оно не определяет главный принцип работы вычислительной машины общего назначения.

Приведем список общих характеристик вычислительных машин общего назначения:

1. Машина способна хранить большое количество информации (как данные, так и команды). Как правило, в целях экономии имеется по крайней мере три уровня скорости и емкости памяти. Емкость памяти является основным фактором, ограничивающим класс решаемых задач.

2. Обычно набор команд небольшой (от 16 до 256 типов команд), но их разумный выбор удовлетворяет требованиям построения любой процедуры.

3. Ссылки на операнды производятся по их именам, которые обрабатываются при помощи команд.

4. Команды выбираются из памяти и выполняются автоматически. Обычно местоположение в памяти следующей команды указывается счетчиком команд (или счетчиком программ). Показание счетчика команд в большинстве случаев изменяется скачками (увеличивается на 1) для определения местоположения следующей команды, но некоторые команды изменяют показание счетчика команд так, чтобы он указывал значение, зависящее от результатов сравнений заданных операндов. Благодаря этому программа может ветвиться на взаимно исключающие друг друга части, т. е. на альтернативные последовательности команд.

На рис. 1.2 показана общая структура типичной вычислительной машины. Ядром этой системы служит центральный процессор (ЦП), включающий главное запоминающее устройство, которое хранит как программу, так и данные, и арифметико-логическое устройство (АЛУ). АЛУ содержит такие схемы, как сумматор, сдвиговый регистр и несколько быстродействующих **регистров** для хранения операндов и команды, выполняемой в текущий момент. Счетчик команд также должен быть включен в АЛУ, хотя в некоторых структурных схемах средства управления программами выделены отдельно. Одну из частей центрального процессора составляет набор схем трассировки, которые обеспечивают пути между запоминающим устройством и АЛУ и контроллерами ввода-вывода или каналами. В системе указанного типа к одному каналу может быть присоединено много запоминающих устройств или устройств ввода-

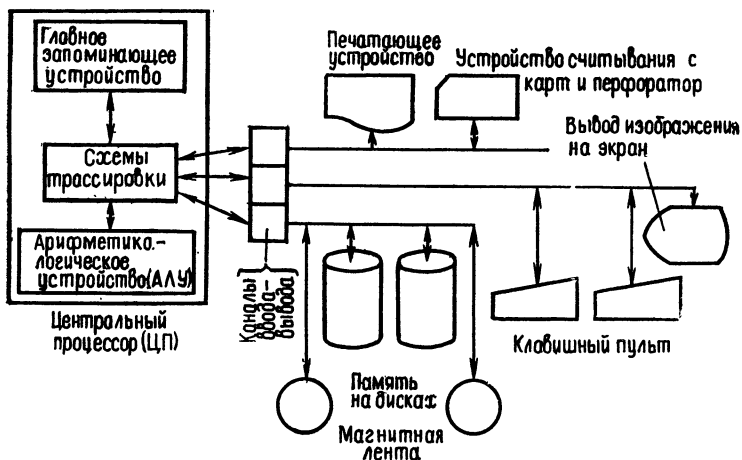


Рис. 1.2. Общая структура типичной цифровой вычислительной машины.

вывода, однако в любой момент времени каждый канал может функционировать только с одним устройством, осуществляя обмен информацией с главным запоминающим устройством. Конечно, это накладывает ограничение на число устройств, которые могут работать одновременно. Однако это оправдано экономией разделения общих путей к главному запоминающему устройству и простотой управления перемещением информации между различными устройствами и запоминающим устройством.

Опишем элементы вычислительной машины, необходимые для выполнения основных операций:

**1. Запоминание.** Средства для хранения довольно значительного объема информации и простой экономичный механизм доступа для выбора пути прохождения элемента информации в запоминающее устройство из одного узла (регистра) или из запоминающего устройства в другой узел. Обычно даже в одной и той же системе имеются различные варианты запоминающих устройств, отличающиеся друг от друга по времени обращения, емкости и стоимости.

**2. Поток данных.** Переключательные схемы, обеспечивающие пути движения информации из одной части вычислительной машины в другую.

**3. Преобразование.** Схемы для арифметической и других видов обработки данных. Эта функция обычно выполняется в одном арифметико-логическом устройстве. Такая централизация экономически себя оправдывает, поскольку один и тот же набор быстродействующих дорогих схем используется последовательно во времени для выполнения всех операций. Преобра-



зующие схемы обрабатывают информацию, полученную из запоминающего устройства, управляя переключением потока данных. В дальнейшем мы увидим, что многие из более сложных преобразований, таких, как вычитание, умножение и деление, могут осуществляться экономично путем управления последовательностями очень простых операций, например сложения, сдвига и т. д.

**4. Управление.** Под управлением подразумевается важная функция определения в потоке данных пути между узлами и осуществления передачи информации этим путем. Функция управления проявляется в ЭВМ на различных уровнях. Обычно управление организуется в виде группы временных последовательностей или **циклов**. Каждый период цикла обычно (но не всегда) разбивается на равноотстоящие друг от друга временные промежутки, которые называются **интервалами синхронизации (тактами)**. Термин «цикл» относится к конкретному типу последовательности интервалов синхронизации, на которых осуществляется выбор пути в схемах потока данных. Например, **имеется цикл считывания**, в течение которого команда, содержащая информацию о преобразовании, передается из запоминающего устройства в регистр АЛУ. На каждом интервале синхронизации цикла выполняется какая-либо элементарная операция, например передача адреса команды в механизм доступа запоминающего устройства, подача сигнала для доступа к запоминающему устройству или передача полученной команды в регистр АЛУ.

**5. Ввод-вывод.** Поскольку информация в процессоре и запоминающем устройстве вычислительной машины представляется в виде электрических сигналов, необходимы устройства для преобразования информации от вида, генерируемого человеком, к виду, считываемому машиной при ее вводе, и для обратного преобразования при ее выводе. Наиболее часто для выполнения этих преобразований используется перфокарта. Оператор считывает информацию с рукописных или машинописных документов и вводит ее с клавиатуры клавишного перфоратора, имеющей большое сходство с клавиатурой печатающей машинки. Перфоратор преобразует символы клавишей в набор отверстий на карте (рис. 1.3). Затем карты подаются в устройство считывания с карт, которое содержит аппаратуру для чтения карт, т. е. для восприятия позиций отверстий и преобразования их во внутреннее представление — электрические сигналы. Информация хранится на перфокарте в неизменном виде и может считываться людьми (просмотром конфигураций отверстий, либо чтением отпечатанных символов в верхней части карты). Перфорационное устройство может управляться ЭВМ для вывода результатов обработки в виде набора перфокарт.

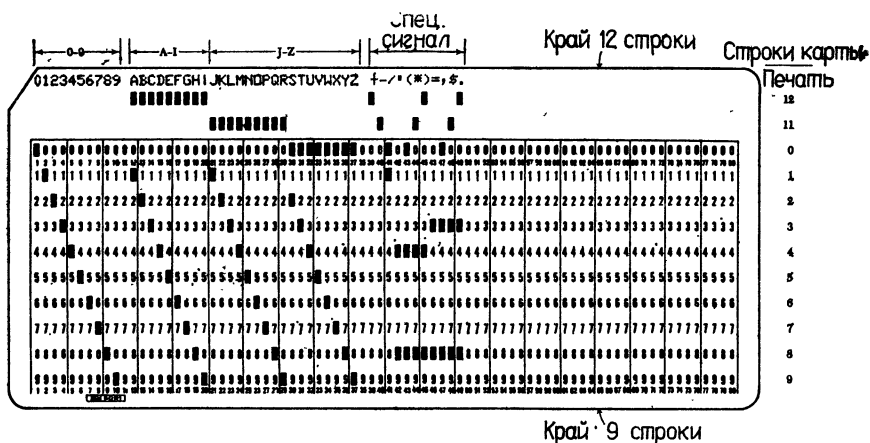


Рис. 1.3. Перфокарта фирмы IBM с 80 колонками, иллюстрирующая нумерацию по строкам и колонкам, с перфорированными отверстиями для группы из 48 символов, используемых в языке Фортран. *Представление данных:* каждая из 80 колонок может содержать одно или несколько отверстий (маленькие темные прямоугольники) для представления буквенно-цифровых знаков. На данной карте пробиты отверстия для представления 10 десятичных цифр, 26 букв латинского алфавита и 12 специальных символов (включая пустой символ) из общепринятого набора символов.

Перфокарты и устройства для их пробивки и считывания служат примерами устройств ввода-вывода. Другими устройствами ввода-вывода являются печатающие машинки, перфорированная бумажная лента, телетайпы, дисплеи на электронно-лучевых трубках, аналого-цифровые преобразователи.

Трудно установить четкую границу между функциями хранения информации и ее ввода-вывода (перфокарта служит для обеих целей). Однако это можно сделать, используя в качестве критерия возможность считывания с устройства вывода информации непосредственно ЭВМ. На этой основе печатающие устройства, телетайпы, дисплеи на электронно-лучевых трубках являются устройствами ввода-вывода, а перфокарты, перфолента и магнитная лента — запоминающими устройствами.

Согласно общепринятой классификации, все устройства и машины, не являющиеся частями ЦП и запоминающего устройства, непосредственно доступного ему, относятся к устройствам ввода-вывода.

## 1.6. ТРЕБОВАНИЯ К ЗАПОМИНАЮЩИМ УСТРОЙСТВАМ

К запоминающим устройствам почти всегда предъявляются определенные требования. Они не зависят от физической природы запоминающего устройства — от того используются ли в нем

магнитные ленты, магнитные диски, полупроводники и т. д. Для устройства памяти основными являются две следующие операции:

**1. СЧИТЫВАНИЕ** (копирование). Содержимое некоторой заданной части памяти копируется и пересылается в определенное стандартное место. Отметим, что операция копирования для пользователя является **неразрушающей**, т. е. информация в памяти не изменяется при ее считывании.

**2. ЗАПИСЬ** (замещение). Производится **замещение** текущего содержимого заданной части памяти **на** содержимое некоторого стандартного места.

Иногда технология запоминающего устройства имеет естественную тенденцию к нарушению этих требований. В таком случае для их удовлетворения разрабатываются дополнительные схемы.

## 1.7. ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ

Предположим, что память состоит из массива ячеек, которые можно представить в виде длинного ряда ящиков (отделений) письменного стола. Каждая ячейка содержит информацию, называемую **операндом**, который может быть ассоциирован с числом, записанным на листе бумаги в ячейке. **Каждой ячейке присваивается имя** — на информацию в этой ячейке можно сослаться только по имени ячейки, а не по ее содержанию. Операнд, на который ссылаются, затем используется при вычислениях. Обычно оборудование ЭВМ имеет аппаратно реализованную схему имен, посредством которой имена ячеек обозначаются целыми числами 0, 1, 2, ... Однако пользователь может выбирать другие имена для ячеек, такие, как **X Y, Z, I**. Преобразование имен, выбранных пользователем, в машинные имена является простым рутинным процессом, поскольку каждое из них присваивается одному машинному имени. Это оправдывает использование мнемонических символов в качестве имен операндов.

Если не оговорено, числа будут обозначать операнды, буквы — имена операндов. Например,

$$X \leftarrow 5 \quad (1)$$

читается как «5 определяет X»; это означает, что операнд или обычное число 5 замещает содержимое ячейки с именем X. Еще одним примером служит оператор

$$Y \leftarrow 1 + X, \quad (2)$$

означающий, что «содержимое (операнд) ячейки с именем X увеличивается на 1 и результат замещает содержимое ячейки с

именем  $Y$ ». Для краткости этот оператор будем интерпретировать как « $X$  плюс 1 определяет  $Y$ ». Важно отметить, что данный оператор хотя и изменяет содержимое  $Y$ , содержимое  $X$  остается прежним. Простая программа состоит из последовательности операторов, сходных с теми, которые были приведены выше. Хотя подробные правила записи операторов (допустимых символов, требуемых знаков препинания и т. д.) сильно варьируются от одного языка программирования к другому, многие принципы программирования можно проиллюстрировать с помощью одного языка (в данном случае языка APL).

Так как в обычных условиях ЭВМ обрабатывает большие объемы информации, обозначение и обработка **массивов** информации являются ключевыми понятиями. Одномерный массив ячеек назовем **вектором**. Примером вектора служит

$$X \equiv (3, 29, 47.4, 82, -977.6). \quad (3)$$

**Элемент**, или **компонента**, **вектора** будет обозначаться наименованием с двумя частями. Одна часть представляет собой имя всего вектора; другая часть, заключенная в скобки, указывает **позицию** элемента, на который ссылаются. В вышеприведенном примере

$$X[2] \equiv 29 \quad (4)$$

(предполагается, что позиции чисел в  $X$  нумеруются слева направо, начиная с 1).

Отметим также смысл индекса переменной. Например,

$$Y \leftarrow X[I] \quad (5)$$

означает «содержимое ячейки  $I$  используется для указания позиции числа  $X$ , а содержимое ячейки с таким наименованием в  $X$  замещает содержимое  $Y$ ».

Например, если  $X$  — вектор, заданный равенством (3), то последовательность  $I \leftarrow 3, Y \leftarrow X[I]$  приводит к результату  $Y$ , повторно замещаемому на число 47.4. Переменная вида  $X[I]$  или  $X[3]$  называется **индексированной**, переменная  $I$  — **индексной переменной**, или **индексом**. Операции над индексами чрезвычайно важны, поскольку они позволяют систематически вычислять имена ячеек, исходя из имен других ячеек или констант.

Почему столь важно умение вычислять имена? Одна из причин этого заключается в том, что в противном случае пришлось бы явно определять каждую ячейку единственным именем. Использование многих тысяч имен было бы громоздко и пришлось бы разработать процедуру систематического присвоения, совпадающую или сходную с той, в которой используется сама индексированная переменная. Вторая причина, отражающая преимущества использования индексированных переменных, состоит в

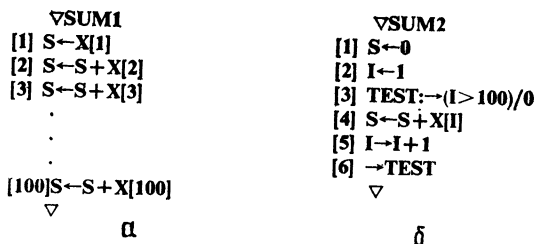


Рис. 1.4. Линейно упорядоченное и циклическое программирование суммы 100 чисел X.

α — линейно-упорядоченная программа; б — циклическая программа.

возможности включения вычислений имен в программу для обработки данных, что значительно сокращает запись программы, но увеличивает время ее выполнения. Например, предположим, что в память введены 100 чисел и обозначены как вектор X. На рис. 1.4 приведены две программы для выполнения одного и того же задания: вычислить сумму этих чисел.

Рис. 1.4, а легко истолковать: здесь представлена линейно-упорядоченная программа, содержащая все 100 шагов, записанных в явной форме. Рис. 1.4, б соответствует значительно более короткой программе, поскольку она содержит **цикл**. Отметим, что на рис. 1.4, б «ядром» программы служит оператор 4, который суммирует элемент X в позиции I с S для получения нового значения S. Этот оператор выполняется многократно, каждый раз с новым значением I. Оператор 5 увеличивает значение I на 1, а оператор 6 направляет программу к оператору 3, так как в шестой строке содержится оператор с меткой TEST. Третья строка соответствует утверждению: «сравнить I с 100, если оно больше 100, перейти к строке 0, которая по соглашению означает выход из программы. В противном случае продолжить вычисление индекса, перейдя к следующему оператору (строка 4)». Ясно, что по этим правилам в данном случае каждая из строк 4, 5, 6 будет выполняться 100 раз, а строка 3 будет выполняться 101 раз; другими словами, строки 3—6 образуют цикл программы. Если сравнить линейно-упорядоченную программу и программу с циклом на рис. 1.4, то можно заметить, что в первом случае число записанных операторов равно 100, во втором случае — 6. Указанное преимущество программ с краткой записью в определенной степени компенсируется тем фактом, что программа с циклом требует выполнения 403 операторов по сравнению со 100 выполняемыми операторами для линейно-упорядоченной программы. В программе с циклом для обновления индексов и проверок требуются дополнительно выполняемые операторы.

## 1.8. СООТНОШЕНИЕ «ПРОСТРАНСТВО — ВРЕМЯ»

Разработчик ЭВМ или ее пользователь должен учитывать ряд фундаментальных аспектов организации вычислительной машины и решаемой задачи для достижения компромисса между факторами пространства и времени. Термин «пространство» будет приблизительно соответствовать «количеству оборудования».

Рассмотрим пример этой идеи компромисса при выборе компонент машины. На рис. 1.5 показаны два способа получения одной и той же функции, состоящей в формировании шести сигналов. Каждый из этих сигналов может быть или сигналом включения ( $= 1$ ) или выключения ( $= 0$ ). Чтобы гарантировать появление выходного сигнала только в синхронизирующих интервалах, каждый сигнал и синхронизирующий импульс подаются в схему И, выходной сигнал которой равен 1 только тогда, когда на выходах линии сигнала и линии синхронизирующих импульсов появляются 1. В остальные моменты времени этот выходной сигнал равен 0.

На рис. 1.5, а показано одно из представлений набора шести сигналов. Здесь каждому сигналу отводится отдельная линия. Выходные сигналы появляются на шести выходных линиях (и требуются шесть схем И). На рис. 1.5, б иллюстрируется вторая возможность — шесть сигналов циркулируют в виде им-

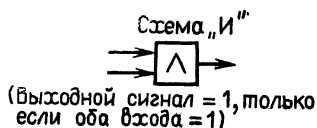
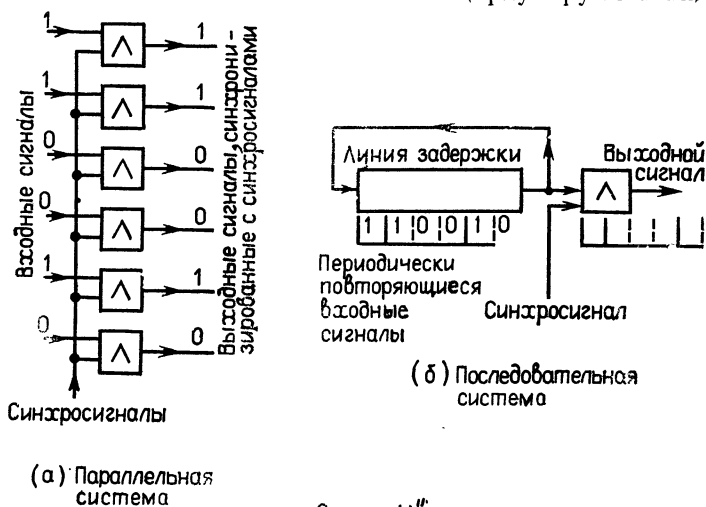


Рис. 1.5. Параллельное и последовательное представления сигналов включения и выключения.

пульсов в линии задержки — в данном случае задержка равна шести периодам сигналов синхронизации. Здесь сигналы появляются **последовательно во времени** на одной выходной линии.

Первая схема содержит больше аппаратуры и стоит дороже, но она предпочтительнее по быстродействию. Вторая схема в противоположность первой схеме содержит меньше аппаратуры, стоит дешевле, но менее быстродействующая. Отметим также, что по мере роста числа сигналов размер параллельной схемы увеличивается пропорционально, однако время для приема всех сигналов остается прежним. С другой стороны, для обработки большего числа сигналов в последовательной схеме нет необходимости в дополнительных линиях (или схемах И), хотя время задержки должно увеличиваться пропорционально.

Многие желательные свойства ЭВМ, особенно надежность, обусловлены использованием простых компонент. Сложные структуры и схемы выполнения операции строятся из большого числа простых компонент и сложных временных последовательностей сигналов, которые они генерируют или изменяют.

Вследствие большого разнообразия устройств, предназначенных для обработки, управления и, в частности, для хранения данных, много усилий было затрачено на создание экономических структур. Соотношение между временем обработки и объемом аппаратуры в рассмотренном выше примере указывает на один метод уменьшения стоимости вычислений за счет увеличения времени обработки. Этот пример поясняет идею **разделения во времени**, т. е. использования одного и того же оборудования (например, сумматора) последовательно во времени посредством пересылки в него чисел, суммируемых последовательно во времени. Поэтому выбор пути и осуществление пересылки информации в вычислительной машине из одного места в другое являются основополагающей операцией. Пути, выделяемые для пересылки данных, определяют структуру **потока данных**, наиболее важную характеристику любой вычислительной машины.

Компромисс между временным и пространственным факторами можно также пояснить на примере организации программирования. Напомним, что процедуру сложения набора чисел можно программировать как линейно-упорядоченную, получая тем самым программу, требующую больше памяти, но выполняемую быстрее. Альтернативный способ программирования заключается в построении программы с циклом. Это обеспечивает экономию памяти, но увеличивает время выполнения программы. Во многих случаях программа с циклом дает выигрыш в отношении объема используемой аппаратуры, более существенный по сравнению со снижением скорости ее выполнения. Этот способ программирования всегда предпочтительнее, за исключением простейших случаев, когда число слагаемых мало.

При решении любой задачи, как правило, можно выбрать несколько решений, которые в первом приближении можно сравнить друг с другом с учетом пространственного и временного факторов.

Отметим некоторые общие свойства вычислительных систем, вытекающие из краткой вводной части этой главы. Прежде всего ЭВМ общего назначения является устройством, которое может воспринимать точное описание процедуры — **программу** — для решения любой задачи (решаемой за конечное число шагов) и затем автоматически выполнять программу для обработки поступающих в него данных.

Алгоритм (или программа) является важным не только для пользователей ЭВМ, но и для ее разработчиков по следующим двум причинам: 1) разработчики могут добиться успеха, если только они понимают, как именно используется их изделие, т. е. как для них составляют программы; 2) последовательности внутренних переключательных операций, необходимых для реализации арифметических и других операций, также являются алгоритмами, которые должны быть указаны и реализованы разработчиком логических схем.

Современная ЭВМ подобна большому пианино, на котором пользователь может сыграть, например, произведения Бетховена или воспроизвести какие-либо сочетания звуков. Получение наибольшей отдачи от средств, затраченных на оборудование, и трудозатрат — это задача оптимизации ресурсов, которая обладает некоторыми особенностями задач комбинаторной математики: «незначительное» изменение исходных условий или критерия оптимизации может привести к весьма существенному сдвигу в производительности. Благодаря универсальности ЭВМ не вызывает сомнения возможность получения «ответов» на правильно поставленную задачу тем или иным способом. Как правило, главным вопросом является выяснение возможности получения ответов таким образом, чтобы предоставить наибольшие удобства пользователю, оптимизировать время решения задачи, емкость памяти, надежность или некоторую комбинацию подобных параметров. Безусловно, все эти факторы взаимосвязаны и улучшение части из них может быть достигнуто только за счет ухудшения других. Это обстоятельство было уже проиллюстрировано в этой главе на примерах, отражающих компромисс между пространственным и временным факторами. Ряд достаточно общих, но пока нераскрытых законов «сохранения» может связать указанные параметры, однако в настоящее время общие взаимосвязи параметров могут рассматриваться лишь качественно, хотя в отдельных случаях возможен и количественный анализ.



# 2

## СТРУКТУРЫ ЭВМ

*В. Гамахер, З. Вранежич, С. Заки*

### 2.1. ВВЕДЕНИЕ

Целью настоящей главы является введение некоторых понятий и связанных с ними терминов. Будет дан только общий обзор основных характеристик ЭВМ; более подробно (и точно) эти вопросы излагаются в последующих главах. В простейшей форме ЭВМ представляет собой машину, которая принимает входную информацию, обрабатывает ее в соответствии с программой, хранящейся в ее памяти, и создает выходную информацию как результат обработки.

### 2.2. ФУНКЦИОНАЛЬНЫЕ УСТРОЙСТВА

Под **ЭВМ** подразумевается большое число разнообразных машин, значительно различающихся по размерам, скорости и цене. Часто используют более специализированные термины для некоторых типов ЭВМ. Малые ЭВМ, обычно называются **мини-ЭВМ**, что отражает их относительно низкие цену, размер и вычислительную производительность. В начале 70-х годов появился термин **«микроЭВМ»** для особо малых ЭВМ, наиболее дешевых и содержащих всего несколько больших интегральных схем (БИС).

Большие ЭВМ значительно отличаются от мини-ЭВМ и микроЭВМ по размерам, производительности, цене, а также степени сложности структуры. Однако основные понятия по существу одинаковы для всех классов ЭВМ, так как они связаны с несколькими хорошо определенными идеями, которые мы попытаемся разъяснить. Таким образом, предлагаемый ниже материал применим к большинству ЭВМ общего назначения.

ЭВМ состоит из пяти основных функционально независимых устройств: ввода, памяти, арифметико-логического, вывода и

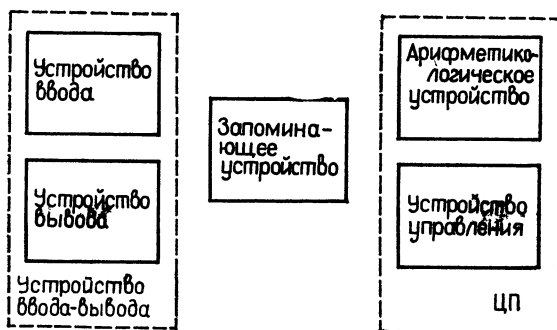


Рис. 2.1. Основные функциональные устройства ЭВМ

управления (рис. 2.1). Устройство ввода принимает кодированную информацию из внешней среды: либо от человека-оператора, либо от электромеханических приборов. Эта информация запоминается в памяти для более позднего использования или же сразу обрабатывается арифметическими и логическими схемами. Шаги обработки определены программой, хранящейся в памяти. Наконец, результаты обработки возвращаются во внешнюю среду через устройство вывода. Все эти действия координируются устройством управления. На рис. 2.1 не показаны связи между различными функциональными устройствами, которые, конечно, должны быть.

Обычно арифметико-логические схемы скомпонованы совместно с основными схемами управления в виде **центрального процессора** (ЦП). Оборудования ввода и вывода объединяются термином **устройство ввода-вывода**. Это обусловлено тем, что некоторая стандартная аппаратура обеспечивает как функции ввода, так и вывода. Простейшим примером является часто используемое телетайпное устройство. Следует подчеркнуть, что функции ввода и вывода здесь разделены. ЭВМ воспринимает это устройство как два отдельных устройства, а оператор воспринимает их как части единого устройства.

В больших ЭВМ основные функциональные устройства могут состоять из отдельных частей, часто обладающих значительными размерами. На рис. 2.2 представлена фотография такой ЭВМ. Мини-ЭВМ имеют значительно меньшие размеры. Основная часть мини-ЭВМ часто уместается на письменном столе, что иллюстрируется двумя ЭВМ, показанными на рис. 2.3. Даже такая сложная система из мини-ЭВМ, как, например, показанная на рис. 2.4, имеет меньшие размеры, чем большие ЭВМ.

В ЭВМ поступает информация двух типов: в виде **команд** и **данных**. Команды, или инструкции

- управляют передачей информации внутри ЭВМ, а также между ЭВМ и устройствами ввода-вывода;

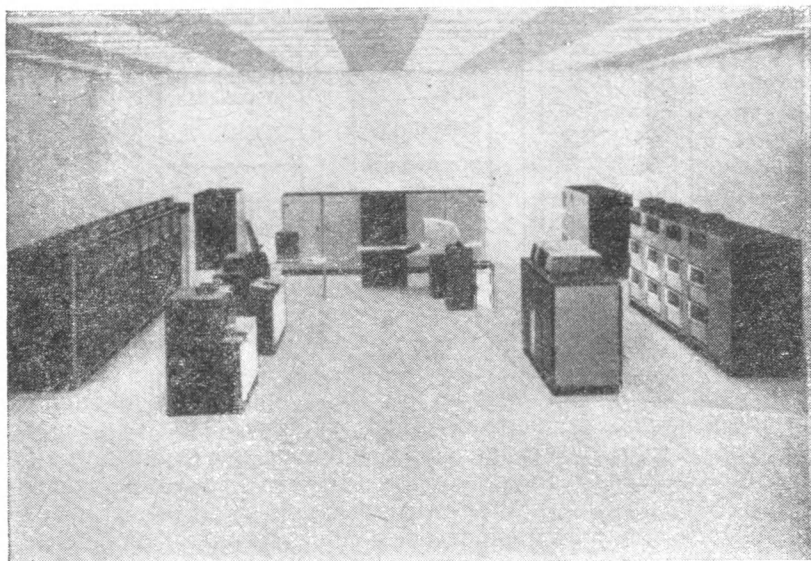


Рис. 2.2. Типичная большая ЭВМ—IBM S 370/158 (Фирма IBM).

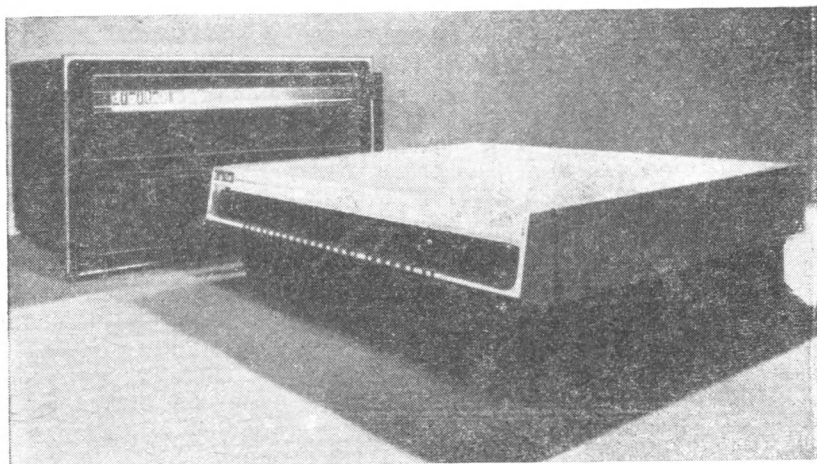


Рис. 2.3. Две мини-ЭВМ—PDP/8M и PDP11/05 (фирма Digital Equipment Corp.).

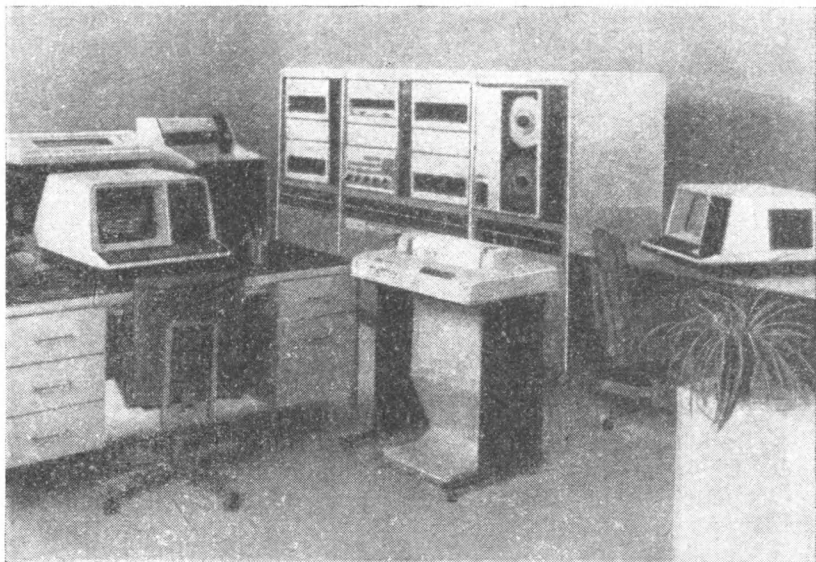


Рис. 2.4. Система мини-ЭВМ (фирма Digital Equipment Corp.).

- указывают подлежащие выполнению арифметические и логические операции.

Последовательность команд, которые обеспечивают выполнение задачи, называется **программой**. Обычно решение задачи начинается с загрузки программы (или нескольких программ) в память. Затем центральный процессор извлекает команды программы из памяти и выполняет предусмотренные операции. Обычно команды выполняются в той последовательности, в которой они хранятся в памяти, однако возможны отклонения от этого порядка, например когда требуется ветвление. Таким образом, ЭВМ находится под полным управлением **хранимой программы**, кроме случаев, когда поступают сигналы прерывания от оператора или цифровых приборов, связанных с ЭВМ.

Данные — числа и кодированные символы, которые используются как операнды команд. Это не следует воспринимать как строгое определение, так как термин «данные» часто используется для любой цифровой информации. Даже по приведенному выше определению данных вполне допустимо, что сама программа (т. е. последовательность команд) может рассматриваться как данные, если она обрабатывается другой программой. Таким примером является задача **компиляции** (перевода) исходной программы с языка высокого уровня на язык

машинных команд и данных. Исходная программа представляет входные данные для программы компилятора. Компилятор переводит исходную программу в программу на машинном языке.

Информация, которая обрабатывается на ЭВМ, должна быть представлена в соответствующем формате. Так как аппаратура большинства современных ЭВМ (электронное и электро-механическое оборудование) состоит из электронных цифровых схем, которые имеют только два устойчивых состояния — включено и выключено, используется двоичное кодирование информации. Таким образом, каждое число, символ текста или команда кодируются как цепочка двоичных цифр (**бит**), причем каждая цифра принимает одно из двух возможных значений. Числа обычно представлены в двоичной позиционной системе счисления. Иногда используется **двоично-десятичное кодирование**, когда каждая цифра кодируется четырьмя битами.

Буквенно-цифровые символы также представляются в виде двоичных кодов. Разработано несколько схем кодирования. Двумя наиболее широко распространенными схемами являются ASCII (Американский стандартный код обмена информацией), где каждый символ представлен 7 бит<sup>1)</sup>, и EBCDIC (расширенный двоично-десятичный код обмена), где для представления символа используется 8 бит<sup>2)</sup>.

### 2.3. УСТРОЙСТВА ВВОДА

ЭВМ принимают кодированную информацию при помощи устройств ввода. Эти устройства имеют приборы, способные «читать» такие данные. Простейшим из них является электрическое **печатающее** устройство, связанное электронными схемами с обрабатывающей частью ЭВМ. Печатающее устройство связано с ЭВМ таким образом, что при нажатии любой клавиши на клавиатуре соответствующая буква или цифра автоматически переводятся в соответствующий код, который непосредственно посылается в память ЭВМ или в центральный процессор.

Другим устройством ввода является **телетайп**, такой, например, как ASR 33. Кроме того, что телетайп является печатающим устройством, он еще содержит перфоленточную станцию, предназначенную для считывания и перфорации. Благодаря низкой цене и универсальности телетайп является одним из наиболее часто используемых устройств ввода и вывода.

Хотя телетайп и печатающее устройство являются несомненно самыми простыми устройствами ввода-вывода, они обла-

---

<sup>1)</sup> В СССР аналогичным свойством обладает стандарт КОИ-код обмена и обработки информации. — *Прим. перев.*

<sup>2)</sup> Советский аналог — ДКОИ — двоичный код обмена и обработки информации. — *Прим. перев.*

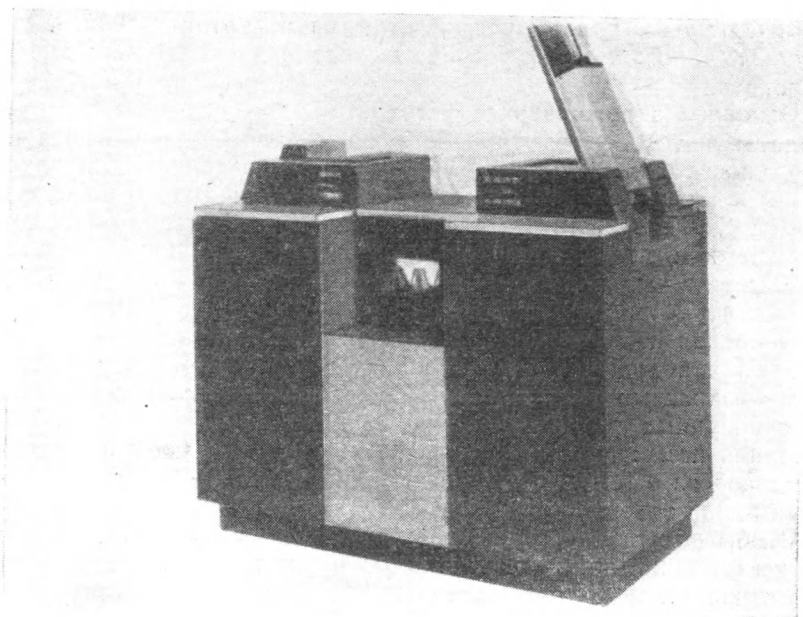


Рис. 2.5. Устройство для чтения с перфокарт (фирма IBM).

дают невысокой скоростью ввода данных и оказываются весьма неудобными для использования, когда работа связана с большим количеством данных. Это делает необходимым разработку быстродействующих устройств ввода-вывода, таких, как **высокоскоростные устройства ввода** с перфоленты, а также ввода с перфокарт. Для получения копий программ или данных удобно использовать перфокарты, разделенные на колонки (обычно 80), каждая из которых соответствует одному символу. Устройство ввода определяет положение отверстий на перфокарте и, таким образом, читает входную информацию. Скорость типичных устройств ввода с перфокарт достигает 1000 карт в 1 мин. На рис. 2.5 показана фотография устройства ввода с перфокарт.

Имеется много других типов устройств ввода. Особенно следует отметить устройства **графического ввода**, в которых используется дисплей с электронно-лучевой трубкой.

#### 2.4. УСТРОЙСТВО ПАМЯТИ

Единственной функцией устройства памяти является хранение программ и данных. С другой стороны, эта функция может быть выполнена различным оборудованием. Обычно

различают два класса устройств памяти, которые включают основные запоминающие устройства и вторичные запоминающие устройства.

**Основное запоминающее устройство**, или **основная память**, представляет собой быстродействующее запоминающее устройство, способное работать со скоростью, характерной для электроники; программы и данные в нем хранятся только во время работы ЭВМ. Типичными компонентами такого запоминающего устройства являются магнитные сердечники или полупроводниковые схемы. Раньше ЭВМ использовали память на **магнитных сердечниках**, теперь на **полупроводниках**.

Основная память содержит большое число ячеек, каждая из которых хранит 1 бит информации. Эти ячейки редко управляются независимо друг от друга. Чаще они связаны в группы фиксированных размеров. Такие группы называются **словами**. Основная память организована таким образом, чтобы содержимое слова из  $n$  битов можно было запомнить или найти при помощи одной основной операции.

Чтобы облегчить доступ к любому слову основной памяти, с каждым положением слова связано некоторое явное имя. Эти имена являются числами, которые указывают соответствующее положение и поэтому называются **адресами**. Каждое слово становится доступным посредством указания его адреса, который выдает команда управления в начале процесса запоминания или поиска (считывания).

**Длиной слова** ЭВМ называют количество битов в каждом слове. Большие ЭВМ обычно имеют 32 и более бит в слове, мини-ЭВМ — от 12 до 24 (самое распространенное 16), некоторые микроЭВМ — всего 4 или 8 бит в слове. Емкость основной памяти — один из главных факторов, характеризующих размер ЭВМ. Малые ЭВМ могут иметь только несколько тысяч слов (4096 — типичный минимум), тогда как большие ЭВМ — миллионы слов. Данные обычно преобразуются в ЭВМ в отдельные слова, группы или подгруппы слов. Типичное обращение к основной памяти приводит к считыванию одного слова из памяти или записи его в память.

Как упоминалось выше, программы и данные должны находиться в основной памяти во время работы. Команды и данные могут быть записаны в память или считаны из нее под управлением процессора. Существенно, чтобы обращение к любому слову основной памяти могло производиться максимально быстро. Память, в которой возможен доступ к любому слову по его адресу, называется **памятью прямого доступа** (ППД). Время, необходимое для получения доступа к одному слову памяти, называется **временным циклом памяти**. Для большинства современных ЭВМ это время обычно составляет от 300 нс до 1 мкс.

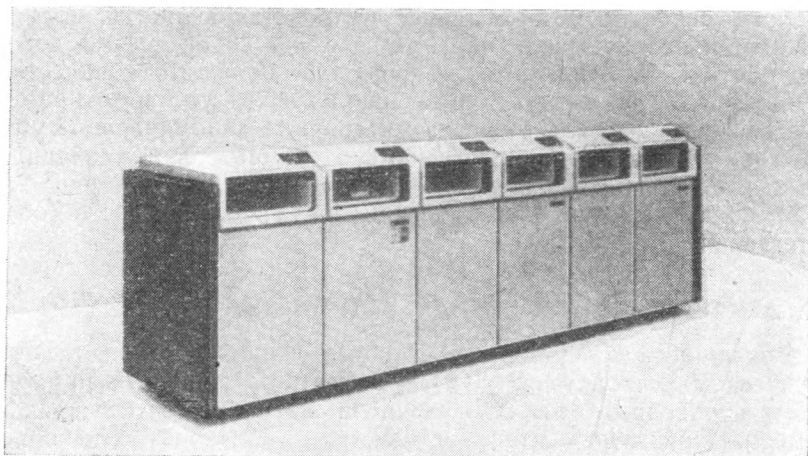


Рис. 2.6. Запоминающее устройство на магнитных дисках (фирма IBM).

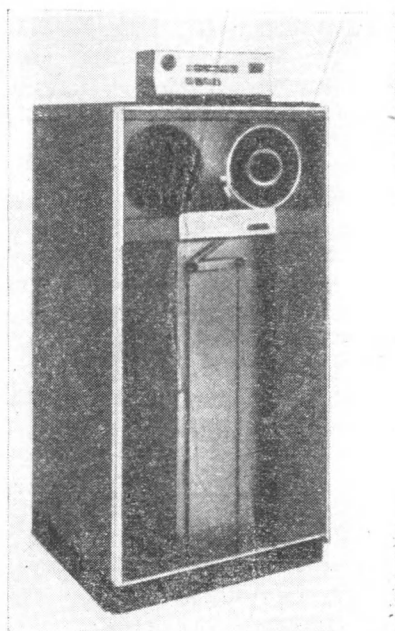


Рис. 2.7. Блок магнитной ленты (фирма IBM).



Хотя основное запоминающее устройство является весьма важным и необходимым, оно представляет собой весьма дорогую часть ЭВМ. Поэтому для хранения большого количества данных, особенно когда данные используются не часто, применяются дополнительные дешевые **вторичные запоминающие устройства**. Имеется широкий выбор вторичных запоминающих устройств — **магнитные диски, барабаны и ленты**. На рис. 2.6 и 2.7 показаны группы дисковых и ленточных устройств соответственно.

## 2.5. АРИФМЕТИКО-ЛОГИЧЕСКИЕ УСТРОЙСТВА

Большинство операций в ЭВМ выполняются в арифметико-логическом устройстве (АЛУ). Рассмотрим типичный пример. Пусть необходимо сложить два числа, находящихся в основной памяти. Они передаются в АЛУ, где происходит сложение. Сумма может быть затем записана в память.

Аналогично любая другая арифметическая или логическая операция (например, умножение, деление, сравнение чисел) осуществляется пересылкой требуемых операндов в АЛУ, где выполняются эти операции. Следует отметить, что не всегда операнды извлекаются из основной памяти, ибо центральный процессор обычно содержит одну или более ячеек высокоскоростной памяти, называемых **регистрами**, которые могут быть использованы для временного хранения часто используемых операндов. Каждый такой регистр может хранить одно слово данных. Время доступа к регистру обычно в 5—10 раз меньше времени доступа к основной памяти.

Устройство управления и арифметическое устройство используются во много раз меньше времени в основном цикле ЭВМ, чем другие ее устройства. Поэтому можно создать относительно сложную систему, состоящую из ряда внешних устройств, управляемых единым центральным процессором. Такими устройствами могут быть телетайп, дисковая и ленточная память, датчики, дисплеи, механические контроллеры и т. д. Разумеется, это возможно только благодаря значительному различию в скорости устройств, позволяющему ЦП, обладающему высокой скоростью, управлять работой более медленных устройств.

## 2.6. УСТРОЙСТВА ВЫВОДА

Устройства вывода в некотором смысле противоположны устройствам ввода. Их функцией является возврат результатов обработки во внешнюю среду.

Некоторые устройства выполняют как функцию ввода, так и функцию вывода. К таким устройствам относятся печатающие

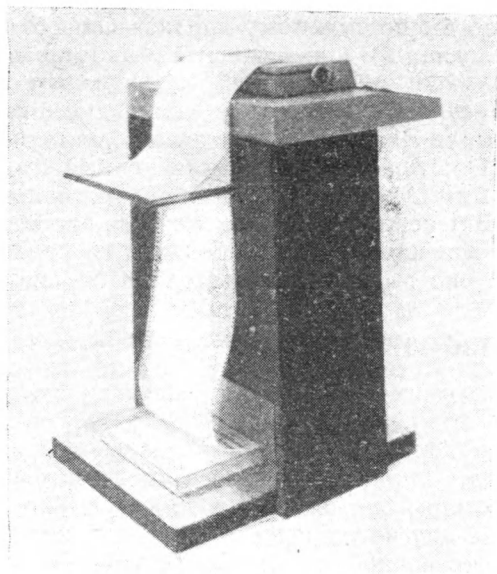


Рис. 2.8. Телетайп (фирма IBM).

устройства, телетайпы и графические дисплеи. Такая двойная роль некоторых устройств является причиной того, что устройства ввода и вывода объединяются одним термином **устройства ввода-вывода**. На рис. 2.8 приведена фотография типичного телетайпа.

Разумеется, есть устройства, которые выполняют только функцию вывода. Наиболее известным примером является высокоскоростное **печатающее устройство**. Можно создать печатающее устройство, способное работать со скоростью 10 000 строк в минуту. Для механических систем это огромная скорость, но она все же меньше скорости электронного процессора.

Иногда необходимо вывести данные в форме, удобной для последующего ввода. Такой вывод осуществляется на перфокарты при помощи перфоратора. Аналогично может быть сделан вывод на **перфоленту** при помощи ленточного перфоратора.

Наконец, следует заметить, что некоторые запоминающие устройства большого объема используются главным образом для вторичного хранения, но могут также применяться как устройства ввода-вывода. Характерным примером является магнитная лента. Предположим, что некоторое задание включает сбор данных от ряда терминальных устройств, что происходит достаточно долго. Вероятно, такую задачу удобнее и экономичнее решать на мини-ЭВМ. Использование большой ЭВМ для

этой цели было бы по-видимому дороже. Однако предположим, что собранные данные должны быть обработаны сложным образом, недоступным для мини-ЭВМ. Разумным в этой ситуации является осуществление сбора и записи данных на магнитную ленту в мини-ЭВМ. Подготовленная лента переносится на большую ЭВМ, которая может ввести данные с ленты и осуществить обработку данных. Таким образом, большая (и дорогостоящая) ЭВМ используется только там, где она необходима, и стоимость всего задания будет меньше по сравнению со случаем, если бы оно выполнялось на одной большой ЭВМ.

## 2.7. УСТРОЙСТВО УПРАВЛЕНИЯ

Описанные выше устройства обеспечивают необходимые средства для хранения и обработки информации. Действия этих устройств должны быть согласованы, что является задачей устройства управления. Это фактически нервный центр всей ЭВМ, который используется для посылки управляющих сигналов во все другие устройства.

Устройство строчной печати напечатает строку символов, если только оно соответствующим образом управляется. Обычно печать является результатом соответствующей команды **печать**, выполняемой центральным процессором. Обработка этой команды включает посылку **распределенных во времени сигналов** в устройство печати и получение сигналов из него, что является функцией устройства управления.

Обычно считается, что процессы передачи данных (ввода-вывода) управляются командами программного обеспечения, которые идентифицируют устройства, участвующие в передаче данных, и тип передачи. Однако фактически распределенные во времени сигналы, которые управляют передачами данных, порождаются управляющими схемами. Передачи данных между центральным процессором и памятью также управляются устройством управления.

Устройство управления целесообразно рассматривать как физически выделенное центральное устройство машины, некоторым образом взаимодействующее с остальными ее частями. Однако на практике это бывает редко. Большинство управляющих схем распределены по всей ЭВМ. Они связаны большим количеством управляющих линий (проводов), которые передают сигналы для временного тактирования и синхронизации событий во всех устройствах.

Важной частью устройства управления является **индикаторная панель**, или **консоль**, с переключателями и световыми индикаторами, которые позволяют оператору наблюдать, что происходит внутри ЭВМ. Обычно панель применяется, когда на-

рушается вычислительный процесс, что бывает довольно часто. В таких ситуациях оператор может использовать панель для выявления причин этих нарушений и их устранения. Разумеется, некоторые ошибки не могут быть легко исправлены (например, вызванные неисправностями электронных компонент), но многие обычно встречающиеся нарушения правильного течения вычислительного процесса могут быть диагностированы и исправлены оператором.

Итак, работа типичной ЭВМ общего назначения может быть описана следующим образом:

- Через устройства ввода принимается информация (программа и данные) и пересылается в память.
- Информация, хранящаяся в памяти, по требованию программ извлекается и направляется в арифметико-логическое устройство для обработки.
- Обработанная информация выводится из ЭВМ через устройства вывода.
- Все действия в ЭВМ происходят под управлением устройства управления.

## 2.8. ОСНОВНЫЕ ОПЕРАЦИОННЫЕ ПОНЯТИЯ

В предыдущем разделе было отмечено, что поведение ЭВМ управляется посредством команд. Для выполнения заданной задачи соответствующая программа, состоящая из последовательности команд, должна храниться в основной памяти. Команды пересылаются из памяти в центральный процессор (ЦП), который выполняет соответствующие операции. Кроме команд необходимо использовать некоторые данные как операнды, которые также хранятся в памяти. Типичная команда

Add LOCA, R0

прибавляет операнд, расположенный в ячейке памяти LOCA, к операнду в регистре ЦП, который называется R0. Эта команда выполняется за несколько шагов. Сначала команда должна быть передана из основной памяти в ЦП. Затем считывается операнд из ячейки памяти LOCA. Этот операнд суммируется с содержимым R0. Наконец, сумма запоминается в регистре R0.

Передачи данных между основной памятью и ЦП начинаются с передачи адреса ячейки памяти, чтобы получить доступ к устройству памяти, и выдачи соответствующих управляющих сигналов. Затем данные пересылаются в память или из нее.

На рис. 2.9 показано, как может быть осуществлена связь между основной памятью и ЦП. На этом рисунке видны также некоторые детали ЦП, которые пока еще не рассматривались, но являются существенными для выполнения операций. Схема

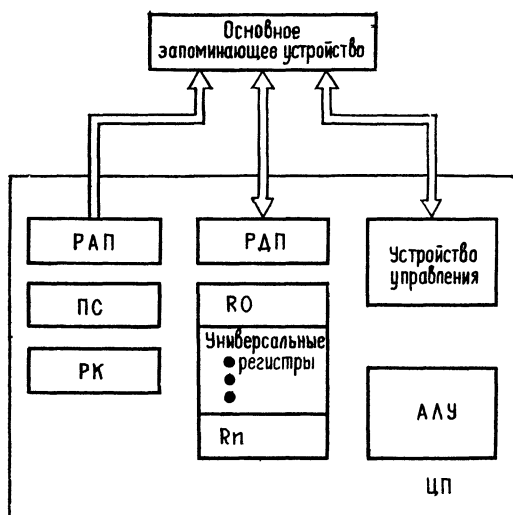


Рис. 2.9. Связи между основной памятью и ЦП.

взаимодействия этих компонент показана не точно, так как мы будем рассматривать только основные характеристики.

ЦП содержит арифметико-логическую схему как основной процессорный элемент и включает также ряд регистров для временного хранения данных. Два регистра представляют особый интерес. **Регистр команд (РК)** содержит команду, которая должна выполняться. Выходы РК связаны с управляющими схемами, которые генерируют распределенные во времени сигналы, необходимые для выполнения команд. **Программный счетчик (ПС)**, или счетчик команд, является регистром, который хранит след выполняемой программы. Он содержит адрес ячейки памяти, в которой находится выполняемая команда. Во время выполнения текущей команды содержимое ПС обновляется на адрес следующей команды, которая должна выполняться. Обычно считается, что ПС указывает команду, которая должна извлекаться из памяти.

Помимо РК и ПС существует по крайней мере еще один, а обычно даже несколько **регистров общего назначения**.

Наконец, имеются два регистра, которые обеспечивают связь с основной памятью — **регистр адреса памяти (РАП)** и **регистр данных памяти (РДП)**. Как подразумевается в названии, РАП используется для хранения адреса ячейки памяти, в которую или из которой должны пересылаться данные. РДП содержит данные, которые должны быть записаны или считаны из адресуемой ячейки памяти.

Рассмотрим теперь несколько типичных шагов выполнения операций. Программы находятся в основной памяти и обычно попадают туда через устройство ввода. Выполнение программы начинается с того, что в ПС засылается адрес первой команды программы. Содержимое ПС пересылается в РАП, и в память посылается сигнал управления считыванием. Через некоторое время (соответствующее времени доступа к основной памяти) адресуемое слово (в данном случае первая команда программы) извлекается из памяти и загружается в РДП. Затем содержимое РДП пересылается в РК; на этой стадии команда готова для декодирования и выполнения.

Если команда содержит операцию, которая должна быть выполнена арифметико-логическим устройством, то необходимо получить требуемые операнды. Если операнд находится в памяти (а он может быть также в универсальном регистре ЦП), его необходимо выбрать из памяти. Для этого в РАП пересылается адрес операнда, и начинается цикл чтения. Операнд, выбранный из памяти в РДП, может быть передан в АЛУ. Выбрав таким образом один или несколько операндов, арифметико-логическое устройство может выполнить требуемую операцию. Если результат этой операции необходимо запомнить в памяти, он должен быть послан в РДП. Адрес ячейки, в которую необходимо поместить результат, пересылается в РАП, и начинается цикл записи. Между тем содержимое ПС увеличивается, указывая следующую команду, которая должна выполняться. Таким образом, как только завершится выполнение текущей команды, может начаться выборка следующей команды.

Помимо передачи данных между основной памятью и ЦП необходимо обеспечить прием данных из устройства ввода и пересылку данных в устройства вывода. Таким образом, должны быть обеспечены машинные команды, управляющие передачами данных (вводом-выводом).

Обычный порядок выполнения программ иногда может изменяться. Довольно часто некоторое устройство требует срочного обслуживания. Например, ведущее устройство в производственном процессе, управляемом ЭВМ, может обнаружить опасную ситуацию. Чтобы разобраться с такой ситуацией достаточно быстро, необходимо прервать нормальный ход программы, которая выполняется в ЦП. Для этого устройство должно выдать **сигнал прерывания**. Прерывание является требованием на обслуживание, которое осуществляется ЦП, выполняющим соответствующую **программу обработки прерывания**. Так как прерывание и его обработка могут изменить внутреннее состояние ЦП, необходимо, чтобы оно сохранялось в основной памяти перед работой программы обработки прерывания. Сохранение состояния достигается пересылкой содержимого РК, универсальных

регистров и некоторой управляющей информации в основную память. После завершения программы обработки прерывания состояние ЦП восстанавливается, так что выполнение прерванной программы может быть продолжено.

## 2.9. СТРУКТУРЫ ШИН

До сих пор рассматривались основные характеристики отдельных частей, составляющих ЭВМ. Чтобы образовать операционную систему, эти части должны быть связаны некоторым образом в структуру. Существует много способов организации структур, мы рассмотрим три наиболее известные структуры.

Если ЭВМ должна обладать достаточной скоростью выполнения операций, она должна быть организована **параллельно**. При такой организации все устройства в данный момент могут работать с полным словом данных, и передачи данных между устройствами происходят параллельно. Это означает, что для организации необходимых связей требуется значительное число проводов (линий). Набор таких проводов, которые имеют ряд общих свойств, называется **шиной**. Помимо проводов, которые передают данные, обязательно должно быть несколько линий, предназначенных для передачи управляющих сигналов. Таким образом, шина содержит как линии данных, так и линии управления.

На рис. 2.10 представлена простейшая форма двухшинной структуры ЭВМ. ЦП взаимодействует с памятью через **шину памяти**. Функции ввода и вывода выполняются при помощи **шины ввода-вывода**, так что данные передаются через ЦП в память. В такой структуре передачи данных ввода-вывода обычно производятся под прямым управлением ЦП, который начинает передачу и управляет ей вплоть до завершения. Обычно для описания операций такого типа используется термин **программируемый ввод-вывод**.

Несколько иной вариант двухшинной структуры показан на рис. 2.11. ЦП и память поменялись местами. Однако по-прежнему имеется шина памяти, связывающая их. Следовательно, передачи данных ввода-вывода производятся непосредственно без участия центрального процессора. Так как память почти не обладает схемно реализованной способностью управления такими передачами, необходимо использовать другой механизм управления. Стандартный метод должен предусматривать **каналы** ввода-вывода, представляющие часть оборудования ввода-вывода, которое обладает необходимой способностью управления передачами. Фактически каналы подобны небольшим процессорам и могут быть выполнены в виде отдельных ЭВМ. Типичная процедура обмена заключается в том, что ЦП инициирует

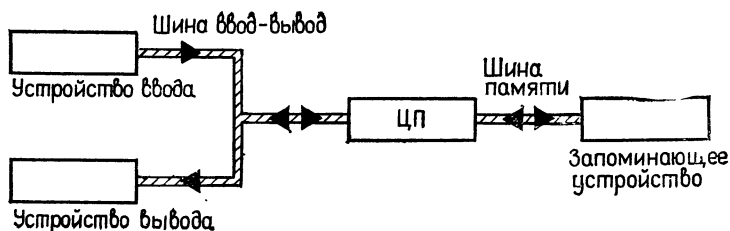


Рис. 2.10. Двухшинная структура ЭВМ.

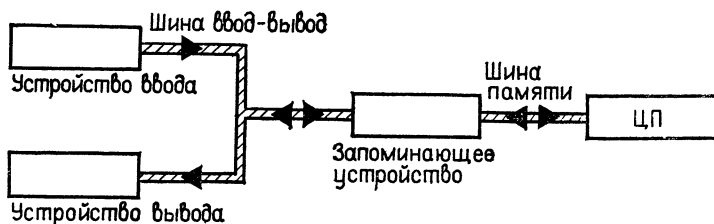


Рис. 2.11. Другой вариант двухшинной структуры ЭВМ.

передачу, посылая требуемую информацию в канал ввода-вывода, который принимает ее и производит управление передачей данных.

Уже упоминалось, что шина содержит ряд линий, которые служат различным целям. Шина памяти включает **шину данных** и **адресную шину**. Шина данных используется для передачи данных. Следовательно, число линий в шине соответствует числу битов в слове. Чтобы обеспечить доступ к данным в памяти, необходимо послать адрес, который укажет на ячейки, в которых хранятся эти данные. ЦП посылает биты адреса в память по адресной шине.

Вышеприведенное описание соответствует большинству ЭВМ. Большие ЭВМ обычно имеют структуру, представленную на рис. 2.11. Многие ЭВМ имеют несколько отдельных шин; это так называемые **многошинные ЭВМ**. Однако их работу можно описать на примере двухшинной ЭВМ, так как основная причина введения дополнительных шин заключается в увеличении скорости обмена.

Существенно иная структура с **общей шиной** показана на рис. 2.12. Все устройства связаны с этой шиной, так что она обеспечивает единство взаимосвязи. Так как шина может быть использована в данный момент только для одной передачи, лишь два устройства могут активно работать в этот момент. Вероятно, такая шина должна содержать шину данных, адресную



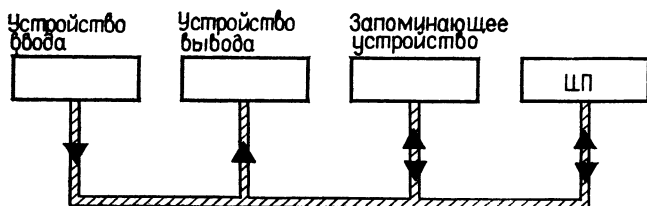


Рис. 2.12. Структура с общей шиной.

шину и несколько управляющих линий. Основной особенностью структуры с общей шиной является ее низкая стоимость и легкость подключения периферийных устройств. Для обменов характерна низкая скорость. Неудивительно, что одношинная структура прежде всего используется в малых ЭВМ — мини-ЭВМ и микроЭВМ.

Различия в структуре шин влияют на характеристики ЭВМ. Тем не менее эти различия не существенны (по крайней мере на этом уровне изложения) при функциональном описании. Действительно, основные принципы работы ЭВМ не зависят существенно от конкретной структуры шин.

Скорость передачи информации по шине редко может быть сравнима со скоростью устройства, связанного с шиной. Некоторые электромеханические устройства относительно медленные, например телетайп, устройство ввода с перфокарт, печатающее устройство. Устройства на магнитных дисках и лентах относительно быстродействующие. Основная память и ЦП работают со скоростью, характерной для электроники; это самые быстродействующие части ЭВМ. Так как все эти устройства должны быть связаны друг с другом через шину, необходимо обеспечить такой механизм передачи, эффективность которого не ограничивалась бы медленными устройствами.

Как правило, для обеспечения такого механизма передачи данных используются **буферные регистры** в устройствах для хранения информации во время передачи. Для иллюстрации рассмотрим передачу кодированного символа из ЦП в телетайп для печати. ЦП осуществляет передачу, посылая символ по шине в выходной буфер телетайпа. Так как буфером является электронный регистр, эта передача требует относительно мало времени. Когда буфер загружен, телетайп начинает печатать без вмешательства со стороны ЦП. В это время шина больше не нужна телетайпу и она освобождается для работы с другими устройствами. Телетайп производит печать символов из своего буфера и запрещает пересылки в буфер, пока он не освободится.

# 3

## СИСТЕМЫ СЧИСЛЕНИЯ И КОДЫ

З. Кохави

### 3.1. СИСТЕМЫ СЧИСЛЕНИЯ

Природа электронных устройств накладывает ограничения на применение десятичной системы счисления в машинных вычислениях, хотя эта система более привычна. В большинстве современных цифровых машин для представления чисел и выполнения арифметических действий используется система счисления, называемая двоичной. Данная глава посвящена вопросам представления чисел в различных системах счисления и способам перехода от одной системы к другой.

#### Представление чисел

Обычное десятичное число представляет собой многочлен по степеням числа 10. Например, число 123.45 есть многочлен

$$123.45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Этот метод представления десятичных чисел соответствует **десятичной системе** счисления, а число 10 называется ее **базой** (или **основанием**). В системе с основанием  $b$  положительное число  $N$  представляет собой многочлен

$$N = a_{q-1}b^{q-1} + \dots + a_0b^0 + \dots + a_{-p}b^{-p} = \sum_{i=-p}^{q-1} a_i b^i,$$

где  $b$  — целое число больше 1,  $a_i$  — целые числа в диапазоне  $0 \leq a_i \leq b-1$ . Последовательность  $a_{q-1}, a_{q-2} \dots a_0$  образует **целую часть**  $N$ , а последовательность  $a_{-1}, a_{-2} \dots a_{-p}$  составляет **дробную часть**  $N$ . Таким образом,  $p$  и  $q$  обозначают количество цифр в дробной и целой частях соответственно. Целая и дробная части разделяются точкой. Цифры  $a_{-p}$  и  $a_{q-1}$  называются **младшим** и **старшим** разрядами соответственно.

---

Adapted from Switching and Finite Automata Theory, 2d ed., by Zvi Kohavi. Copyright © 1978, 1970. Used by permission of McGraw-Hill, Inc. All rights reserved.

При  $b = 2$  число представлено в двоичной системе счисления. Например, двоичное число 1101.01 соответствует многочлену

$$1101.01 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}, \quad \text{т. е.}$$

$$1101.01 = \sum_{i=-2}^3 a_i 2^i,$$

где  $a_{-2} = a_0 = a_2 = a_3 = 1$  и  $a_{-1} = a_1 = 0$

Число  $N$  по основанию  $b$  обозначается через  $(N)_b$ . Всегда, когда основание не указано, подразумевается основание 10. В табл. 3.1 даны представления целых чисел

Таблица 3.1. Представление целых чисел

2	Основание			
	4	8	10	12
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	10	4	4	4
0101	11	5	5	5
0110	12	6	6	6
0111	13	7	7	7
1000	20	10	8	8
1001	21	11	9	9
1010	22	12	10	$\alpha$
1011	23	13	11	$\beta$
1100	30	14	12	10
1101	31	15	13	11
1110	32	16	14	12
1111	33	17	15	13

от 0 до 15 в нескольких системах счисления.

Дополнение цифры  $a$  по основанию  $b$  определяется как

$$a' = (b - 1) - a.$$

Дополнение<sup>1)</sup>  $a'$  равно разности между наибольшей цифрой по основанию  $b$  и цифрой  $a$ . В двоичной системе счисления  $0' = 1$  и  $1' = 0$ , так как  $b = 2$ . В десятичной системе счисления

наибольшей является цифра 9. Поэтому, например, дополнением<sup>1)</sup> 3 будет  $9 - 3 = 6$ .

### Преобразование чисел из одной системы счисления в другую

Пусть  $N$  — число в системе с основанием  $b_1$ , которое требуется представить в системе с основанием  $b_2$ . Удобно различать два случая. В первом случае  $b_1 < b_2$ , и следовательно, при переходе к основанию  $b_2$  можно использовать арифметику этой системы. Метод преобразования состоит в представлении  $(N)_{b_1}$  в виде многочлена по степеням  $b_1$  и вычислении этого многочлена с помощью арифметики по основанию  $b_2$ .

<sup>1)</sup> В десятичной системе дополнение также называется поразрядным дополнением до 9, а в двоичной системе — дополнением до 1.

**Пример.** Преобразуем числа  $(432.2)_8$  и  $(1101.01)_2$  в числа в системе с основанием 10.

$$(432.2)_8 = 4 \cdot 8^2 + 3 \cdot 8^1 + 2 \cdot 8^0 + 2 \cdot 8^{-1} = (282.25)_{10},$$

$$(1101.01)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} =$$

$$= (13.25)_{10}.$$

В обоих случаях арифметические действия выполняются по основанию 10.

Если  $b_1 > b_2$ , то удобнее пользоваться арифметикой по основанию  $b_1$ . Рассмотрим процедуру преобразования отдельно для целой и дробной частей  $N$ . Пусть  $(N)_{b_1}$  — целое число, значение которого по основанию  $b_2$  дается многочленом

$$(N)_{b_1} = a_{q-1}b_2^{q-1} + a_{q-2}b_2^{q-2} + \dots + a_1b_2^1 + a_0b_2^0.$$

Для нахождения значений  $a_i$  разделим этот многочлен на  $b_2$ :

$$(N)_{b_1}/b_2 = \underbrace{a_{q-1}b_2^{q-2} + a_{q-2}b_2^{q-3} + \dots + a_1 + a_0/b_2}_{Q_0}.$$

Таким образом, младший разряд  $(N)_{b_2}$ , т. е.  $a_0$ , равен первому остатку. Следующий значащий разряд  $a_1$  получается делением частного  $Q_0$  на  $b_2$ , т. е.

$$(Q_0/b_2)_{b_1} = \underbrace{a_{q-1}b_2^{q-3} + a_{q-2}b_2^{q-4} + \dots + a_1/b_2}_{Q_1}.$$

Остальные  $a$  вычисляются посредством повторных делений остатков до тех пор, пока  $Q_{q-1}$  не станет равным нулю. Если  $N$  — конечное число, то этот процесс должен завершиться.

Если  $(N)_{b_1}$  — дробь, то применяется двойственная процедура.  $(N)_{b_1}$  можно выразить в системе с основанием  $b_2$  как

$$(N)_{b_1} = a_{-1}b_2^{-1} + a_{-2}b_2^{-2} + \dots + a_{-p}b_2^{-p}.$$

Старшая цифра  $a_{-1}$  может быть найдена умножением этого многочлена на  $b_2$ , т. е.

$$b_2 \cdot (N)_{b_1} = a_{-1} + a_{-2}b_2^{-1} + \dots + a_{-p}b_2^{-p+1}.$$

Если это произведение меньше 1, то  $a_{-1}$  равна 0, если же оно больше или равно 1, то  $a_{-1}$  равна целой части произведения. Следующая старшая цифра  $a_{-2}$  определяется путем умножения дробной части указанного выше произведения на  $b_2$  и выделения его целой части и т. д. Этот процесс может быть бесконечным, так как не всегда можно представить дробь по основанию  $b_2$  конечным набором цифр<sup>1)</sup>.

<sup>1)</sup> Преобразование дробной части числа прекращается при достижении требуемой точности. — *Прим. ред.*

При преобразовании восьмеричных чисел (чисел по основанию 8) в двоичные и обратном преобразовании можно воспользоваться более простой процедурой. Поскольку  $8 = 2^3$ , каждую восьмеричную цифру можно представить тремя двоичными разрядами. Например,  $(6)_8$  можно записать как  $(110)_2$  и т. д. Процедура преобразования двоичного числа в восьмеричное сводится к разбиению двоичного числа на группы из трех разрядов, начиная от двоичной точки, и замене каждой группы на соответствующую восьмеричную цифру.

### Пример

$$(123.4)_8 = (001\ 010\ 011.100)_2,$$

$$(1010110.0101)_2 = (001\ 010\ 110.010\ 100) = (126.24)_8.$$

Аналогичная процедура может применяться при переходе от двоичных к шестнадцатеричным числам (по основанию 16) с тем отличием, что теперь четыре двоичных разряда задают одну шестнадцатеричную цифру. На самом деле при переходе от представления любого числа по основанию  $b_1$  к представлению по основанию  $b_2$ , где  $b_2 = b_1^k$ ,  $k$  цифр числа по основанию  $b_1$  группируются и могут быть заменены на одну цифру по основанию  $b_2$ .

## Двоичная арифметика

Двоичная система счисления широко используется в дискретных системах. Хотя подробное рассмотрение дискретной арифметики не является нашей целью, мы изложим простейшие методы двоичной арифметики. В табл. 3.2 приведены основные арифметические операции; здесь указаны значения суммы, цифры переноса в следующий разряд, разности и заема из старшего разряда, а также произведения для каждой комбинации двоичных разрядов (битов) 0 и 1.

Двоичное сложение сходно с десятичным сложением. Складываются соответствующие биты, и, если перенос в следующий

Таблица 3.2. Простейшие бинарные операции

Биты $a$ $b$	Сумма $a + b$	Перенос	Разность $a - b$	Зем	Произведение $a \cdot b$
0 0	0	0	0	0	0
0 1	1	0	1	1	0
1 0	1	0	1	0	0
1 1	0	1	0	0	1

разряд равен 1, он прибавляется к двоичному разряду слева.

**Пример**

$$\begin{array}{r}
 1111 \quad = \text{Переносы единиц, равные 1} \\
 + \quad 1111.01 = (15.25)_{10} \\
 \quad 0111.10 = (7.50)_{10} \\
 \hline
 10110.11 = (22.75)_{10}
 \end{array}$$

При вычитании, если имеет место заем 1, а следующий слева разряд уменьшаемого равен 1, то значение этого разряда заменяется на 0, и вычитание продолжается обычным образом. Однако если указанный разряд уменьшаемого равен 0, то он заменяется на 1, так же как и любой последующий левый разряд, равный 0. Первый слева единичный разряд уменьшаемого заменяется на нулевой, и вычитание продолжается.

**Пример**

$$\begin{array}{r}
 11 \quad = \text{Заемы единиц} \\
 - \quad 10010.11 = (18.75)_{10} \\
 \quad 01100.10 = (12.50)_{10} \\
 \hline
 00110.01 = (6.25)_{10}
 \end{array}$$

Как и в случае десятичных чисел, умножение двоичных чисел выполняется с помощью последовательного сложения, а деление — с помощью последовательного вычитания.

**Пример.** Перемножить следующие двоичные числа:

$$\begin{array}{r}
 \times 11001.1 = (25.5)_{10} \\
 \quad 110.1 = (6.5)_{10} \\
 \hline
 110011 \\
 000000 \\
 110011 \\
 110011 \\
 \hline
 10100101.11 = (165.75)_{10}
 \end{array}$$

**Пример.** Разделить двоичное число 1000100110 на 11001:

$$\begin{array}{r}
 10110 = \text{Частное} \\
 11001 \overline{) 1000100110} \\
 \underline{11001} \phantom{00} \\
 00100101 \\
 \underline{11001} \phantom{00} \\
 0011001 \\
 \underline{11001} \phantom{00} \\
 00000 = \text{Остаток}
 \end{array}$$

### 3.2. ДВОИЧНЫЕ КОДЫ

Хотя двоичная система счисления обладает многими практическими преимуществами и широко используется в ЭВМ, часто удобно иметь дело с десятичной системой счисления, особенно при значительном объеме вводимой и выводимой информации, поскольку большая часть данных, которые используют люди, представлены в десятичной форме. С целью упрощения проблемы связи человека с машиной разработан ряд кодов, в которых десятичные цифры представлены последовательностями двоичных разрядов.

#### Взвешенные коды

Для 10 цифр 0, 1, ..., 9 в двоичной форме необходимо иметь по крайней мере четыре двоичных разряда. Так как возможны 16 комбинаций для четырех двоичных разрядов, из которых используются лишь десять, можно построить очень большое число различных кодов. Особый интерес представляет класс **взвешенных кодов**, характеризующихся тем, что в них каждому двоичному разряду приписывается «вес» и каждая группа из четырех битов задает десятичное число, равное сумме весов тех двоичных разрядов, значения которых равны 1. Другими словами, если  $w_1, w_2, w_3$  и  $w_4$  — веса двоичных разрядов и  $x_1, x_2, x_3, x_4$  — соответствующие их значения, то десятичная цифра  $N = w_4x_4 + w_3x_3 + w_2x_2 + w_1x_1$  представляется двоичной последовательностью  $x_4x_3x_2x_1$ . Последовательность двоичных разрядов, представляющая десятичную цифру, называется **кодовым набором**.

Таблица 3.3. Примеры взвешенных двоичных кодов

Десятичная цифра					$w_4 \quad w_3 \quad w_2 \quad w_1$							
	8	4	2	1	2	4	2	1	6	4	2	-3
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	1
2	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1	1	0	1	1
6	0	1	1	0	1	1	0	0	0	1	1	0
7	0	1	1	1	1	1	0	1	1	1	0	1
8	1	0	0	0	1	1	1	0	1	0	1	0
9	1	0	0	1	1	1	1	1	1	1	1	1

Таким образом, указанная последовательность  $x_4x_3x_2x_1$  является кодовым набором для  $N$ . Ряд взвешенных четырехразрядных двоичных кодов приведен в табл. 3.3.

Двоичным разрядам в первом коде табл. 3.3 приписаны веса 8, 4, 2, 1. В результате кодовый набор, соответствующий любому десятичному разряду, является двоичным эквивалентом этого разряда; например, 5 представляется кодовым набором 0101 и т. д. Рассмотренный код называется ДД (двоично-десятичным)-кодом. Для каждого кода из табл. 3.3 десятичный разряд, соответствующий данному кодовому набору, равен сумме весов тех двоичных разрядов, которые имеют 1. Во втором коде с весами 2, 4, 2, 1 десятичное число 5 записывается в виде 1011, поскольку  $2 \cdot 1 + 4 \cdot 0 + 2 \cdot 1 + 1 \cdot 1 = 5$ . Веса, присвоенные двоичным разрядам, могут оказаться и отрицательными, что имеет место для кода (6,4,2, —3). В этом коде десятичное число 5 представляется в виде 1011, так как  $6 \cdot 1 + 4 \cdot 0 + 2 \cdot 1 - 3 \cdot 1 = 5$ .

Очевидно, что некоторые десятичные числа представляются в (2,4,2,1)- и (6,4,2, —3)-кодах не единственным образом. Например, десятичное число 7 можно записать в (2,4,2,1)-коде как 1101, так и 0111. Выбор представления чисел из табл. 3.3 обуславливается самодополняемостью кодов. Код называется **самодополняющимся**, если кодовый набор дополнения числа  $N$  до 9, т. е.  $9 - N$ , может получаться из кодового набора  $N$  путем замены всех единиц на нули и нулей на единицы. Например, десятичное число 3 записывается как 1001 в (6,4,2, —3)-коде, а десятичное число 6 — как 0110. В (2,4,2,1)-коде десятичное число 2 задается как 0010, а десятичное число 7 — как 1101. Отметим, что ДД-код не является самодополняющимся. Можно показать, что, для того чтобы взвешенный код был самодополняющимся, необходимо чтобы сумма его весов равнялась 9. Существуют лишь четыре самодополняющихся кода с положительными весами, а именно (2,4,2,1), (3,3,2,1), (4,3,1,1), (5,2,1,1). Кроме того, имеются 13 самодополняющихся кодов как с положительными, так и отрицательными весами.

### Невзвешенные коды

Существует много невзвешенных двоичных кодов, примеры двух из которых приведены в табл. 3.4. Код с **избытком три**<sup>1)</sup> формируется путем сложения каждого кодового набора с 0011. Так, например, десятичное число 7 записывается в коде с избытком три в виде  $0111 + 0011 = 1010$ . Этот код является самодополняющимся и обладает рядом свойств, оправдавших его применение в первых ЭВМ с десятичным основанием.

Во многих практических приложениях, например при аналого-цифровом преобразовании данных, желательно пользоваться

<sup>1)</sup> В этом коде десятичная цифра  $N$  представляется двоичным эквивалентом  $N + 3$ . — *Прим. перев.*



Таблица 3.4. Невзвешенные двоичные коды

Десятичная цифра	Код с избытком три	Циклический
0	0 0 1 1	0 0 0 0
1	0 1 0 0	0 0 0 1
2	0 1 0 1	0 0 1 1
3	0 1 1 0	0 0 1 0
4	0 1 1 1	0 1 1 0
5	1 0 0 0	1 1 1 0
6	1 0 0 1	1 0 1 0
7	1 0 1 0	1 0 0 0
8	1 0 1 1	1 1 0 0
9	1 1 0 0	0 1 0 0

кодами, в которых все последовательные кодовые наборы отличаются друг от друга лишь одним разрядом. Эти коды называют **циклическими кодами**. Второй код в табл. 3.4 служит примером такого кода. (В нем, так же как во всех циклических кодах, кодовые наборы для десятичных чисел 0 и 9 различаются только в одном разряде.) Особенно важным среди циклических кодов является **код Грея**. Четырехразрядный код Грея приведен в табл. 3.5. Этот циклический код часто применяется благодаря простоте процедуры перехода от двоичной системы счисления к коду Грея.

Пусть  $g_n \dots g_2 g_1 g_0$  — кодовый набор в коде Грея с  $(n + 1)$  разрядами, а  $b_n \dots b_2 b_1 b_0$  — соответствующее двоичное число,

Таблица 3.5. Полный четырехразрядный код Грея

Десятичное число	Код Грея				Двоичный код			
	$g_3$	$g_2$	$g_1$	$g_0$	$b_3$	$b_2$	$b_1$	$b_0$
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

где индексы 0 и  $n$  обозначают младший и старший разряды соответственно.

Тогда разряд  $g_i$  можно выразить через соответствующее двоичное число следующим образом:

$$g_i = b_i \oplus b_{i+1}, \quad 0 \leq i \leq n-1,$$

$$g_n = b_n,$$

где  $\oplus$  — символ операции сложения по модулю 2, которая определяется соотношениями

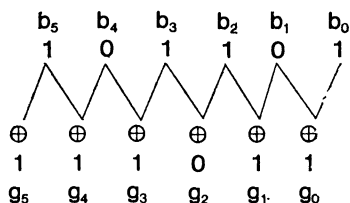
$$0 \oplus 0 = 0, \quad 1 \oplus 1 = 0,$$

$$0 \oplus 1 = 1, \quad 1 \oplus 0 = 1.$$

09	0 00	0 000
01	0 01	0 001
11	0 11	0 011
10	0 10	0 010
	1 10	0 110
	1 11	0 111
	1 01	0 101
	1 00	0 100
		1 100
		1 101
		1 111
		1 110
		1 010
		1 011
		1 001
		1 000

Рис. 3.1. Рефлексивность кода Грея.

Например, кодовый набор Грея 111011, соответствующий двоичному числу 101101, находится следующим образом:



Для перехода от кода Грея к двоичному следует просмотреть его, начиная с крайнего левого разряда, и положить  $b_i = g_i$ , если число единиц, предшествующих  $g_i$ , четно и  $b_i = g_i'$ , если это число нечетно. (Отметим, что нулевое число единиц также считается четным.) Например, кодовый набор Грея 1001011 соответствует двоичному числу 1110010. Доказательства правильности описанных преобразований предоставляются читателю в качестве упражнений.

$n$ -Разрядный код Грея относится к классу так называемых **рефлексивных (отраженных) кодов**. Термин «рефлексивный» используется для обозначения кодов, обладающих таким свойством, что для построения  $n$ -разрядного кода следует найти отображение  $(n - 1)$ -разрядного кода (рис. 3.1). На рис. 3.1, а приведен двухразрядный код Грея. Трехразрядный код Грея можно построить путем отражения двухразрядного кода относительно горизонтальной оси, расположенной ниже описывающей его таблицы, и присвоения наибольшему значащему разряду выше оси значения 0, а ниже оси — значения 1. Аналогично можно построить четырехразрядный код Грея, исходя из трехразрядного кода (рис. 3.1, в).

### 3.3. ОБНАРУЖЕНИЕ И ИСПРАВЛЕНИЕ ОШИБОК

В рассмотренных кодах каждый кодовый набор состоит из четырех двоичных разрядов, количество которых минимально для представления 10 десятичных цифр. Хотя такие коды пригодны для задания десятичных цифр, они весьма чувствительны к ошибкам, возникающим из-за сбоев в работе аппаратуры или помех в каналах передачи данных. В любой практической системе всегда существует ненулевая вероятность появления одиночной ошибки. Вероятность одновременного возникновения двух или более ошибок, хотя и отлична от 0, но значительно меньше. Поэтому мы ограничимся рассмотрением обнаружения и исправления одиночных ошибок.

#### Коды с обнаружением ошибок

Появление одиночной ошибки в одном из разрядов четырех-разрядного двоичного кода может привести к неправильному, но допустимому кодовому набору. Например, если в ДД-коде ошибочным оказался младший разряд в 0110, то это приведет к кодовому набору 0111, который будучи допустимым кодовым набором, будет неправильно интерпретирован приемным устройством. Если код такой, что появление любой одиночной ошибки преобразует допустимый кодовый набор в недопустимый кодовый набор, то его называют **кодом с обнаружением (одиночной) ошибки**. Два таких кода приведены в табл. 3.6.

Таблица 3.6. Коды с обнаружением ошибки

Десятичная цифра	ДД-код с проверкой на четность					Код «2 из 5»				
	8	4	2	1	p	0	1	2	4	7
0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	1	1	1	0	0	0
2	0	0	1	0	1	1	0	1	0	0
3	0	0	1	1	0	0	1	1	0	0
4	0	1	0	0	1	1	0	0	1	0
5	0	1	0	1	0	0	1	0	1	0
6	0	1	1	0	0	0	0	1	1	0
7	0	1	1	1	1	1	0	0	0	1
8	1	0	0	0	1	0	1	0	0	1
9	1	0	0	1	0	0	0	1	0	1

Обнаружение ошибки в любом коде из табл. 3.6 производится **проверкой на четность**. Эта проверка основана на присоединении к каждому кодовому набору дополнительного разряда с тем, чтобы количество единиц в любом кодовом наборе данного

кода было нечетным или четным<sup>1)</sup>). В кодах из табл. 3.6 осуществляется **проверка на четность**. ДД-код с проверкой на четность непосредственно получается из ДД-кода табл. 3.3. Дополнительный разряд  $p$  называется **контрольным разрядом четности**. Код, состоящий из всех 10 возможных комбинаций пятиразрядных кодовых наборов с двумя единицами, называется **кодом «2 из 5»**. Этот код является взвешенным (если не считать кодовый набор для десятичной цифры 0) и может быть получен из (1,2,4,7)-кода.

В каждом из кодов табл. 3.6 число единиц в любом наборе четно. Если появляется одиночная ошибка, она преобразует допустимый кодовый набор в неправильный, вследствие чего сразу можно обнаружить ошибку. Хотя проверка на четность предназначена для обнаружения лишь одиночных ошибок, на самом деле она выявляет любое нечетное число ошибок и некоторое четное число ошибок. Например, при получении кодового набора 10100 в ДД-коде с проверкой на четность сразу становится ясно, что он ошибочен, хотя проверка на четность не выявляет это. Однако определить исходный кодовый набор не удается.

В общем случае для построения  $n$ -разрядного кода с обнаружением ошибок требуется не более половины из  $2^n$  возможных комбинаций разрядов. Выбор кодовых наборов производится таким образом, чтобы для преобразования одного допустимого кодового набора в другой допустимый кодовый набор по крайней мере два разряда должны иметь противоположные значения. Для четырехразрядных кодовых наборов это условие означает, что из 16 возможных кодовых наборов можно выделить лишь 8 допустимых кодовых наборов. Так, для построения кода с обнаружением ошибки для 10 десятичных цифр необходимо иметь по крайней мере 5 двоичных разрядов. Естественно, определить **расстояние** между двумя кодовыми наборами как число их несовпадающих разрядов. Например, расстояние между 1010 и 0100 равно трем, поскольку эти кодовые наборы отличаются в трех разрядах. **Минимальным кодовым расстоянием** называется наименьшее число разрядов, в которых различаются любые два кодовых набора. Например, минимальное кодовое расстояние для ДД-кода или кода с избытком три равно 1, а для кодов из табл. 3.6 оно равно двум. Ясно, что код является кодом с обнаружением ошибок тогда и только тогда, когда его минимальное расстояние не меньше двух.

<sup>1)</sup> При кодировании целесообразно число единиц в кодовом наборе с обнаружением одиночной ошибки выбирать нечетным. Тогда любое кодовое представление, в том числе и для нуля, будет иметь хотя бы одну 1. Это дает возможность отличить полное отсутствие информации от передачи нуля в том случае, когда 1 изображается наличием электрического сигнала, а нуль — его отсутствием. — *Прим. ред.*

### Коды с исправлением ошибок

Для того чтобы код исправлял ошибки, необходимо увеличить его минимальное расстояние. Например, рассмотрим трехразрядный код, состоящий лишь из двух допустимых кодовых наборов 000 и 111. Если одиночная ошибка появляется в первом кодовом наборе, то он может преобразоваться в 001, 010 или 100. Второй кодовый набор из-за одиночной ошибки может превратиться в 110, 101 или 011. Отметим, что в каждом из этих случаев появляется недопустимый кодовый набор, отличный от остальных. Ясно, что указанный код обнаруживает ошибки, так как его минимальное кодовое расстояние равно трем. Если еще предположить, что может возникнуть лишь одиночная ошибка, то удастся установить местоположение этой ошибки и исправить ее, поскольку каждая ошибка приводит к недопустимому кодовому набору, соответствующему только одному из допустимых кодовых наборов. Например, два кодовых набора 000 и 111 образуют код с исправлением ошибок, минимальное расстояние для которого равно трем. Код называется **кодом с исправлением ошибок**, если всегда из неправильного кодового набора можно получить правильный кодовый набор. В данном разделе будут рассмотрены коды Хэмминга, являющиеся одним из типов кодов с исправлением ошибок.

Если минимальное кодовое расстояние равно трем, то любая одиночная ошибка переводит допустимый кодовый набор в недопустимый, находящийся на расстоянии, равном единице от исходного кодового набора, и на расстоянии, равном двум от любого другого допустимого кодового набора. Поэтому в коде с минимальным расстоянием, равным трем, можно исправить любую одиночную ошибку или обнаружить любую двойную ошибку. Аналогично если код имеет минимальное расстояние, равное четырем, то при его использовании можно либо исправить одиночную ошибку и обнаружить двойную либо же обнаружить тройную ошибку. Главным для исправления ошибок является то, что необходимо иметь возможность **обнаруживать и выделять** местоположение ошибочных разрядов. Если местоположение ошибки определено, то исправление ее производится путем замены ошибочного разряда на его дополнение.

Рассмотрим основные принципы построения кодов Хэмминга с исправлением ошибок. К каждому набору из  $m$  информационных разрядов, или сообщению, присоединяются  $k$  разрядов  $p_1, p_2 \dots p_k$  проверки на четность. Затем присваивают десятичное значение позиции каждого из  $(m + k)$  разрядов кодового набора, начиная со значения 1 для старшего разряда и кончая значением  $(m + k)$  для младшего разряда. Производится  $k$  проверок на четность числа единиц в выбранных разрядах каж-

дого кодового набора. Результат каждой проверки на четность записывается как 1 и 0 в зависимости от того, обнаружена ошибка или нет. По результатам этих проверок строится двоичное число  $c_1 c_2 \dots c_k$ , равное десятичному значению, присвоенному местоположению ошибочного разряда, если произошла ошибка, и нулю при ее отсутствии. Это число называется **номером позиции**.

Число разрядов  $k$ , предназначенных для обнаружения ошибочных позиций, должно быть достаточно большим для указания положения любой из  $(m + k)$  возможных одиночных ошибок; при отсутствии ошибки эти разряды имеют нулевые значения. Следовательно,  $k$  должно удовлетворять неравенству  $2^k \geq m + k + 1$ . Так, например, если исходное сообщение представлено в ДД-коде, для которого  $m = 4$ , то  $k = 3$  и потребуются по крайней мере три дополнительных разряда для проверки на четность. Таким образом, результирующий код будет состоять из семи разрядов. Тогда, если номер позиции равен 101, ошибочным является пятый разряд. Однако если номер позиции равен 000, то сообщение правильное.

Для определения контрольных разрядов лишь через информационные разряды и независимо друг от друга их размещают в позициях 1, 2, ...,  $2^{k-1}$ . Таким образом, если  $m = 4$ ,  $k = 3$ , эти разряды находятся в позициях 1, 2, 4, а на остальных позициях находятся биты исходного сообщения в ДД-коде. Например, в кодовом наборе 1100110 контрольные разряды (выделенные полужирным шрифтом)  $p_1 = 1$ ,  $p_2 = 1$ ,  $p_3 = 0$ , а информационными разрядами являются 0, 1, 1, 0, которые соответствуют десятичному числу 6.

Теперь опишем способ построения кода Хэмминга при  $m = 4$  и  $k = 3$ . Как уже упоминалось, контрольные разряды должны задаваться так, чтобы номер позиции соответствовал положению ошибочного разряда. Табл. 3.7 содержит перечень семи ошибочных позиций и соответствующих им значений номеров

Таблица 3.7. Номера позиций

Позиция ошибки	Номер позиции		
	$c_1$	$c_2$	$c_3$
0 (ошибка отсутствует)	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

позиций. Ясно, что если ошибка появляется в позициях 1, 3, 5 или 7, то младший разряд номера позиции, т. е.  $c_3$ , должен равняться 1. Если код построен так, что в каждом кодовом наборе сумма битов в позициях 1, 3, 5 и 7 четная, то появление одиночной ошибки в любой из этих позиций приведет к нечетному числу единиц в указанных позициях. В этом случае младший разряд номера позиции принимается равным 1. Если среди этих разрядов нет ошибочных, то проверка на четность сведется к проверке их четности и младший разряд будет равен 0.

Из табл. 3.7 видно, что если произойдет ошибка в позициях 2, 3, 6 или 7, то центральный разряд номера позиции окажется равным 1. Следовательно, при построении кода должна проверяться четность числа единиц в позициях 2, 3, 6, 7. Если же их проверка на четность сводится к проверке нечетности, то соответствующий разряд номера позиции, т. е.  $c_2$ , полагается равным 1; в противном случае  $c_2 = 0$ . Наконец, если ошибка происходит в позициях 4, 5, 6 или 7, то старший разряд номера позиции, т. е.  $c_1$ , должен равняться 1. Следовательно, если позиции 4, 5, 6 или 7 при отсутствии ошибки имеют четное число единиц, то ошибка в любой из них будет отмечаться единицей в старшем разряде номера позиции. Таким образом,  $p_1$  выбирается так, чтобы установить четное число единиц в позициях 1, 3, 5, 7;  $p_2$  выбирается так, чтобы установить четное число единиц в позициях 2, 3, 6, 7;  $p_3$  выбирается так, чтобы установить четное число единиц в позициях 4, 5, 6, 7.

Теперь можно построить код, присоединяя соответствующие контрольные разряды к информационным разрядам. Рассмотрим, например, сообщение 0100 (т. е. десятичное число 4).

Позиция:	1	2	3	4	5	6	7
	$p_1$	$p_2$	$m_1$	$p_3$	$m_2$	$m_3$	$m_4$
Исходное сообщение в ДД-коде:			0		1	0	0
Для проверки на четность в позициях 1, 3, 5, 7 требуется $p_1 = 1$ :	1		0		1	0	0
Для проверки на четность в позициях 2, 3, 6, 7 требуется $p_2 = 0$ :	1	0	0		1	0	0
Для проверки на четность в позициях 4, 5, 6, 7 требуется $p_3 = 1$ :	1	0	0	1	1	0	0
Код сообщений:	1	0	0	1	1	0	0

Разряд  $p_1$  полагается равным 1, чтобы проверить четность в позициях 1, 3, 5 и 7. Аналогично легко видеть, что  $p_2$  должен равняться 0,  $p_3 = 1$  с тем, чтобы осуществить проверку четно-

сти в позициях 2, 3, 6, 7 и 4, 5, 6, 7. Код Хэмминга для десятичных цифр в ДД-коде приведен в табл. 3.8.

Таблица 3.8. Код Хэмминга для ДД-кода

Десятичная цифра	Позиция						
	1	2	3	4	5	6	7
	$p_1$	$p_2$	$m_1$	$p_3$	$m_2$	$m_3$	$m_4$
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1

Опишем способ выявления положения ошибки и ее исправления. Пусть, например, передана последовательность 1101001, но из-за ошибки в пятой позиции принята последовательность 1101101. Положение ошибки можно определить путем выполнения трех проверок на четность:

Позиция:	1	2	3	4	5	6	7	
Полученное сообщение:	1	1	0	1	1	0	1	
Проверка на четность позиций 4, 5, 6, 7:				1	1	0	1	$c_1 = 1$ , так как результат проверки нечетный
Проверка на четность позиций 2, 3, 6, 7:		1	0			0	1	$c_2 = 0$ , так как результат проверки четный
Проверка на четность позиций 1, 3, 5, 7:	1		0		1		1	$c_3 = 1$ , так как результат проверки нечетный

Таким образом, полученный номер позиции равен 101; это означает, что ошибка в пятой позиции. Для ее исправления необходимо заменить пятый разряд его дополнением, после чего будем иметь правильное сообщение 1101001.

Легко показать, что для построенного кода Хэмминга минимальное расстояние равно трем. Рассмотрим, например, два исходных четырехразрядных кодовых набора 1001 и 0001, отличающиеся друг от друга только в одной позиции. Поскольку каждый разряд сообщения участвует по крайней мере в двух проверках на четность, результаты проверок на четность, охватывающие разряд, по которому различаются эти кодовые наборы, будут отличаться четностью, поэтому в эти наборы будут добавлены разные контрольные разряды, что сделает расстояние



между наборами равным трем. В качестве примера рассмотрим следующие два кодовых набора.

Позиция:		1	2	3	4	5	6	7
		$p_1$	$p_2$	$m_1$	$p_3$	$m_2$	$m_3$	$m_4$
Первый набор:				1		0	0	1
Второй набор:				0		0	0	1
Первый набор с контрольными								
разрядами четности:		0	0	1		0	0	1
Второй набор с контрольными								
разрядами четности:		1	1	0		0	0	1

Эти наборы различаются только по  $m_1$  (т. е. в третьей позиции). Проверки позиций 1, 3, 5, 7 и 2, 3, 6, 7 дают разные результаты. Поэтому контрольные разряды четности  $p_1$  и  $p_2$  должны быть различными для этих наборов. Ясно, что это утверждение справедливо и тогда, когда исходные кодовые наборы отличаются друг от друга в двух позициях из четырех. Таким образом, для кода Хэмминга минимальное расстояние равно трем.

Если это расстояние увеличить до четырех, присоединяя контрольный разряд четности к коду в табл. 3.8, с тем чтобы все восемь разрядов проверялись на четность, то полученный код можно использовать для исправления одиночной ошибки и обнаружения двойной ошибки следующим образом. Предположим, что произошли две ошибки, тогда полная проверка на четность будет успешной, но номер позиции (определенный так же как и ранее, исходя из первых семи разрядов) укажет на ошибку. Ясно, что подобная ситуация свидетельствует о наличии двойной ошибки. Однако определить местоположение ошибок не удастся. Если имеет место только одиночная ошибка, полная проверка на четность позволит ее обнаружить. Если же номер позиции равен 0, то ошибочным является последний контрольный разряд четности; в противном случае местоположение ошибочного разряда определяется номером позиции. Если все четыре проверки на четность выявляют четное число разрядов, то сообщение правильное.

# 4

## БУЛЕВА АЛГЕБРА И ЛОГИЧЕСКИЕ СХЕМЫ

*Д. Гивоне, Р. Рессер*

### 4.1. ВВЕДЕНИЕ

Для описания поведения и структуры логических схем можно использовать простую алгебру, называемую **булевой**. Этот формальный математический аппарат удобен при описании функционирования составных частей ЭВМ и помогает проектировать логические системы. Кроме того, поскольку одно из главных применений микропроцессоров связано с заменой аппаратной логики на программное обеспечение, операции булевой алгебры часто встречаются и в программах для микроЭВМ.

В этой главе излагаются основы булевой алгебры для двузначной логики. Даются определения нескольких математических операций, и приводятся булевы алгебраические выражения различных типов. Рассматриваются примеры преобразований алгебраических выражений. В заключение устанавливается связь между выражениями булевой алгебры и логическими диаграммами.

### 4.2. БУЛЕВА АЛГЕБРА

Формальная математическая система в общем случае состоит из множества элементов, множества операций и множества постулатов. Булева алгебра как математическая система также определяется этими тремя множествами. Некоторые схемы в ЭВМ обеспечивают обработку двоичных символов, представляющих числа или иную информацию. Другие схемы используются для передачи информации по соответствующим путям. Наконец, имеется ряд схем, применяемых для управления, например для выполнения конкретных функций и указаний различных состояний или условий. Во всех трех случаях сигналы, появляющиеся в тех или иных точках схем, как правило, имеют

---

Adapted from Microprocessors/Microcomputers: An Introduction, by Donald D. Givone and Robert P. Roesser, Copyright © 1980, Used by permission of McGraw-Hill, Inc. All rights reserved.

два различных уровня. Другими словами, сигналы могут представлять двоичные символы или соответствовать значениям истинно/ложно в зависимости от того, должно или не должно произойти некоторое событие, или произошло оно или нет. Таким образом, в алгебре, с которой мы будем работать, достаточно иметь два элемента, соответствующие двум значениям сигналов. Поэтому везде в дальнейшем мы рассматриваем двузначную булеву алгебру. Эта алгебра известна так же, как **переключа- тельная алгебра**.

Два элемента булевой алгебры называются ее **константами**, и мы будем обозначать их 0 и 1. Чтобы не смешивать с двоичными цифрами, эти символы часто называют **логическим 0** и **логической 1**. Иногда логическим 0 и 1 соответствуют двоичные цифры 0 и 1. В других случаях логическая 1 соответствует выполнению некоторого условия, а логический 0 — его невыполнению. Достоинство абстрактной алгебры состоит в том, что ее результаты применимы к элементам различной природы, лишь бы элементы удовлетворяли определенным постулатам.

Таблица 4.1. Определение операции И

$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Для того чтобы с помощью алгебры описать поведение и структуру логической схемы, входным, выходным и внутренним узлам схемы ставят в соответствие **булевы переменные**, которые могут принимать только два значения: логические 0 и 1. Формально переменная в булевой алгебре — это символ, такой, что

$$x = 0, \text{ если } x \neq 1;$$

$$x = 1, \text{ если } x \neq 0.$$

Для обозначения булевых переменных мы будем пользоваться буквами латинского алфавита.

Теперь, после того как определены элементы булевой алгебры, нужно ввести множество операций и задать постулаты, которым эти операции удовлетворяют. Существует несколько булевых операций. Наиболее важные из них И, ИЛИ и НЕ.

Операция И обозначается точкой ( $\cdot$ ) или подразумевается при выписывании рядом булевых переменных. Так, операция И между двумя переменными  $x$  и  $y$  записывается в виде

$$x \cdot y \text{ или } xy.$$

Эту операцию часто называют также **логическим умножением**, или **конъюнкцией**.

Постулаты для операции И даны в табл. 4.1. Как видно из таблицы,  $x \cdot y$  принимает значение логической 1 тогда и только тогда, когда и  $x$ , и  $y$  равны логической 1; в противном случае  $x \cdot y$  равно логическому 0. Хотя в табл. 4.1 логическое умножение определено только для двух переменных, его легко обобщить на любое число переменных. А именно, значение  $x_1 \cdot x_2 \dots x_n$  есть логическая 1 тогда и только тогда, когда значение каждой переменной  $x_1, x_2, \dots, x_n$  равно логической 1; в противном случае  $x_1 \cdot x_2 \dots x_n$  есть логический 0.

Выше было сказано, что операцию И мы будем обозначать при помощи точки или просто записью переменных рядом. В литературе для обозначения этой операции часто используется также символ  $\wedge$ . Операция И над двумя переменными  $x$  и  $y$  при этом записывается в виде  $x \wedge y$ .

Следующая булева операция, которую мы введем,— это операция ИЛИ. Ее мы будем обозначать знаком плюс (+). Таким образом, операция ИЛИ над переменными  $x$  и  $y$  записывается в виде

$$x + y.$$

Таблица 4.2. Определение операции ИЛИ

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Эту операцию часто называют **логическим сложением**, или **дизъюнкцией**.

Постулаты для этой операции даны в табл. 4.2. Из таблицы следует, что значение  $x + y$  есть логический 0 тогда и только тогда, когда обе переменные  $x$  и  $y$  имеют значение логического 0; в противном случае  $x + y$  принимает значение логической 1. Эту операцию также можно обобщить на  $n$  переменных. А именно,  $x_1 + x_2 + \dots + x_n$  есть логическая 1 тогда и только тогда, когда по крайней мере одна из переменных имеет значение логической 1; в противном случае  $x_1 + x_2 + \dots + x_n$  есть логический 0.

Хотя для обозначения операции ИЛИ мы будем использовать только знак плюс, заметим, что в литературе также употребляется символ  $\vee$ . В этом случае операция ИЛИ над переменными  $x$  и  $y$  записывается в виде  $x \vee y$ .

Последняя операция, которую мы введем в этом разделе,— операция НЕ. Ее также называют **отрицанием**, **инверсией** или **дополнением**. Мы будем обозначать ее черточкой над переменной ( $\neg$ ). Так, отрицание переменной  $x$  записывается в виде  $\bar{x}$

Таблица 4.3. Определение операции НЕ

$x$	$\bar{x}$
0	1
1	0

Как показано в табл. 4.3, операция НЕ удовлетворяет следующим постулатам:

$$\bar{x} = 1, \text{ если } x = 0;$$

$$\bar{x} = 0, \text{ если } x = 1$$

или, что то же самое,  $\overline{\bar{0}} = 1$  и  $\overline{\bar{1}} = 0$ .

В литературе также используется штрих (') как символ операции НЕ. В этом случае отрицание  $x$  записывается как  $x'$ .

Теперь можно определить булеву алгебру как математическую систему с элементами логических 0 и логическая 1 и операциями И, ИЛИ и НЕ, постулаты которых заданы в табл. 4.1—4.3 соответственно.

### 4.3. ТАБЛИЦЫ ИСТИННОСТИ И БУЛЕВЫ ВЫРАЖЕНИЯ

Определив компоненты булевой алгебры, мы должны остановиться на том, как они используются. Цель булевой алгебры — описание поведения и структуры логических схем. На рис. 4.1 логическая схема показана в виде «черного ящика». На входе — булевы переменные  $x_1, x_2, \dots, x_n$ , на выходе — переменная  $f$ . Чтобы описать поведение этого черного ящика, нужно выразить выход  $f$  как функцию входных переменных  $x_1, x_2, \dots, x_n$ . Это можно сделать или при помощи таблицы истинности (комбинационной таблицы), или при помощи булевых выражений.

Логическая схема, которую можно полностью описать таблицами истинности или булевыми выражениями, называется **комбинационной схемой**. Комбинационная схема — это такая схема, в которой значения входных переменных в текущий момент времени полностью определяют значения выходных переменных. Другой класс логических схем составляют схемы с внутренней памятью. Такие схемы называют **последовательностными**. Для них значения выходных переменных определяются не только текущими значениями входных переменных, но также их значениями в предыдущие моменты времени. В этой главе мы остановимся на комбинационных схемах.

Как уже отмечалось, каждая из булевых переменных  $x_1, x_2, \dots, x_n$  может принимать только два значения. Все точки внутри

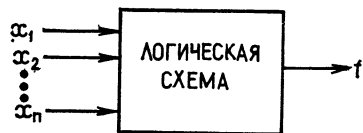


Рис. 4.1. Логическая схема как черный ящик.

черного ящика и его выходные узлы обладают этим свойством. Таблица, содержащая всевозможные комбинации значений входных переменных вместе с соответствующими им значениями выходных переменных, т. е. значениями функций, называется **таблицей истинно-**

Таблица 4.4. Таблица истинности

$x_1$	$x_2$	...	$x_{n-1}$	$x_n$	$f$
0	0	...	0	0	$f(0, 0, \dots, 0, 0)$
0	0	...	0	1	$f(0, 0, \dots, 0, 1)$
0	0	...	1	0	$f(0, 0, \dots, 1, 0)$
0	0	...	1	1	$f(0, 0, \dots, 1, 1)$
...	...	...	...	...	...
1	1	...	1	1	$f(1, 1, \dots, 1, 1)$

сти (или комбинационной таблицей). При  $n$  входных и одной выходной переменной таблица содержит  $2^n$  строк и  $n + 1$  столбцов, как показано в табл. 4.4. Заметим, что самый простой способ включения в таблицу истинности всех возможных входных значений состоит в последовательном переборе в двоичной системе счисления всех чисел от 0 до  $2^n - 1$ . Конечно,  $f$  в каждом столбце будет принимать значение либо 0, либо 1 в зависимости от задаваемой функции.

Второй способ описать поведение комбинационной схемы — это задать **булево выражение**. Оно представляет собой формулу, состоящую из булевых констант и переменных, связанных операциями И, ИЛИ и НЕ. Как в обычных алгебраических выражениях, для задания порядка действий используются скобки. Примером булевого выражения от трех переменных может служить формула

$$f(x_1, x_2, x_3) = [(x_1 + \bar{x}_2)(\bar{x}_1 + x_3)] + (x_2 x_3).$$

Чтобы уменьшить количество скобок, обычно предполагается, что выполнение операции И (логическое умножение) предшествует операции ИЛИ (логическому сложению). Таким образом, приведенное выше выражение обычно записывается в виде

$$f(x_1, x_2, x_3) = (x_1 + \bar{x}_2)(\bar{x}_1 + x_3) + x_2 x_3.$$

### Построение таблицы истинности по булевому выражению

Как таблицы истинности, так и булевы выражения оказываются полезными средствами для описания логических схем. Поскольку они описывают одну и ту же схему, иногда нужно уметь переходить от одной формы описания к другой. Ясно, что, подставив вместо переменных их значения и пользуясь определениями операций И, ИЛИ и НЕ, можно вычислить значение выражения. Одно такое вычисление соответствует ровно одной строке таблицы истинности. Если провести вычисления для каждой комбинации входных значений, то мы получим всю

таблицу истинности. Чтобы проиллюстрировать, как получается таблица истинности по булевому выражению, обратимся опять к формуле

$$f(x_1, x_2, x_3) = (x_1 + \bar{x}_2)(\bar{x}_1 + x_3) + x_2x_3. \quad (4.1)$$

Поскольку эта функция зависит от трех переменных, в таблице истинности должно быть  $2^3 = 8$  строк. В табл. 4.5 восемь комбинаций значений для трех переменных занимают три левых столбца.

Таблица 4.5. Таблица истинности для булева выражения

$$f(x_1, x_2, x_3) = (x_1 + \bar{x}_2)(\bar{x}_1 + x_3) + x_2x_3$$

$x_1$	$x_2$	$x_3$	$\bar{x}_1$	$\bar{x}_2$	$x_1 + \bar{x}_2$	$\bar{x}_1 + x_3$	$(x_1 + \bar{x}_2)(\bar{x}_1 + x_3)$	$x_2x_3$	$f$
0	0	0	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	0	1
0	1	0	1	0	0	1	0	0	0
0	1	1	1	0	0	1	0	1	1
1	0	0	0	1	1	0	0	0	0
1	0	1	0	1	1	1	1	0	1
1	1	0	0	0	1	0	0	0	0
1	1	1	0	0	1	1	1	1	1

Чтобы завершить построение таблицы истинности, нужно вычислить значение выражения (4.1) для каждой из восьми комбинаций. Например, при  $x_1 = 0$ ,  $x_2 = 1$  и  $x_3 = 1$  получим

$$\begin{aligned}
 f(0, 1, 1) &= (0 + \bar{1})(\bar{0} + 1) + 1 \cdot 1 \\
 &= (0 + 0)(1 + 1) + 1 \\
 &= 0 \cdot 1 + 1 \\
 &= 0 + 1 \\
 &= 1.
 \end{aligned}$$

Последний столбец табл. 4.5 содержит окончательные результаты всех восьми вычислений.

Для построения таблицы истинности по булевому выражению можно воспользоваться и другой процедурой. Эта процедура состоит в выполнении отдельных операций выражения над значениями в столбцах таблицы; получаемые результаты заносятся в столбцы. Для пояснения этого подхода снова обратимся к соотношению (4.1). Поскольку в выражении присутствуют  $\bar{x}_1$  и  $\bar{x}_2$ , в табл. 4.5 добавим два столбца для инверсных значений из первого и второго столбцов. Затем выполним операцию ИЛИ над  $x_1$  из первого столбца и  $\bar{x}_2$  из пятого столбца. Результаты заносим в шестой столбец табл. 4.5, который содержит

значения  $x_1 + \bar{x}_2$ . Аналогично по  $\bar{x}_1$  и  $x_3$  получаем  $\bar{x}_1 + x_3$ , которые заносим в седьмой столбец. Затем значения в шестом и седьмом столбцах используем для выполнения операции И, помещая значения  $(x_1 + \bar{x}_2)(\bar{x}_1 + x_3)$  в восьмой столбец. Значения  $x_2x_3$ , содержащиеся в девятом столбце, также легко вычисляются по значениям второго и третьего столбцов. И наконец, выполняем операцию ИЛИ над элементами восьмого и девятого столбцов. Результаты заносим в последний столбец; здесь приведены значения функции  $f$  для всевозможных комбинаций переменных  $x_1, x_2, x_3$ .

### Каноническая сумма минтермов

По таблице истинности также можно составить булево выражение. Одна из его форм называется **канонической суммой минтермов**, или **стандартной суммой произведений**.

Чтобы понять, как строится каноническая сумма минтермов, рассмотрим табл. 4.6. Вторая строка таблицы, в которой функция принимает единичное значение, соответствует входной комбинации  $x_1=0, x_2=0$  и  $x_3=1$ . Рассмотрим терм, являющийся произведением  $\bar{x}_1\bar{x}_2x_3$ . Подставляя в него значения 0, 0 и 1 для  $x_1, x_2$  и  $x_3$ , мы, очевидно, получим логическую 1. Нетрудно убедиться, что для любой из остальных семи комбинаций терм равен логическому 0. Таким образом, в известном смысле терм  $\bar{x}_1\bar{x}_2x_3$  можно использовать для описания второй строки табл. 4.6.

Таблица 4.6. Таблица истинности

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Продолжая аналогичным образом, выберем следующую строку в табл. 4.6 с единичным значением функции. Это третья строка, соответствующая значениям  $x_1=0, x_2=1$  и  $x_3=0$ . Если этот набор значений подставить в терм  $\bar{x}_1x_2\bar{x}_3$ , получим логическую 1. И опять это единственная комбинация значений, приводящая к логической 1. Поэтому можно считать, что терм  $\bar{x}_1x_2\bar{x}_3$  описывает третью строку табл. 4.6. И наконец, рассуждая аналогично, приходим к тому, что шестую строку в табл. 4.6 с комбинацией  $x_1=1, x_2=0$  и  $x_3=1$  представляет терм  $x_1\bar{x}_2x_3$ .

Объединяя полученные результаты, видим, что булево выражение

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2x_3$$

точно описывает табл. 4.6, поскольку каждый терм-произведение соответствует ровно одной строке с единичным значением функ-



ции, а вся сумма (дизъюнкция) соответствует совокупности из трех строк. Для остальных пяти комбинаций входных значений выражение дает нулевой результат. Выражение рассмотренного типа называется **канонической суммой минтермов**, или **стандартной суммой произведений**. Оно определяется как сумма термов-произведений, в каждом из которых каждая переменная встречается ровно один раз либо с отрицанием, либо без него. Сами термы-произведения называются **минтермами**.

В общем случае каждая строка таблицы истинности с единичным значением функции может быть описана минтермом. Переменные, имеющие нулевые значения в строке, входят в минтерм с отрицанием, а переменные со значением единица — без отрицания. Объединение с помощью операции ИЛИ всех минтермов, построенных для строк с единичными значениями функции, дает каноническую сумму минтермов для заданной таблицы истинности.

#### 4.4. ТЕОРЕМЫ БУЛЕВОЙ АЛГЕБРЫ

Можно сформулировать несколько теорем, отражающих основные соотношения булевой алгебры. Наиболее важные из них перечислены в табл. 4.7.

Таблица 4.7. Теоремы булевой алгебры

1a	$\bar{0} = 1$	1b	$\bar{1} = 0$	
2a	$x + 0 = x$	2b	$x \cdot 1 = x$	
3a	$x + 1 = 1$	3b	$x \cdot 0 = 0$	
4a	$x + x = x$	4b	$x \cdot x = x$	Закон идемпотентности
5a	$x + \bar{x} = 1$	5b	$x \cdot \bar{x} = 0$	
6a	$\overline{(\bar{x})} = x$			Двойного отрицания
7a	$x + y = y + x$	7b	$xy = yx$	Коммутативности
8a	$x + xy = x$	8b	$x(x + y) = x$	Поглощения
9a	$x + \bar{x}y = x + y$	9b	$x(\bar{x} + y) = xy$	
10a	$\overline{(x + y)} = \bar{x}\bar{y}$	10b	$\overline{(xy)} = \bar{x} + \bar{y}$	Де Моргана
11a	$\overline{(x + y) + z} = \bar{x} + \bar{y} + \bar{z}$	11b	$\overline{(xy)z} = \bar{x} + \bar{y} + \bar{z}$	Ассоциативности
12a	$x + yz = (x + y)(x + z)$	12b	$x(y + z) = xy + xz$	Дистрибутивности

Первые три пары теорем по существу описывают свойства операций И, ИЛИ, НЕ. Переменные в теоремах могут, вообще говоря, обозначать произвольные булевы выражения. Например, теорема 3а устанавливает, что операция ИЛИ над логической 1 в качестве одного операнда при любом значении другого операнда дает логическую 1. Четвертая теорема, известная как

**закон идемпотентности**, устанавливает, что повторяющиеся переменные в выражении излишни и их можно опустить. Таким образом, понятия возведения в степень и умножения на коэффициенты, отличные от логического 0 или логической 1 (т. е. числа), не имеют смысла в булевой алгебре. Пятая и шестая теоремы подчеркивают взаимодополнительную природу булевых переменных. Закон **двойного отрицания** устанавливает, что дважды выполненное отрицание эквивалентно пустой операции.

В следующих четырех теоремах участвуют по две переменных. Первая из этих теорем — **закон коммутативности** — устанавливает, что порядок переменных при выполнении операции не влияет на результат этой операции. Теоремы 8 и 9 часто бывают полезны при упрощении булевых выражений. Теорема 10 — **закон де Моргана** — описывает эффект отрицания переменных, связанных операциями И и ИЛИ.

В последних двух теоремах участвуют по три переменных. Согласно **закону ассоциативности**, переменные можно группировать в любом порядке как для операции И, так и для операции ИЛИ. **Закон дистрибутивности** устанавливает, что в булевой алгебре допускается вынесение общего множителя за скобки. Особое внимание нужно обратить на тождество теоремы 12а.

Следует отметить свойство симметрии, присущее теоремам булевой алгебры. Все теоремы, кроме теоремы 6, представлены парой соотношений. В каждой паре одно соотношение получается из другого заменой всех операций И на ИЛИ, всех операций ИЛИ на И, всех вхождений логического 0 на логические 1 и всех вхождений логической 1 на логические 0. Это свойство симметрии известно как **принцип двойственности**.

Многие теоремы, приведенные в табл. 4.7, можно обобщить на случай большего числа переменных. Например, закон де Моргана в обобщенной форме можно записать так:

$$\overline{(x + y + \dots + z)} = \bar{x}\bar{y} \dots \bar{z}$$

и

$$\overline{(xy \dots z)} = \bar{x} + \bar{y} + \dots + \bar{z},$$

а закон дистрибутивности:

$$w + xy \dots z = (w + x)(w + y) \dots (w + z)$$

и

$$w(x + y + \dots + z) = wx + wy + \dots + wz.$$

Все перечисленные в табл. 4.7 теоремы можно легко доказать методом **совершенной индукции**, т. е. перебором всех возможностей. Это означает, что справедливость теоремы устанавливается подстановкой в ее левую и правую части всех возможных комбинаций значений переменных и проверкой выполнения равенства для каждой комбинации. Поскольку перемен-

ные в булевой алгебре принимают только два значения, такая процедура оказывается вполне допустимой.

Для иллюстрации доказательства методом совершенной индукции рассмотрим закон дистрибутивности

$$x + yz = (x + y)(x + z).$$

Можно построить таблицы истинности для  $x + yz$  и  $(x + y)(x + z)$ , следуя методике предыдущего раздела. Результаты приведены в табл. 4.8. В пятом столбце содержатся резуль-

Таблица 4.8. Доказательство закона дистрибутивности  $x + yz = (x + y)(x + z)$  методом совершенной индукции

$x$	$y$	$z$	$yz$	$x + yz$	$x + y$	$x + z$	$(x + y)(x + z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

гаты вычисления  $x + yz$  для восьми комбинаций значений трех переменных, а в восьмом — результаты вычисления  $(x + y)(x + z)$ . Поскольку эти столбцы совпадают, можно утверждать, что равенство  $x + yz = (x + y)(x + z)$  выполняется всегда. Таким образом, справедливость теоремы доказана.

#### 4.5. ПРИМЕНЕНИЕ ТЕОРЕМ БУЛЕВОЙ АЛГЕБРЫ

Существует много способов использования рассмотренных выше теорем. Вообще говоря, они дают правила преобразования булевых выражений. С их помощью можно получать эквивалентные выражения. Новые выражения могут оказаться проще или сложнее исходного в зависимости от целей преобразования. Например, мы можем стремиться к простейшему выражению в смысле минимального числа символов или же нашей целью может быть построение канонической суммы минтермов без предварительного построения таблицы истинности.

##### Упрощение выражений

Проиллюстрируем процесс упрощения выражений на следующих трех примерах:

**Пример 4.1.** Выражение

$$(x_1 + x_3)(x_1 + \bar{x}_3)(\bar{x}_2 + x_3)$$

можно упростить следующим образом:

$$\begin{aligned}(x_1 + x_3)(x_1 + \bar{x}_3)(\bar{x}_2 + x_3) &= (x_1 + x_3\bar{x}_3)(\bar{x}_2 + x_3) \text{ по теореме 12a} \\ &= (x_1 + 0)(\bar{x}_2 + x_3) \text{ по теореме 5b} \\ &= x_1(\bar{x}_2 + x_3) \text{ по теореме 2a.}\end{aligned}$$

**Пример 4.2.** Выражение

$$\bar{x}_1\bar{x}_2 + x_1\bar{x}_2 + x_1x_2 + x_2x_3$$

упрощается следующим образом:

$$\begin{aligned}\bar{x}_1\bar{x}_2 + x_1\bar{x}_2 + x_1x_2 + x_2x_3 &= \\ &= \bar{x}_1\bar{x}_2 + x_1\bar{x}_2 + x_1\bar{x}_2 + x_1x_2 + x_2x_3 \text{ по теореме 4a} \\ &= \bar{x}_2\bar{x}_1 + \bar{x}_2x_1 + x_1\bar{x}_2 + x_1x_2 + x_2x_3 \text{ по теореме 7b} \\ &= \bar{x}_2(\bar{x}_1 + x_1) + x_1(\bar{x}_2 + x_2) + x_2x_3 \text{ по теореме 12b} \\ &= \bar{x}_2(x_1 + \bar{x}_1) + x_1(x_2 + \bar{x}_2) + x_2x_3 \text{ по теореме 7a} \\ &= \bar{x}_2 \cdot 1 + x_1 \cdot 1 + x_2x_3 \text{ по теореме 5a} \\ &= \bar{x}_2 + x_1 + x_2x_3 \text{ по теореме 2b} \\ &= x_1 + \bar{x}_2 + x_2x_3 \text{ по теореме 7a} \\ &= x_1 + \bar{x}_2 + x_3 \text{ по теореме 9a.}\end{aligned}$$

**Пример 4.3.** Выражение

$$x_1x_2 + x_2x_3 + \bar{x}_1x_3$$

упрощается следующим образом:

$$\begin{aligned}x_1x_2 + x_2x_3 + \bar{x}_1x_3 &= \\ &= x_1x_2 + x_2x_3 \cdot 1 + \bar{x}_1x_3 \text{ по теореме 2b} \\ &= x_1x_2 + x_2x_3(x_1 + \bar{x}_1) + \bar{x}_1x_3 \text{ по теореме 5a} \\ &= x_1x_2 + x_2x_3x_1 + x_2x_3\bar{x}_1 + \bar{x}_1x_3 \text{ по теореме 12b} \\ &= x_1x_2 + x_1x_2x_3 + \bar{x}_1x_3x_2 + \bar{x}_1x_3 \text{ по теореме 7b} \\ &= x_1x_2 + x_1x_2x_3 + \bar{x}_1x_3 + \bar{x}_1x_3x_2 \text{ по теореме 7a} \\ &= x_1x_2 + \bar{x}_1x_3 \text{ по теореме 8a.}\end{aligned}$$

Как видно из этих примеров, процесс упрощения булевых выражений не является алгоритмическим. Далеко не очевидно, какое из тождеств следует применить на том или ином шаге. Искусство приходит только с опытом. Поэтому для упрощения булевых выражений были разработаны алгоритмические методы.

### Отрицание выражения

Часто над выражениями приходится выполнять операцию отрицания. Это делается с помощью закона де Моргана. Рассмотрим пример.

**Пример 4.4.** Чтобы выполнить отрицание выражения

$$(\bar{x}_1 \bar{x}_2 + x_3) \bar{x}_1 x_4,$$

будем действовать следующим образом:

$$\begin{aligned} \overline{[(\bar{x}_1 \bar{x}_2 + x_3) \bar{x}_1 x_4]} &= \overline{(\bar{x}_1 \bar{x}_2 + x_3)} + \overline{(\bar{x}_1)} + \bar{x}_4 = \\ &= \overline{(\bar{x}_1 \bar{x}_2 + x_3)} + x_1 + \bar{x}_4 = \\ &= \overline{(\bar{x}_1 \bar{x}_2)} \bar{x}_3 + x_1 + \bar{x}_4 = \\ &= [\overline{(\bar{x}_1)} + \overline{(\bar{x}_2)}] \bar{x}_3 + x_1 + \bar{x}_4 = \\ &= (x_1 + x_2) \bar{x}_3 + x_1 + \bar{x}_4. \end{aligned}$$

### Каноническая сумма минтермов

Построение канонической суммы минтермов по заданному выражению — это другое применение рассмотренных теорем. Очевидно, если задано выражение, можно построить таблицу истинности и по ней каноническую сумму минтермов. Однако, применяя теорему  $x + \bar{x} = 1$  и дистрибутивный закон  $x(y + z) = xy + xz$ , можно развернуть в сумму минтермов непосредственно само выражение.

Рассмотрим в качестве иллюстрации выражение

$$x + \bar{x}(z + y\bar{z}).$$

Сначала, чтобы выражение приняло форму суммы произведений, применим закон дистрибутивности. В данном случае получим

$$x + \bar{x}z + \bar{x}y\bar{z}.$$

Затем нужно модифицировать те члены, которые не являются минтермами, чтобы включить в них соответствующие переменные. Первый член  $x$  не содержит  $y$  и  $z$ . Эти переменные можно ввести, доумножая  $x$  на  $(y + \bar{y})(z + \bar{z})$ , что эквивалентно умножению на логическую 1. По тем же причинам второй член, не содержащий  $y$ , следует домножить на  $y + \bar{y}$ . Третий член содержит все переменные, и с ним ничего делать не нужно. Таким образом, можно переписать исходное выражение в виде

$$x(y + \bar{y})(z + \bar{z}) + \bar{x}(y + \bar{y})z + \bar{x}y\bar{z}.$$

Если теперь снова воспользоваться законом дистрибутивности и затем опустить повторяющиеся члены, получим каноническую сумму минтермов для нашего выражения

$$xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}y\bar{z}.$$

### Каноническое произведение макстермов

Каноническим называется выражение, которое единственно для заданной функции и имеет стандартную форму. Каноническое выражение может быть полезным при установлении эквивалентности функций. А именно, две функции эквивалентны, если их канонические выражения совпадают. Каноническая сумма минтермов представляет собой сумму термов-произведений, причем каждая переменная входит в каждый терм. Другая стандартная формула в булевой алгебре называется **каноническим произведением макстермов**, или **стандартным произведением сумм**<sup>1)</sup>. Так же как и каноническая сумма минтермов, каноническое произведение макстермов может быть построено либо по таблице истинности, либо путем развертывания заданного булевого выражения.

Таблица 4.9. Таблица истинности для инверсии функции, заданной в табл. 4.6

$x_1$	$x_2$	$x_3$	$f$	$\bar{f}$
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Обратимся снова к табл. 4.6. Таблица определяет функцию  $f$ . Таблица истинности для инверсной функции, т. е. для функции  $\bar{f}$ , получается инверсией значений в последнем столбце. Результат показан в табл. 4.9. Пользуясь способом, изложенным в разд. 4.3, можно записать каноническую сумму минтермов для  $\bar{f}$  в следующем виде:

$$\bar{f}(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + x_1x_2x_3.$$

Если теперь выполнить отрицание над обеими частями этого равенства и воспользоваться законом де Моргана, то получим

$$\begin{aligned} \overline{\bar{f}(x_1, x_2, x_3)} &= \\ &= \overline{f(x_1, x_2, x_3)} = \overline{\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + x_1x_2x_3} = \\ &= \overline{(\bar{x}_1\bar{x}_2\bar{x}_3)(\bar{x}_1x_2x_3)(x_1\bar{x}_2\bar{x}_3)(x_1x_2\bar{x}_3)(x_1x_2x_3)} = \\ &= (x_1 + x_2 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3) \times \\ &\quad \times (\bar{x}_1 + \bar{x}_2 + \bar{x}_3). \end{aligned}$$

<sup>1)</sup> В математической литературе каноническое произведение макстермов называют совершенной конъюнктивной нормальной формой. — *Прим. перев.*

Последнее выражение и является каноническим произведением макстермов функции  $f$ .

**Каноническое произведение макстермов**, или **стандартное произведение сумм**, определяется как такое произведение сумм, в котором каждая переменная встречается в каждой сумме ровно один раз либо с отрицанием, либо без него. Термы в выражении называются **макстермами**.

В общем случае, чтобы построить каноническое произведение макстермов по таблице истинности, нужно сначала выписать таблицу истинности для инверсной функции, заменив 1 на 0, а 0 на 1 в столбце значений функции. Затем нужно записать каноническую сумму минтермов для этой инверсной функции. И наконец, взять отрицание полученного выражения, пользуясь законом де Моргана.

Каноническое произведение макстермов можно также вывести путем алгебраических преобразований заданного выражения. При этом пользуются тождеством  $xx = 0$  и дистрибутивным законом  $x + yz = (x + y)(x + z)$ .

В качестве примера рассмотрим выражение

$$\bar{x}y + \bar{y}z.$$

Поскольку каноническая форма макстермов является произведением сумм, прежде всего нужно привести выражение к соответствующему виду. Осуществить это можно, воспользовавшись законом дистрибутивности. В нашем случае имеем

$$\begin{aligned}\bar{x}y + \bar{y}z &= (\bar{x}y + \bar{y})(\bar{x}y + z) = (\bar{x} + \bar{y})(y + \bar{y})(\bar{x} + z)(y + z) = \\ &= (\bar{x} + \bar{y}) \cdot 1 \cdot (\bar{x} + z)(y + z) = (\bar{x} + \bar{y})(\bar{x} + z)(y + z).\end{aligned}$$

Получив выражение в виде произведения сумм, следует проверить, является ли каждая сумма макстермом. Если нет, то нужно ввести недостающие переменные, используя теорему  $x\bar{x} = 0$ . Для рассматриваемого примера получим

$$\begin{aligned}(\bar{x} + \bar{y})(\bar{x} + z)(y + z) &= (\bar{x} + \bar{y} + 0)(\bar{x} + 0 + z)(0 + y + z) = \\ &= (\bar{x} + \bar{y} + z\bar{z})(\bar{x} + y\bar{y} + z)(x\bar{x} + y + z).\end{aligned}$$

Осталось воспользоваться законом дистрибутивности и удалить повторяющиеся суммы согласно закону идемпотентности. В итоге получаем

$$\begin{aligned}(\bar{x} + \bar{y})(\bar{x} + z)(y + z) &= (\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z})(\bar{x} + y + z) \times \\ &\quad \times (\bar{x} + \bar{y} + z)(x + y + z)(\bar{x} + y + z) = \\ &= (\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z})(\bar{x} + y + z)(x + y + z).\end{aligned}$$

## 4.6. УПРОЩЕНИЕ БУЛЕВЫХ ВЫРАЖЕНИЙ С ПОМОЩЬЮ КАРТ КАРНО

В предыдущем разделе было показано, каким образом теоремы булевой алгебры позволяют осуществлять тождественные преобразования булевых выражений. Поскольку получающиеся при этом выражения эквивалентны, то и комбинационные логические схемы, описываемые ими, будут эквивалентными. Естественно, возникает интересный вопрос: какое выражение в некотором смысле является «простейшим»? Для решения этой задачи разработано несколько методов. На одном из них, использующем карты Карно, мы остановимся в этом разделе.

### Карты Карно

Карты Карно — это графическое представление таблиц истинности. Структура карт Карно для функций двух, трех и четырех переменных вместе с соответствующими таблицами истинности в общей форме показана на рис. 4.2 — 4.4. Как легко видеть, между строками таблицы истинности и клетками (ячейками) на карте Карно существует взаимно однозначное соответствие. Положение каждой ячейки определяется некоторой системой горизонтальных и вертикальных координат, показанных на рисунках. Запись в каждой ячейке есть значение функции при значениях переменных, соответствующих координатам ячейки. На рис. 4.5 приведена таблица истинности и карта Карно для булевой функции

$$f(x, y, z) = \bar{x}(\bar{y} + \bar{z}) + xz.$$

Таблица истинности получена вычислением значения выражения для всех восьми комбинаций переменных, а карта Карно построена в соответствии с общей формой, показанной на рис. 4.3.

Когда картой Карно пользуются для упрощения выражений, на ней строят прямоугольные группы ячеек. В общем случае каждая группа размера  $2^a \times 2^b$  соответствует терму-произведению с  $n - a - b$  переменными, где  $n$  — полное число переменных рассматриваемой функции, а  $a$  и  $b$  — целые неотрицательные числа. Поскольку группы имеют размер  $2^a \times 2^b$ , полное число ячеек в любой группе всегда будет степенью двойки. В дальнейшем мы будем рассматривать только группы размером  $2^a \times 2^b$ .

### Минимальные суммы

Один из методов построения выражения по карте Карно состоит в том, что рассматриваются только ячейки, содержащие логическую 1. Назовем их ячейками с единицей. Они соответст-



$x$	$y$	$f(x,y)$
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

**а**

	$y$	
	0	1
$x$	0	$f(0,0)$ $f(0,1)$
	1	$f(1,0)$ $f(1,1)$

**б**

Рис. 4.2. Булева функция двух переменных.

а — таблица истинности; б — карта Карно.

$x$	$y$	$z$	$f(x,y,z)$
0	0	0	$f(0,0,0)$
0	0	1	$f(0,0,1)$
0	1	0	$f(0,1,0)$
0	1	1	$f(0,1,1)$
1	0	0	$f(1,0,0)$
1	0	1	$f(1,0,1)$
1	1	0	$f(1,1,0)$
1	1	1	$f(1,1,1)$

$a$

	$yz$				
	00	01	11	10	
$x$	0	$f(0,0,0)$	$f(0,0,1)$	$f(0,1,1)$	$f(0,1,0)$
	1	$f(1,0,0)$	$f(1,0,1)$	$f(1,1,1)$	$f(1,1,0)$

$b$

Рис. 4.3 Булева функция трех переменных.

а — таблица истинности; б — карта Карно.

w	x	y	z	f(w,x,y,z)
0	0	0	0	f(0,0,0,0)
0	0	0	1	f(0,0,0,1)
0	0	1	0	f(0,0,1,0)
0	0	1	1	f(0,0,1,1)
0	1	0	0	f(0,1,0,0)
0	1	0	1	f(0,1,0,1)
0	1	1	0	f(0,1,1,0)
0	1	1	1	f(0,1,1,1)
1	0	0	0	f(1,0,0,0)
1	0	0	1	f(1,0,0,1)
1	0	1	0	f(1,0,1,0)
1	0	1	1	f(1,0,1,1)
1	1	0	0	f(1,1,0,0)
1	1	0	1	f(1,1,0,1)
1	1	1	0	f(1,1,1,0)
1	1	1	1	f(1,1,1,1)

**а**

	<b>yz</b>			
	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>wx</b>	<b>00</b>	f(0,0,0,0) f(0,0,0,1) f(0,0,1,1) f(0,0,1,0)		
	<b>01</b>	f(0,1,0,0) f(0,1,0,1) f(0,1,1,1) f(0,1,1,0)		
	<b>11</b>	f(1,1,0,0) f(1,1,0,1) f(1,1,1,1) f(1,1,1,0)		
	<b>10</b>	f(1,0,0,0) f(1,0,0,1) f(1,0,1,1) f(1,0,1,0)		

**б**

Рис. 4.4. Булева функция четырех переменных.

а — таблица истинности; б — карта Карно.

$x$	$y$	$z$	$f$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

**а**

	$yz$			
	00	01	11	10
$x$	0	1	1	0
	1	0	1	0

**б**

Рис. 4.5. Булева функция  $f(x, y, z) = \bar{x}(\bar{y} + \bar{z}) + xz$ .

а — таблица истинности; б — карта Карно.

вуют минтермам в каноническом выражении. Каждая группа  $2^a \times 2^b$  ячеек с единицей соответствует терму-произведению, с помощью которого можно описать какую-то часть таблицы истинности. Если выбрать достаточное количество групп так, чтобы каждая ячейка, содержащая 1, вошла хотя бы в одну группу, и выполнить операцию ИЛИ над соответствующими термами-произведениями, то функция будет полностью описана. Разумно выбирая группы, можно получать простые выражения. Одна из мер степени сложности булевых выражений определяется числом букв, входящих в выражение, т. е. числом переменных и их инверсий. (Переменные и их инверсии часто называют **литералами**.) Выражения в форме суммы произведений с минимальным числом литералов называют **минимальными суммами**.

При выборе групп, дающих минимальную сумму, руководствуются следующими двумя принципами. Во-первых, группа должна быть как можно больше. Дело в том, что чем больше группа, тем меньше литералов в соответствующем произведении. Во-вторых, групп должно быть как можно меньше. Это следует из того факта, что каждой группе соответствует терм-произведение. Уменьшая количество групп, мы уменьшаем количество термов-произведений и как следствие число литералов в выражении.

На рис. 4.6 показана карта Карно с оптимальным выбором групп ячеек с единицей. В данном примере нельзя ни увеличить размеры групп, ни уменьшить их число. Группа в первом столбце имеет размер  $2^2 \times 2^0 = 4 \times 1$ , квадратная группа справа вверху —  $2^1 \times 2^1 = 2 \times 2$ , и самая маленькая группа из двух ячеек в третьей строке —  $2^1 \times 2^0 = 2 \times 1$ . Обратите внимание, что группы могут пересекаться.

Для того чтобы написать выражение по карте Карно, следует воспользоваться «разметкой» координатных осей карты. Нужно для каждой группы отобрать те переменные на координатных осях, значения которых не изменяются в пределах группы. Эти переменные и будут входить в соответствующие термы-произведения. Если переменные равны логическому 0, то они должны входить с отрицанием, если они равны логической 1 — без отрицания.

Для иллюстрации снова обратимся к рис. 4.6. Мы видим, что квадратная группа расположена в первой и второй строках карты. В этих строках переменная  $w$  равна логическому 0.

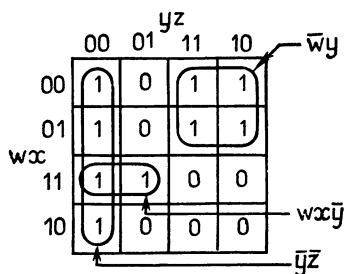


Рис. 4.6. Группы на карте Карно для функции четырех переменных.

Поэтому терм для этой группы должен содержать  $\bar{w}$ . Поскольку переменная  $x$  меняет значение при переходе от первой строки ко второй, она не должна входить в терм. Рассмотрев два столбца, в которых лежит группа, мы обнаружим, что переменная  $y$  имеет постоянное значение, равное логической 1, и, следовательно, должна войти в терм, а  $z$  изменяет значение и поэтому входить в терм не должна. Итак, мы установили, что квадратная группа соответствует терму  $\bar{w}y$ .

Применив аналогичную процедуру к двум другим группам на рис. 4.6, можно найти соответствующие им термы-произведения. Группа в первом столбце соответствует терму  $\bar{y}\bar{z}$ , поскольку и  $y$ , и  $z$  в этом столбце равны логическому 0, а обе переменные  $w$  и  $x$  не сохраняют своих значений для этого столбца. Что же касается самой маленькой группы, то ей соответствует терм  $wx\bar{y}$ . Таким образом, минимальная сумма для рассматриваемой карты Карно равна

$$f(w, x, y, z) = \bar{w}y + \bar{y}\bar{z} + wx\bar{y}.$$

Хотя обычно карты Карно для функций трех и четырех переменных изображаются на плоскости, как показано на рис. 4.3 и 4.4, с точки зрения формирования прямоугольных групп карту нужно считать трехмерной.

Для карт с тремя переменными (рис. 4.3) карту следует рассматривать как цилиндр со склеенными правым и левым краями. Поскольку прямоугольные группы формируются на таком цилиндре, на плоском рисунке та или иная группа может оказаться разорванной. Пример такой разорванной группы приведен на рис. 4.7. Соответствующий этой группе терм получается по описанным выше правилам и равен  $\bar{x}\bar{z}$ .

Разорванные прямоугольные группы могут появляться и на картах с четырьмя переменными. На таких картах нужно считать склеенными не только правый и левый края, но также верхний и нижний. Таким образом, карта с четырьмя переменными (рис. 4.4) должна рассматриваться как поверхность тора. На рис. 4.8 приведены примеры разорванных групп для карт с четырьмя переменными. На рис. 4.8, а группа из четырех ячеек соответствует терму  $\bar{x}\bar{z}$ , а группа из двух ячеек — терму  $\bar{x}yz$ . Особое внимание нужно обратить на группу на рис. 4.8, б. Ячейки в четырех углах образуют одну группу  $2^1 \times 2^1$ , если поверхность считать тороидальной. Этой группе соответствует терм  $\bar{x}\bar{z}$ .

Подводя итог, можно следующим образом сформулировать основной метод оптимального поиска групп на карте Карно, приводящих к минимальным суммам. Прежде всего выбирается ячейка с единицей, которая войдет только в одну группу, не являющуюся подгруппой другой, большей группы. Затем фор-

мируется наибольшая группа, содержащая выбранную ячейку. Далее выбирается другая ячейка с единицей, обладающая тем же свойством и еще не вошедшая в ранее сформированные группы, и формируется ее группа. Этот процесс повторяется до тех пор, пока либо все ячейки, содержащие 1, не окажутся в каких-то группах, либо останутся только такие ячейки, которые можно сгруппировать более чем одним способом. Теперь строится минимальное число групп, покрывающих все оставшиеся единичные ячейки. Следующие примеры поясняют процедуру построения минимальных сумм по картам Карно.

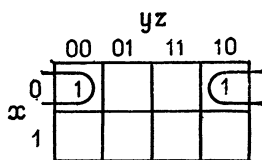


Рис. 4.7. Разорванная группа на карте Карно для функции трех переменных.

**Пример 4.5.** Рассмотрим карту Карно на рис. 4.9. Ячейка в верхнем правом углу может быть объединена в группу с ячейками в других трех углах. Причем эту ячейку нельзя отнести ни к одной другой группе, не являющейся подгруппой этих четырех ячеек. Таким образом, в минимальной сумме должен присутствовать член  $\bar{x}\bar{z}$ . Далее замечаем, что ячейка в первой строке и втором столбце еще не входит ни в одну группу. Ее можно поместить в группу из четырех ячеек, порождающую член  $\bar{x}\bar{y}$ . Наконец, единственную оставшуюся единичную ячейку можно объединить с ячейкой, находящейся непосредственно под ней, что дает член  $wy\bar{z}$ . Таким образом, получаем минимальную сумму

$$f(w, x, y, z) = \bar{x}\bar{z} + \bar{x}\bar{y} + wy\bar{z},$$

содержащую семь литералов.

**Пример 4.6.** Рассмотрим карту Карно на рис. 4.10. Ячейку в левом верхнем углу можно сгруппировать только с ее ближайшим соседом справа. Аналогично ячейка в правом нижнем углу может быть объединена только с ее соседом сверху.

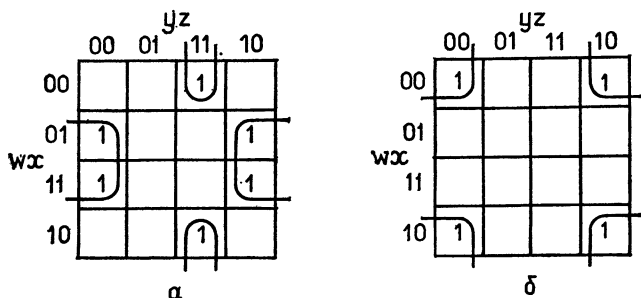


Рис. 4.8. Примеры разорванных групп на карте Карно для функции четырех переменных.

	yz			
	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	0	0	0	1
10	1	1	0	1

Рис. 4.9. Пример 4.5.

	yz			
	00	01	11	10
00	1	1	0	0
01	0	0	1	0
11	1	1	0	1
10	0	1	0	1

Рис. 4.10. Пример 4.6.

Ячейка во второй строке и третьем столбце может группироваться только сама с собой. Теперь осталось три ячейки с единицей, и мы видим, что в отличие от предыдущих случаев их можно поместить более чем в одну группу. Здесь, чтобы завершить процесс, следует выбрать минимальное число групп, покрывающих оставшиеся ячейки. Группы, показанные на карте, соответствуют минимальной сумме

$$f(w, x, y, z) = \bar{w}\bar{x}\bar{y} + wy\bar{z} + \bar{w}xyz + wx\bar{y} + \bar{w}\bar{y}z,$$

которая содержит 16 литералов. Существуют еще две минимальные суммы с тем же числом литералов

$$f(w, x, y, z) = \bar{w}\bar{x}\bar{y} + wy\bar{z} + \bar{w}xyz + \bar{w}\bar{y}z + wx\bar{z},$$

$$f(w, x, y, z) = \bar{w}\bar{x}\bar{y} + wy\bar{z} + \bar{w}xyz + wx\bar{y} + \bar{x}\bar{y}z,$$

которые могли быть получены при другом выборе групп на заключительном этапе. На этом примере видно, что для одной функции может существовать несколько минимальных сумм.

### Минимальные произведения

Выше было рассмотрено построение минимальных сумм по картам Карно. По картам Карно можно также строить минимальные по числу литералов выражения, представляющие собой произведения сумм. Такие выражения будем называть **минимальными произведениями**.

Для получения минимального произведения нужно обратиться к тем ячейкам карты Карно, в которых находятся нули. Назовем их нулевыми ячейками. Фактически мы должны выписать минимальную сумму для инверсии заданной функции, включая в группы нулевые и только нулевые ячейки, максимизируя размеры групп и минимизируя их число. Как и раньше, следует помнить о трехмерной природе карт. Затем нужно взять отрицание полученной суммы, воспользоваться законом де Моргана. Полученное выражение будет соответствовать кар-

	yz			
	00	01	11	10
wx	00	1	1	0
	01	0	0	0
	11	0	0	1
	10	1	1	0

Рис. 4.11. Пример 4.7.

	yz			
	00	01	11	10
wx	00	1	1	0
	01	0	0	1
	11	1	1	0
	10	0	1	0

Рис. 4.12. Пример 4.8.

те Карно (и, следовательно, таблице истинности) и иметь форму произведения сумм с минимальным числом литералов.

**Пример 4.7.** Рассмотрим функцию в примере 4.5 с картой Карно на рис. 4.9. Карта повторена на рис. 4.11, где показаны группы нулевых ячеек для построения минимальной суммы отрицания исходной функции:

$$\bar{f}(w, x, y, z) = yz + \bar{w}x + x\bar{y}$$

или

$$f(w, x, y, z) = \overline{(yz + \bar{w}x + x\bar{y})}.$$

Применив закон де Моргана, получим минимальное произведение

$$f(w, x, y, z) = (\bar{y} + \bar{z})(w + \bar{x})(\bar{x} + y),$$

состоящее только из 6 литералов. В данном случае число литералов в минимальном произведении оказалось меньшим, чем в минимальной сумме.

**Пример 4.8.** Рассмотрим функцию примера 4.6 с картой, приведенной на рис. 4.10 и повторенной на рис. 4.12. Для этой функции можно получить три минимальных произведения в зависимости от выбора групп. Минимальное произведение, соответствующее группам, показанным на рис. 4.12, имеет вид

$$f(w, x, y, z) = (w + \bar{x} + y)(\bar{w} + \bar{y} + \bar{z})(\bar{w} + x + y + z)(w + \bar{y} + z)(w + x + \bar{y}).$$

Два других произведения — это

$$f(w, x, y, z) = (w + \bar{x} + y)(\bar{w} + \bar{y} + \bar{z})(\bar{w} + x + y + z)(w + \bar{y} + z)(x + \bar{y} + \bar{z})$$

и

$$f(w, x, y, z) = (w + \bar{x} + y)(\bar{w} + \bar{y} + \bar{z})(\bar{w} + x + y + z)(w + x + \bar{y})(w + \bar{x} + z).$$

В каждом из них по 16 литералов. Таким образом, для данной функции число литералов в минимальной сумме и минимальном произведении оказалось одинаковым.

### Неопределенные условия

Чтобы закончить рассмотрение карты Карно, нужно остановиться еще на одном вопросе. Напомним, что булевыми выражениями пользуются для описания поведения и структуры логических схем. Каждая строка таблицы истинности (или ячейка на карте Карно) соответствует реакции (переменной на выходе) схемы на определенную комбинацию значений входных сигналов (значений входных переменных). В некоторых случаях известно, что какие-то комбинации на входе появиться не могут или же если они появляются, то значение на выходе не существенно. Для таких ситуаций нет необходимости определять выходные значения схемы, т. е. значения функции в таблице истинности. В таких случаях говорят о **неопределенных условиях**. Карты Карно позволяют строить минимальные суммы и произведения и в этих ситуациях.

На карте Карно неопределенное условие обозначается прочерком в соответствующей ячейке. Такие ячейки могут произвольным образом включаться в группы при построении минимальных сумм и произведений. Любую из них можно включать как в группу единичных, так и в группу нулевых ячеек. Более того, их можно вообще никуда не включать.

На рис. 4.13 показана карта Карно с неопределенными условиями. Карта на рис. 4.13, а порождает минимальную сумму

$$\bar{f}(w, x, y, z) = y\bar{z} + \bar{w}x\bar{y},$$

а карта на рис. 4.13, б — минимальное произведение

$$f(w, x, y, z) = (y + z)(\bar{y} + \bar{z})(\bar{w} + y).$$

Обратите внимание, что ячейка, соответствующая значениям  $w = 0, x = 1, y = 0$  и  $z = 0$ , участвует и в сумме, и в произве-

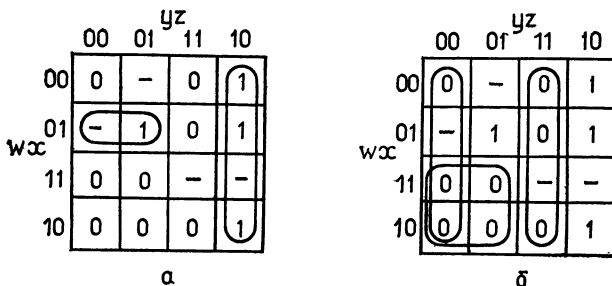


Рис. 4.13. Карты Карно с недоопределенными условиями.

дении, в то время как ячейка, соответствующая значениям  $w = 0$ ,  $x = 0$ ,  $y = 0$  и  $z = 1$ , не используется совсем.

Метод карт Карно можно обобщить для функций с числом переменных больше четырех, однако при этом существенно труднее анализировать карты. Для решения задач большой размерности разработаны машинные методы.

#### 4.7. ЛОГИЧЕСКИЕ СХЕМЫ

Булева алгебра позволяет описать логические аспекты поведения и структуры логических схем. Однако фактически пока мы рассматривали только аспекты поведения. А именно и алгебраические выражения, и таблицы истинности использовались для описания значений выходных переменных в зависимости от значений входных переменных. Однако выражения булевой алгебры могут содержать сведения и о структуре логических схем.

Булева алгебра, как уже отмечалось в предыдущих разделах, содержит три логические операции И, ИЛИ и НЕ. Если мы располагаем элементарными схемами, зависимость выходных значений которых от входных значений соответствует этим трем операциям, то, соединив их между собой в соответствии с булевым выражением, получим логическую схему. Более того, логическая связь входов и выходов этой схемы будет описываться этим выражением. Такие элементарные схемы существуют и называются **вентильми**. Конечно, на входах и выходах вентилях появляются электрические сигналы. Однако если эти сигналы по существу имеют только два различимых уровня, то с одним из них можно ассоциировать логический 0, а с другим — логическую 1. При этом от физических характеристик сигналов можно абстрагироваться и считать, что на входах и выходах появляются непосредственно логические значения.

Символические обозначения вентилях для трех рассмотренных логических операций показаны на рис. 4.14. Поскольку эти символы обозначают булевы операции, поведение вентилях описывается определениями, приведенными в табл. 4.1—4.3: на выходе вентиля И будет логическая 1 тогда и только тогда, когда на всех его входах логические 1; на выходе вентиля ИЛИ будет логическая 1 тогда и только тогда, когда по крайней мере

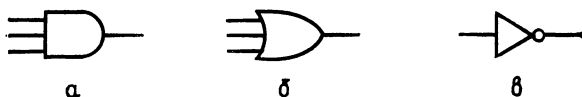


Рис. 4.14. Графические обозначения вентилях.

а — вентиль И; б — вентиль ИЛИ; в — вентиль НЕ (или инвертор).



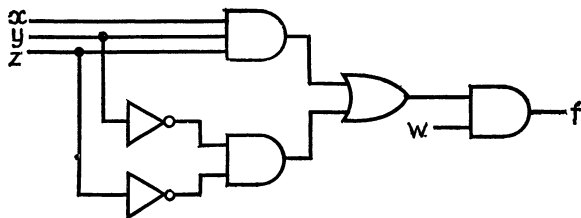


Рис. 4.15. Логическая диаграмма, поведение которой описывается булевым выражением  $f(w, x, y, z) = w(xyz + \bar{y}\bar{z})$ .

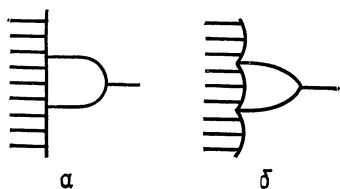


Рис. 4.16. Графические обозначения вентилях со многими входами.

а — вентиль И; б — вентиль ИЛИ.

на одном входе логическая 1, и на выходе вентиля НЕ будет логическая 1 тогда и только тогда, когда на его входе логический 0. Вентиль НЕ также называется **инвертором**.

Рисунки с изображениями элементарных логических схем и их связей называются **логическими диаграммами**. В общем случае, если диаграмма состоит из вентилях и на ней нет обратных связей, соответствующую ей схему называют **комбинационной**. Комбинационная схема не обладает запоминающими свойствами, и поэтому значения на ее выходах определяются полностью значениями на входах в текущий момент времени.

Существует взаимно однозначное соответствие между диаграммами комбинационных схем и булевыми выражениями. Следовательно, булевы выражения являются описаниями комбинационных схем. Рассмотрим в качестве примера диаграмму, показанную на рис. 4.15. Два инвертора используются для получения значений  $\bar{y}$  и  $\bar{z}$ . Выход левого верхнего вентиля И соответствует  $xyz$ , а выход левого нижнего вентиля И дает  $\bar{y}\bar{z}$ . Выходы обоих этих вентилях служат входами вентиля ИЛИ. Поэтому на выходе вентиля ИЛИ будет  $xyz + \bar{y}\bar{z}$ . Наконец, выход этого вентиля связан со входом еще одного вентиля И, на другой вход которого поступает  $w$ . Окончательно логическая диаграмма рис. 4.15 описывается функцией

$$f(w, x, y, z) = w(xyz + \bar{y}\bar{z}).$$

Очевидно, можно провести и обратный процесс, а именно по заданному булевому выражению построить соответствующую логическую диаграмму.

Для того чтобы символы всех вентилях на диаграмме были одного размера, а также для того чтобы на схемах не появля-

лись слишком большие сгущения соединяющих линий, на входах вентилей И и ИЛИ часто используют обобщенные символы для многих входов, показанные на рис. 4.16.

#### 4.8. ЛОГИЧЕСКИЕ ВЕНТИЛИ ДРУГИХ ТИПОВ

В предыдущем разделе были введены логические вентили трех типов. Однако на логических диаграммах употребляются и другие вентили. На рис. 4.17 приведены графические обозначения вентилей наиболее распространенных типов. Нужно отметить, во-первых, что на рисунке представлено несколько

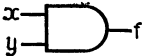


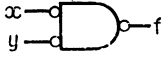
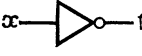









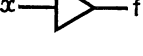
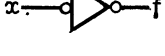
Название функции	Графическое обозначение вентилей		Булево выражение
И			$f = xy = (\overline{\overline{x} + \overline{y}})$
ИЛИ			$f = x + y = (\overline{\overline{x} \overline{y}})$
НЕ (инвертор)			$f = \overline{x}$
И - НЕ			$f = \overline{xy} = \overline{x} + \overline{y}$
ИЛИ - НЕ			$f = \overline{x + y} = \overline{x} \overline{y}$
Исключающее ИЛИ			$f = x \oplus y = \overline{x} \overline{y} + x y$ $= x \overline{y} + \overline{x} y$
Исключающее ИЛИ - НЕ (ЭКВИВАЛЕНТНОСТЬ)			$f = \overline{(x \oplus y)} = \overline{x} \overline{y} + x y$ $= x y + \overline{x} \overline{y}$
ИДЕНТИЧНОСТЬ (буферный усилитель)			$f = x$

Рис. 4.17. Наиболее распространенные вентили.

новых функций и, во-вторых, что каждая функция представлена парой графических символов. Здесь используется система обозначений, в которой кружочек обозначает инверсию.

### Система обозначений

Треугольник на рис. 4.17 обозначает **буферный усилитель**. С помощью такого устройства осуществляются разделение, усиление и восстановление сигналов, согласование импедансов в различных частях логической схемы. В логическом смысле его выходное значение совпадает с входным, т. е. реализуется функция ИДЕНТИЧНОСТЬ.

В принятых обозначениях кружочек на входе или выходе вентиля соответствует булевой операции отрицания. Таким образом, треугольник с кружочком на входе или на выходе, но не на том и другом одновременно оказывается вентилем НЕ, т. е. инвертором, а треугольник с кружочками и на входе, и на выходе — просто еще одним символом буферного усилителя, поскольку  $f = x = (\bar{x})$ .

Инвертирующие кружочки можно применить и к символам базисных сигналов вентилях И и ИЛИ. Например, булево выражение  $f = xy$  можно записать в виде  $f = (\bar{x} + \bar{y})$ . Это выражение объясняет, почему второе обозначение вентиля И на рис. 4.17 эквивалентно первому. Аналогично  $f = x + y$  можно записать как  $f = (\bar{x}\bar{y})$ ; это служит обоснованием второго символа для вентиля ИЛИ.

### Функция И-НЕ

На рис. 4.17 вводятся четыре новые логические функции: И-НЕ, ИЛИ-НЕ, ИСКЛЮЧАЮЩЕЕ ИЛИ, ИСКЛЮЧАЮЩЕЕ

Таблица 4.10. Определение операции И-НЕ

$x$	$y$	$f = (\overline{xy})$
0	0	1
0	1	1
1	0	1
1	1	0

ИЛИ-НЕ. Сначала рассмотрим И-НЕ. Ее определение как функции двух переменных  $x$  и  $y$  дано в табл. 4.10. Алгебраически ее можно записать в виде  $(\overline{xy})$ , а символически она изображается в виде вентиля И со входами  $x$  и  $y$  и последующего инвертора. Кроме того, поскольку  $(\overline{xy}) = \bar{x} + \bar{y}$ , операцию И-НЕ можно представить в виде вентиля ИЛИ, входные сигналы

которого подвергаются инвертированию. Обе эти интерпретации соответствуют приведенным на рис. 4.17 графическим обозначениям,

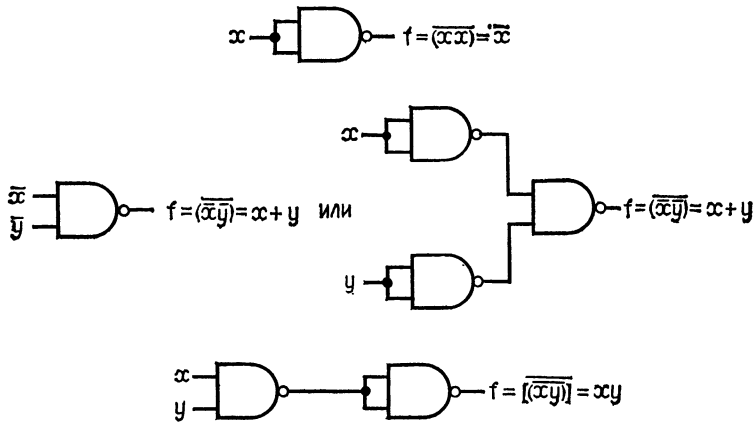


Рис. 4.18. Свойство универсальности вентилей И-НЕ.

В общем случае выход вентилей И-НЕ определяется как  $(x_1 x_2 \dots x_n) = \bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n$ , т. е. на его выходе будет логическая 1 тогда и только тогда, когда по крайней мере один вход равен логическому 0; в противном случае на выходе будет логический 0.

Одна из причин распространенности вентилей И-НЕ в логических схемах заключается в том, что эта операция **универсальна**. Универсальной называют такую операцию, с помощью которой можно реализовать все три базисные булевы операции И, ИЛИ и НЕ. Свойство универсальности поясняется на рис. 4.18, из которого следует, что любую комбинационную логическую схему можно реализовать, используя только вентили И-НЕ.

В одной из процедур построения по булевому выражению логической диаграммы, состоящей только из вентилей И-НЕ, используется алгебраическое определение вентилей И-НЕ, а именно:

$$f(x_1, x_2, \dots, x_n) = \overline{x_1 x_2 \dots x_n}.$$

Для иллюстрации рассмотрим выражение

$$f(w, x, y, z) = w + \bar{y}z + \bar{w}(x + y).$$

Применив закон де Моргана, запишем его в виде

$$f(w, x, y, z) = \{\bar{w}(\bar{y}z) \overline{\bar{w}(x + y)}\}.$$

Поскольку это общая алгебраическая форма вентилей И-НЕ со входами  $\bar{w}$ ,  $(\bar{y}z)$  и  $\overline{\bar{w}(x + y)}$ , можно построить схему, показанную на рис. 4.19, а. Далее, член  $(\bar{y}z)$  является алгебраической

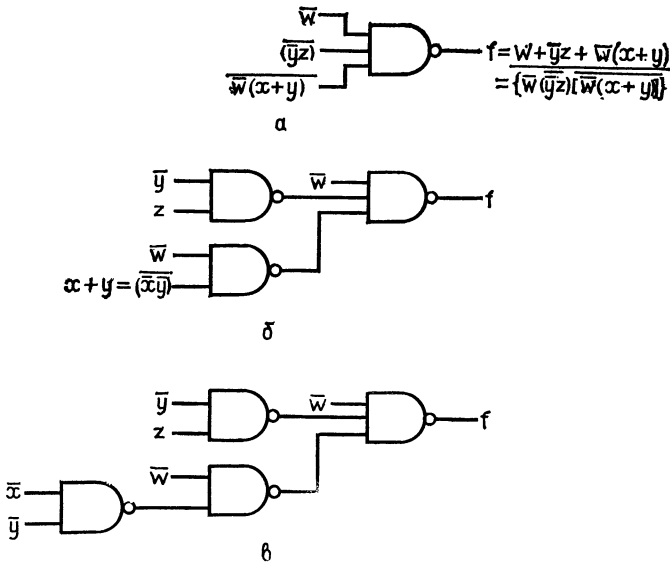


Рис. 4.19. Построение логической диаграммы, состоящей только из вентилях И-НЕ, по булевому выражению  $f(w, x, y, z) = w + yz + \bar{w}(x + y)$ .

формой для вентиля И-НЕ со входами  $\bar{y}$  и  $z$ , в то время как член  $[\bar{w}(x + y)]$  является алгебраической формой для вентиля И-НЕ со входами  $\bar{w}$  и  $(x + y)$ . Эти рассуждения приводят к схеме на рис. 4.19, б. Наконец, член  $x + y$  можно переписать в виде  $(\bar{x}\bar{y})$ , и мы приходим к схеме 4.19, в). В предположении что инвертированные значения входных переменных также доступны, схема 4.19, в только с помощью вентилях И-НЕ реализует булеву функцию

$$f(w, x, y, z) = w + \bar{y}z + \bar{w}(x + y).$$

Если значения входных переменных не могут поступать в инвертированном виде, то для их получения нужно воспользоваться вентилями НЕ.

Описанная выше процедура требует, чтобы в исходном выражении старшей операцией (т. е. операцией, выполняемой последней) была операция ИЛИ. Если это не так и старшей является операция И, то выражение инвертируется с использованием закона де Моргана. В результате старшей становится операция ИЛИ. Далее строится диаграмма из вентилях И-НЕ, а на выходе ставится вентиль НЕ или И-НЕ с объединенными входами.

# Функция ИЛИ-НЕ

Функция ИЛИ-НЕ — это еще одна функция, обладающая свойством универсальности. Ее определение как функции входных переменных  $x$  и  $y$  дано в табл. 4.11.

Алгебраически она записывается как  $\overline{(x + y)}$  и представляется вентилем ИЛИ со входами  $x$  и  $y$  с инвертором на выходе. Поскольку  $\overline{(x + y)} = \bar{x}\bar{y}$ , функцию ИЛИ-НЕ можно также представить в виде вентиля И с инверторами на входах. Обе эти интерпретации показаны на рис. 4.17.

В общем случае выход вентиля ИЛИ-НЕ определяется как

$\overline{(x_1 + x_2 + \dots + x_n)} = \bar{x}_1 \bar{x}_2 \dots \bar{x}_n$ , т. е. на выходе вентиль дает логическую 1 тогда и только тогда, когда на всех входах логические 0; в противном случае — на выходе логический 0. Свойство универсальности проиллюстрировано на рис. 4.20, где показано построение трех базисных булевых функций с помощью одних только вентилях ИЛИ-НЕ.

Благодаря универсальности вентилях ИЛИ-НЕ с их помощью можно реализовать любую комбинационную схему. В одной из процедур построения логической диаграммы из вентилях ИЛИ-НЕ по заданному булевому выражению используется алгебраическое определение вентиля, а именно соотношение  $f(x_1, x_2, \dots, x_n) = \overline{(x_1 + x_2 + \dots + x_n)}$ . Единственное требование, предъявляемое к форме исходного выражения, заключается в том, чтобы старшей операцией в выражении была операция

Таблица 4.11. Определение операции ИЛИ-НЕ

$x$	$y$	$f = \overline{(x + y)}$
0	0	1
0	1	0
1	0	0
1	1	0

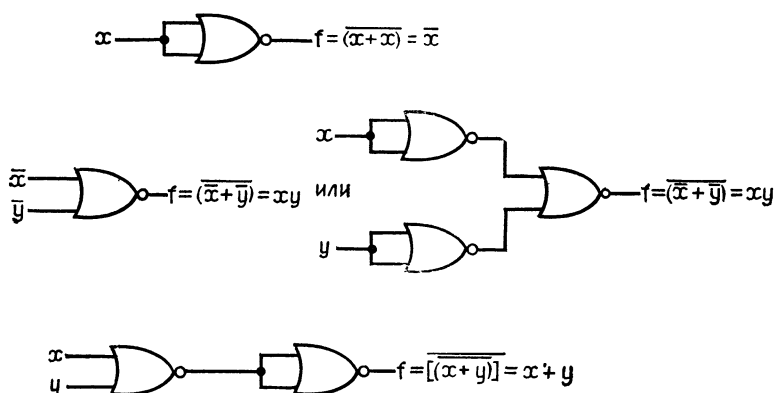


Рис. 4.20. Свойство универсальности вентиля ИЛИ-НЕ.

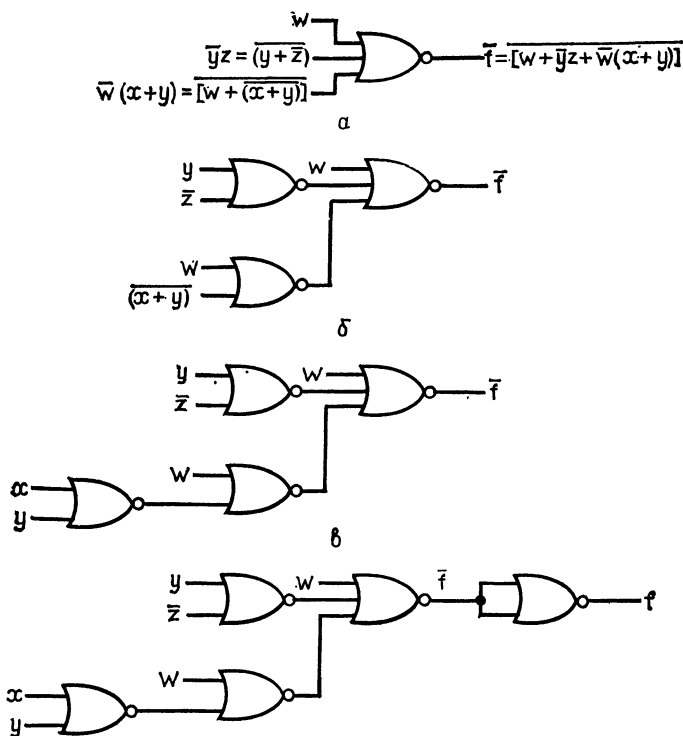


Рис. 4.21. Построение логической диаграммы, состоящей только из вентилях ИЛИ-НЕ, по булевому выражению  $f(w, x, y, z) = w + \bar{y}z + \bar{w}(x + y)$ .

И. Если это требование не выполнено и старшей является операция ИЛИ, то строится сначала диаграмма с вентилями ИЛИ-НЕ для инверсии выражения, а затем на выходе ставится вентиль НЕ или вентиль ИЛИ-НЕ с объединенными входами.

Чтобы проиллюстрировать построение логической диаграммы, снова рассмотрим выражение

$$f(w, x, y, z) = w + \bar{y}z + \bar{w}(x + y).$$

Поскольку старшими в этом выражении являются операции ИЛИ, оно инвертируется:

$$\bar{f}(w, x, y, z) = \overline{w + \bar{y}z + \bar{w}(x + y)}.$$

Для инвертированного выражения логическая диаграмма строится, как показано на рис. 4.21, а — в. Завершается построение диаграммы добавлением на выходе вентиля ИЛИ-НЕ с объединенными входами, как показано на рис. 4.21, г.

**Функции ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ**

Последние два типа вентилях, введенных на рис. 4.17, соответствуют функции ИСКЛЮЧАЮЩЕЕ ИЛИ и ее отрицанию — функции ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ. Операция ИСКЛЮЧАЮЩЕЕ ИЛИ часто обозначается символом  $\oplus$  и называется также сложением по модулю 2. По определению  $x \oplus y$  принимает значение логической 1 тогда и только тогда, когда или  $x$ , или  $y$ , но не оба сразу равны логической 1; в противном случае значение  $x \oplus y$  равно логическому 0.

Таблица 4.12. Определение операции ИСКЛЮЧАЮЩЕЕ ИЛИ

$x$	$y$	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Это определение приведено в табл. 4.12. По этой таблице легко установить справедливость соотношения

$$x \oplus y = \bar{x}y + x\bar{y}.$$

Для отрицания — ИСКЛЮЧАЮЩЕГО ИЛИ-НЕ — имеем

$$\overline{(x \oplus y)} = \overline{(\bar{x}y + x\bar{y})} = (x + \bar{y})(\bar{x} + y) = xy + \bar{x}\bar{y}.$$

Из этого уравнения следует, что  $\overline{(x \oplus y)}$  принимает значение логической 1 тогда и только тогда, когда  $x$  и  $y$  имеют одинаковые значения; в противном случае  $\overline{(x \oplus y)}$  равно логическому 0. По этой причине операцию ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ часто называют ЭКВИВАЛЕНТНОСТЬЮ.

В то время как вентили И-НЕ и ИЛИ-НЕ, как правило, бывают многовыходовыми, вентили для ИСКЛЮЧАЮЩЕГО ИЛИ и ЭКВИВАЛЕНТНОСТИ имеют обычно только по два входа.

Так же как для всех ранее рассмотренных функций, для ИСКЛЮЧАЮЩЕГО ИЛИ и ЭКВИВАЛЕНТНОСТИ на рис. 4.17 дано по два графических обозначения. Помня о том, что кружочек обозначает инверсию, легко показать корректность обоих вариантов,



# 5

## ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ

*Дж. Голт, Р. Пиммел*

### 5.1. ВВЕДЕНИЕ

**Последовательностной схемой** называется схема, выходные сигналы которой зависят не только от текущих значений входных сигналов, но и от последовательности значений входных сигналов в предыдущие моменты. По этой причине говорят, что последовательностные схемы обладают памятью.

### 5.2. ТРИГГЕРНЫЙ ЭЛЕМЕНТ

Триггер является основным последовательностным элементом. Этот термин может использоваться как для электронных схем, входными и выходными сигналами которой служат уровни напряжения  $L$  и  $H$ , так и для логического элемента с входными и выходными переменными, равными логическому  $0$  и логической  $1$ .

#### Триггерная схема

Триггерная схема представляет собой специальную электронную схему с двумя выходными сигналами  $Q$  и  $\bar{Q}$ , которая показана на рис. 5.1. Выходной сигнал  $Q$  считается **истинным**, а выходной сигнал  $\bar{Q}$  — **дополнительным** или **ложным**. Этим сигналам соответствует один из двух уровней напряжения и они дополняют друг друга, т. е. если  $Q$  соответствует высокому уровню напряжения  $H$ ,  $\bar{Q}$  должен соответствовать низкому уровню напряжения  $L$ , и наоборот. Поскольку выходные сигналы триггера постоянны до тех пор, пока они не будут изменены под воздействием входных сигналов, он имеет два устойчивых режима  $Q = H$  и  $\bar{Q} = L$  или  $Q = L$  и  $\bar{Q} = H$ . Эти два режима называются **состояниями**. Первое из них  $Q = H$  называется состоянием **установки**, второе — состоянием **сброса**. Говорят, что

триггер **установлен**, если он приведен в состояние установки, и, что он **сброшен**, если он приведен в состояние сброса. Вообще предполагается, что для построения триггеров используется положительная логика<sup>1)</sup>. Таким образом, состояниям установки и сброса ставятся в соответствие логические **состояния 1 и 0**.

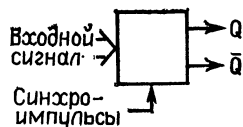


Рис. 5.1. Обозначение триггера.

Опишем четыре триггерные схемы типа *D*, *T*, *SR* и *JK*. Для каждого из них существует однозначное соответствие между входными сигналами и соответствующими переходами состояний. Это соответствие задается таблицей состояний, описываемых напряжениями. В таблице указывается напряжение на выходе *Q*, которое устанавливается после поступления сигнала синхронизации переключений для всех комбинаций входных сигналов как в состоянии установки, так и сброса. На рис. 5.2 приведены графические обозначения и таблицы состояний напряжений для триггеров *D*, *T*, *SR* и *JK*. В этих таблицах условиям до и после переключения соответствуют текущее и следующее состояние.

*D*-триггер имеет только один входной сигнал и его состояние определяется этим сигналом, т. е. триггер приводится в состояние сброса, когда входной сигнал имеет низкий уровень, и в состояние установки, когда входной сигнал имеет высокий уровень. Состояние *T*-триггера изменяется тогда, когда его единственный входной сигнал принимает высокий уровень, в противном случае триггер остается в том же состоянии. *SR*-триггер можно перевести в состояние установки или сброса путем подачи сигнала на соответствующий вход, т. е. схема оказывается в состоянии установки, когда на вход *S* подается сигнал высокого уровня, и в состоянии сброса, когда сигнал высокого уровня подается на вход *R*. Если на оба входа триггера подаются сигналы низкого уровня, то его состояние не меняется, но если на оба входа подаются сигналы высокого уровня, то состояние триггера будет неопределенным. *JK*-триггер аналогичен *SR*-триггеру, за исключением того, что, когда на оба входа пода-

<sup>1)</sup> В потенциальных цифровых схемах логический 0 и логическая 1 представляются двумя существенно различающимися уровнями электрического напряжения. В случае когда низкому напряжению ставят в соответствие логический 0, а высокому — логическую 1, говорят, что логика положительная. При использовании отрицательной логики за логический 0 принимают высокое напряжение, а за логическую 1 — низкое. В импульсных схемах одно из значений логического сигнала (0 или 1) определяется наличием импульса определенной длительности и амплитуды, а другое — отсутствием. При положительной логике отсутствие импульсов соответствует логическому 0, а наличие импульсов — логической 1. — *Прим. ред.*

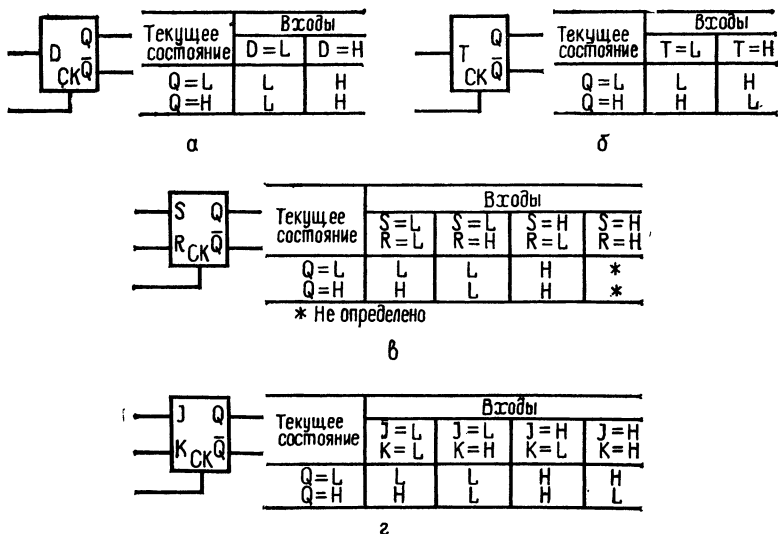


Рис. 5.2. Обозначение и таблица состояний напряжений для схемы *D*-триггера (а), схемы *T*-триггера (б), схемы *SR*-триггера (в), схемы *JK*-триггера (г).

ются сигналы высокого уровня, триггер изменяет состояния независимо от своего текущего состояния.

В зависимости от того, используется ли синхронизирующий сигнал или нет, триггер является асинхронным или синхронным. **Асинхронные триггеры** функционируют без синхронизирующих сигналов, и переход из одного состояния в другое начинается при изменении входных сигналов. **Синхронные триггеры** приводятся в действие синхронизирующим сигналом, так что воздействие входных сигналов и соответствующие переходы состояний происходят только во время определенной части синхронизирующего сигнала. Поэтому во время этой части синхронизирующего периода входные сигналы должны оставаться неизменными. Мы будем рассматривать синхронизирующий сигнал как периодический импульс; **положительный** и **отрицательный** фронты определяют два различных временных события, которые называют **активными фазами**. В триггерах, которые запускаются фронтом синхронизирующего сигнала, используется одна и та же активная фаза как для распознавания входных сигналов, так и для запуска переключения состояний. В триггерах типа «ведущий — ведомый» положительный фронт синхронизирующего сигнала используется для распознавания входных сигналов, а отрицательный фронт — для запуска переключения состояний. Название этих устройств отражает тот факт, что они состоят из двух внутренних триггерных секций, соединенных

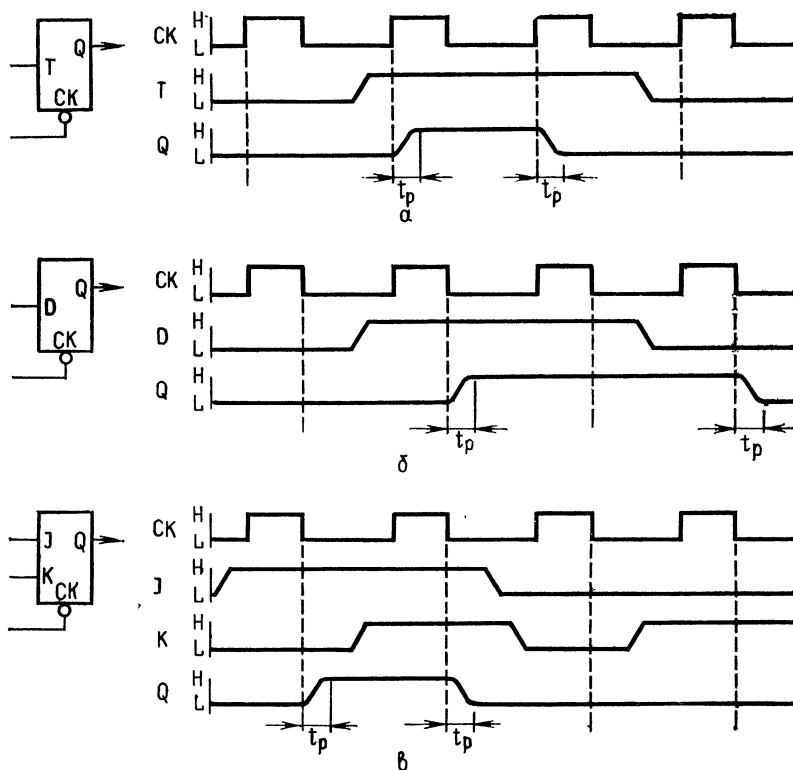


Рис. 5.3. Временные диаграммы схемы  $T$ -триггера, запускаемого передним фронтом (а), схема  $D$ -триггера, запускаемого задним фронтом (б), схема  $JK$ -триггера (в) типа «ведущий — ведомый».

каскадно. Первая секция — «ведущая» — распознает входные сигналы и осуществляет запуск переключения состояний положительным фронтом синхронизирующего импульса, вторая секция — «ведомая» — принимает выходные сигналы «ведущей» секции в качестве входных сигналов и осуществляет запуск переключения состояний отрицательным фронтом синхронизирующего импульса.

На рис. 5.3 показаны временные диаграммы  $T$ -триггера, переключаемого положительным фронтом импульса,  $D$ -триггера, переключаемого отрицательным фронтом импульса, и  $JK$ -триггера типа «ведущий — ведомый». Здесь предполагается, что в начальный момент времени триггеры находятся в состоянии сброса. На рис. 5.3, а входной сигнал  $T$  имеет низкий уровень при появлении положительных фронтов первого и последнего

синхронизирующих импульсов, поэтому триггер сохраняет свое состояние. При двух средних синхронизирующих импульсах входной сигнал  $T$  имеет высокий уровень, поэтому триггер изменяет свое состояние. Так же как в случае комбинационных дискретных схем, при переключении триггера имеет место временная задержка, называемая **задержкой распространения**  $t_p$ , которая равна времени, необходимому для того, чтобы выходной сигнал принял стабильные значения. Эта задержка является характеристикой рассматриваемого устройства; ее величина зависит от технологии производства. **Второй** важной временной характеристикой триггеров является **время установки**, которое равно интервалу времени до активной фазы синхронизирующего импульса, в течение которого входные сигналы должны иметь установившиеся значения. Она также зависит от технологии производства устройства.

Для  $D$ -триггера на рис. 5.3,б активной фазой синхронизирующего импульса является задний фронт импульса (отмечено кружочком на входе  $C$ ). Во время первого синхронизирующего импульса на входе  $D$  сигнал имеет низкий уровень, но, поскольку триггер уже находится в состоянии сброса, его состояние не меняется. Во время второго синхронизирующего импульса на входе  $D$  сигнал имеет высокий уровень и поэтому триггер переключается в состояние установки. Во время третьего синхронизирующего импульса, когда на входе  $D$  сигнал имеет высокий уровень, состояние триггера не изменяется, так как он уже находится в состоянии установки. Триггер переключается в состояние сброса во время четвертого синхронизирующего импульса.

На рис. 5.3,в изменение выходного сигнала начинается на отрицательном фронте синхронизирующего импульса (отмечено кружочком на входе  $C$ ). Однако на положительном фронте синхронизирующего импульса входные сигналы не вызывают таких изменений, так как устройство является триггером типа «ведущий—ведомый». Во время первого синхронизирующего импульса сигнал на входе  $J$  имеет высокий уровень, а сигнал на входе  $K$  — низкий уровень, так что триггер переключается в состояние установки. Оба входных сигнала имеют высокий уровень во время второго синхронизирующего импульса, и поэтому триггер изменяет свое состояние. Во время третьего синхронизирующего импульса состояние триггера не изменяется, поскольку оба входных сигнала имеют низкий уровень. Во время четвертого импульса входные сигналы соответствуют переключению триггера в состояние сброса, однако из-за того, что он уже находится в этом состоянии, его состояние не изменяется.

### Триггерный логический элемент

При сопоставлении уровней напряжения на входах и выходах триггерной схемы с двоичными значениями будем использовать положительную логику. Логическое поведение триггера описывается логическими таблицами состояний или просто таблицами состояний (табл. 5.1). Таблицы состояний триггера аналогичны таблицам уровней напряжений, приведенным на рис. 5.2; в них указано двоичное значение  $Q$  после появления синхронизирующего импульса для всех комбинаций входных сигналов и обоих начальных состояний ( $Q = 0$  и  $Q = 1$ ). Таблицы состояний получаются из соответствующих таблиц напряжений на рис. 5.2 путем замены  $L$  на 0 и  $H$  — на 1.

При проектировании последовательностных схем мы пользуемся таблицами переходов, являющимися другим описанием поведения триггера. **Таблица переходов**, которую можно получить из таблицы состояний, определяет входное условие, необходимое для каждого возможного перехода состояний. В табл. 5.1 приведены также таблицы переходов для триггеров четырех типов.

На примере  $JK$ -триггера проиллюстрируем процедуру преобразования таблицы состояний в таблицу переходов. Первый переход 0 в 0 возникает при  $J = 0$  и  $K = 0$ , что соответствует отсутствию входных сигналов, или же при  $J = 0$  и  $K = 1$ , что является входным условием для перехода триггера в состояние сброса. Так как  $K$  может быть как 0, так и 1, его значение может считаться неопределенным. Таким образом, для перехода 0 в 0 требуется, чтобы  $J = 0$  и  $K = d$ . Для перехода 0 в 1 необходимо, чтобы  $J = 1$  и  $K = 0$ , что является входным условием для перехода триггера в состояние установки, или же  $J = 1$  и  $K = 1$ , что представляет комбинацию входных сигналов, соответствующую изменению состояния триггера. Поэтому для этого перехода требуемыми входными сигналами будут  $J = 1$  и  $K = d$ . Входные требования для остальных двух переходов определяются аналогичным образом.

### 5.3. ТАБЛИЦЫ И ДИАГРАММЫ СОСТОЯНИЙ

При описании поведения последовательностных схем, содержащих несколько триггеров, будем пользоваться понятием состояния. В начале данной главы было отмечено, что триггер имеет два устойчивых состояния: состояние сброса при  $Q = 0$  и состояние установки при  $Q = 1$ . В схеме с  $N$  триггерами состояния характеризуются  $N$ -разрядным двоичным словом, каждый разряд которого ассоциируется с одним из триггеров. Так как существуют  $2^N$  различных комбинаций  $N$ -разрядного слова, имеются  $2^N$  устойчивых состояний.

Таблица 5.1. Таблицы состояний и таблицы переходов для четырех триггерных логических элементов ( $d$  — недоопределенное состояние)

D-триггер					
Таблица состояний			Таблица переходов		
Текущее состояние	Вход		Текущее состояние	Следующее состояние	D
	D=0	D=1			
Q = 0	0	1	0	0	0
Q = 1	0	1	0	1	1
			1	0	0
			1	1	1

T-триггер					
Таблица состояний			Таблица переходов		
Текущее состояние	Вход		Текущее состояние	Следующее состояние	T
	T=0	T=1			
Q = 0	0	1	0	0	0
Q = 1	1	0	0	1	1
			1	0	1
			1	1	0

SR-триггер								
Таблица состояний			Таблица переходов					
Текущее состояние	Входы SR				Текущее состояние	Следующее состояние	S	R
	00	01	10	11				
Q = 0	0	0	1	*	0	0	0	d
Q = 1	1	0	1	*	0	1	1	0
					1	0	0	1
					1	1	d	0

JK-триггер								
Таблица состояний			Таблица переходов					
Текущее состояние	Входы JK				Текущее состояние	Следующее состояние	J	K
	00	01	10	11				
Q = 0	0	0	1	1	0	0	0	d
Q = 1	1	0	1	0	0	1	1	d
					1	0	d	1
					1	1	d	0

\* не определено

### Символическое обозначение

Часто оказывается удобным для каждой допустимой комбинации входных и выходных сигналов и состояний вместо двоичных кодовых наборов использовать символическое кодовое представление. Для представления допустимых комбинаций входных и выходных сигналов и состояний будем использовать соответственно символы  $I_0, I_1, I_2, \dots, R_0, R_1, R_2$  и  $S_0, S_1, S_2, \dots$ . Так же как триггер характеризуется таблицей состояний, по которой определяется следующее состояние для всех комбинаций входных сигналов и текущих состояний, последовательная схема, построенная из триггеров, характеризуется **таблицей состояний**, по которой определяются текущий выходной сигнал и следующее состояние для всевозможных комбинаций входных сигналов и текущих состояний. Информация таблицы состояний схемы также может быть представлена в виде **диаграммы состояний схемы**, или **диаграммы состояний**, на которой указаны текущий выходной сигнал и следующее состояние для всевозможных комбинаций входных сигналов и текущих состояний. На рис. 5.4 приведены таблица состояний и диаграмма состояний для последовательностной схемы. Эта схема имеет два допустимых входных сигнала ( $I_0$  и  $I_1$ ), три допустимых выходных сигнала ( $R_0, R_1$  и  $R_2$ ) и четыре допустимых состояния ( $S_0, S_1, S_2$  и  $S_3$ ).

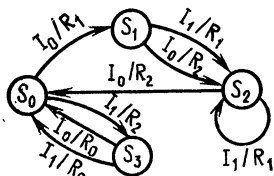
В таблице состояний на рис. 5.4 текущие входные сигналы указаны в верхней строке, а текущие состояния — в крайнем левом столбце. Внутренние элементы таблицы указывают следующее состояние и текущий выходной сигнал для каждой комбинации входного сигнала и текущего состояния. Например, если схема находится в состоянии  $S_0$  и входным сигналом является  $I_0$ , то текущим входным сигналом будет  $R_1$ , а следующим состоянием —  $S_1$ . Аналогично если входным сигналом является  $I_1$  и текущее состояние есть  $S_0$ , то текущим выходным сигналом будет  $R_2$ , а следующим состоянием —  $S_3$ . Другим примером является случай, когда текущее состояние есть  $S_3$ , а текущим выходным сигналом является  $R_0$  и следующее состояние есть  $S_0$  независимо от значений входного сигнала. Наконец, для текущего состояния  $S_2$  и входного сигнала  $I_1$  состояние сохраняется, а текущим входным сигналом будет  $R_1$ .

На диаграмме состояний, приведенной на рис. 5.4, б, каждое состояние представляется кружком, а каждому входному сигналу соответствует ориентированная линия, стрелка которой указывает следующее состояние. Для каждой ориентированной линии указаны значения входного сигнала и текущего выходного сигнала. В этом примере из  $S_0$  выходят две дуги, по одной



Текущее состояние	Входы	
	$I_0$	$I_1$
$S_0$	$S_1/R_1$	$S_3/R_2$
$S_1$	$S_2/R_2$	$S_2/R_1$
$S_2$	$S_0/R_2$	$S_2/R_1$
$S_3$	$S_0/R_0$	$S_0/R_0$

а



б

Рис. 5.4. Таблица состояний (а) и диаграмма состояний (б).

для каждого допустимого входного сигнала. Если входным сигналом является  $I_0$ , то выходным сигналом будет  $R_1$ , а следующим состоянием —  $S_1$ ; если же входным сигналом является  $I_1$ , то выходным сигналом будет  $R_2$ , а следующим состоянием —  $S_3$ . Для состояния  $S_3$  следующим состоянием будет  $S_0$ , а выходным сигналом —  $R_0$  независимо от значения входного сигнала. В то же время для состояния  $S_2$  и входного сигнала  $I_1$  будущее состояние совпадает с  $S_2$ , и выходным сигналом является  $R_1$ .

Поскольку таблица состояний и диаграмма состояний дают одну и ту же информацию, их можно преобразовать друг в друга. При преобразовании таблицы состояний в диаграмму состояний каждое состояние из крайнего левого столбца представляется кружком. Каждый элемент таблицы преобразуется в отрезок ориентированной линии, соединяющей кружки. Например, на рис. 5.4, а для текущего состояния  $S_0$

имеются два элемента, по одному для каждого значения входного сигнала, и поэтому на диаграмме состояний две ориентированные линии выходят из кружка  $S_0$ . Для входного сигнала  $I_0$  направленная линия входит в кружок  $S_1$ , и значения входного и выходного сигналов записываются как  $I_0/R_1$ . Для входного сигнала  $I_1$  эта линия входит в кружок для состояния  $S_3$ , а значения входного и выходного сигналов записываются как  $I_1/R_2$ . Точно так же можно отобразить остальные три текущих состояния.

Процедура преобразования диаграммы состояний в таблицу состояний выполняется аналогично. Сначала определяется структура таблицы, т. е. строка для каждого состояния на диаграмме состояний и столбец для каждого отдельного значения входного сигнала. Затем каждая направленная линия преобразуется в некоторый элемент таблицы. Например, для  $S_0$  на рис. 5.4, б направленная линия, соответствующая  $I_0$ , определяет выходной сигнал  $R_1$  и следующее состояние  $S_1$ ; эта информация записывается в верхнем левом элементе таблицы в виде  $S_1/R_1$ . Направленная линия, соответствующая  $I_1$ , определяет выходной сигнал  $R_2$  и следующее состояние  $S_3$ . Эта информация содержится в верхнем правом элементе таблицы в виде  $S_3/R_2$ . Остальные три состояния можно рассмотреть таким же образом.

# Последовательности состояний

Чтобы показать, как таблица состояний и диаграмма состояний схемы характеризуют поведение последовательностной схемы, рассмотрим реакцию схемы, определенной на рис. 5.4, на входную последовательность  $I_1, I_1, I_0, I_1$  и  $I_0$  при начальном состоянии  $S_1$ . Во время первого синхронизирующего импульса входным сигналом является  $I_1$ , а текущим состоянием —  $S_1$ . Из таблицы или диаграммы состояний на рис. 5.4 видно, что для комбинации  $S_1$  и  $I_1$  выходным сигналом во время синхронизирующего импульса является  $R_1$ , а следующим состоянием после синхронизирующего импульса будет  $S_2$ , как показано на рис. 5.5, а. При втором синхронизирующем импульсе входным сигналом является  $I_1$ , а новым текущим состоянием будет  $S_2$ . Из таблицы или диаграммы состояний видно, что при комбинации  $I_1$  и  $S_2$  выходным сигналом является  $R_1$ , а следующим состоянием будет  $S_2$ . При третьем синхронизирующем импульсе задан входной сигнал  $I_0$ , а текущим состоянием является  $S_2$ . Этой комбинации соответствует выход  $R_2$  и новое состояние  $S_0$ . Выходные сигналы и переходы состояний для четвертого и пятого синхронизирующих импульсов могут быть рассмотрены аналогичным образом.

В таблице на рис. 5.5, б в более компактной форме представлены последовательность состояний и выходные сигналы, порожденные заданной последовательностью входных сигналов. Каждую вертикальную линию можно рассматривать в качестве синхронизирующего импульса, а элементы предыдущего столбца указывают значения входного, выходного сигналов и состояние во время этого синхронизирующего импульса. Например, во время первого синхронизирующего импульса допусти-



Входная последовательность	$I_1$	$I_1$	$I_0$	$I_1$	$I_0$
Последовательность состояний	$S_1$	$S_2$	$S_2$	$S_0$	$S_3$
Выходная последовательность	$R_1$	$R_1$	$R_2$	$R_2$	$R_0$

б

Рис. 5.5. Входная и выходная последовательности и последовательность состояний для схемы, описанной на рис. 5.4.

мыми значениями являются  $I_1$ ,  $S_1$  и  $R_1$ , при втором синхронизирующем импульсе —  $I_1$ ,  $S_2$  и  $R_1$ .

**Пример.** Пусть последовательностная схема определена таблицей состояний, приведенной ниже. Найдем последовательности выходных сигналов и состояний для входной последовательности  $I_0, I_2, I_2, I_3$  и  $I_1$ . Пусть  $S_2$  — начальное состояние.

Текущее состояние	Входные сигналы			
	$I_0$	$I_1$	$I_2$	$I_3$
$S_0$	$S_0/R_0$	$S_1/R_1$	$S_2/R_0$	$S_1/R_0$
$S_1$	$S_0/R_0$	$S_1/R_0$	$S_1/R_0$	$S_0/R_0$
$S_2$	$S_1/R_2$	$S_1/R_0$	$S_2/R_0$	$S_1/R_0$

<i>Решение</i>					
Входная последовательность	$I_0$	$I_2$	$I_2$	$I_3$	$I_1$
Последовательность состояний	$S_2$	$S_1$	$S_1$	$S_1$	$S_0$
Выходная последовательность	$R_2$	$R_0$	$R_0$	$R_0$	$R_1$

### Двоично-кодированные таблицы и диаграммы состояний

Таблицы и диаграммы состояний можно построить, используя для представления значений входных и выходных сигналов состояний двоичные коды. На рис. 5.6, *a* приведена одна группа кодовых наборов для последовательностной схемы, заданной таблицей<sup>1)</sup> и диаграммой состояний, которые приведены на рис. 5.4.

Для кодирования двух значений входного сигнала требуется лишь одна двоичная переменная  $X_0$ . Существуют два варианта выбора кода: положить  $X_0 = 0$  для  $I_0$  и  $X_0 = 1$  для  $I_1$  или же  $X_0 = 0$  для  $I_1$  и  $X_0 = 1$  для  $I_0$ . На рис. 5.6, *a* выбран первый из них.

Для кодирования трех значений выходного сигнала необходимы две переменные  $Z_1$  и  $Z_0$ . Существует несколько вариантов выбора кода, на которые накладывается лишь одно ограничение, выражающее требование однозначности кодового набора для каждого значения. В коде на рис. 5.6, *a* значению  $R_0$  присваивается кодовый набор 11, значению  $R_1$  — кодовый набор

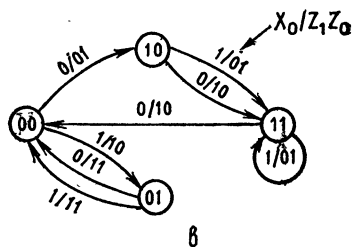
<sup>1)</sup> Во всех кодированных таблицах состояний элементы, соответствующие следующему состоянию и выходной переменной, упорядочены по убывающим индексам слева направо.

Код входа		Код выхода		Код состояний	
Обозначение	$X_0$	Обозначение	$Z_1$ $Z_0$	Обозначение	$Q_1$ $Q_0$
$I_0$	0	$R_0$	1 1	$S_0$	0 0
$I_1$	1	$R_1$	0 1	$S_1$	1 0
		$R_2$	1 0	$S_2$	1 1
				$S_3$	0 1

а

Текущее состояние		$X_0$	
$Q_2$	$Q_0$	0	1
0	0	10/01	01/10
1	0	11/10	11/01
1	1	00/10	11/01
0	1	00/11	00/11

б



в

Рис. 5.6. а — Множество кодовых наборов для схемы, описанной на рис. 5.4. б — Результирующая таблица кодированных состояний. в — Диаграмма состояний.

01, значению  $R_2$  — кодовый набор 10, кодовый набор 00 не используется. Эти кодовые наборы задают значение двоичным переменным  $Z_1$  и  $Z_0$ , где  $Z_1$  — старший разряд. При кодировании четырех состояний необходимо использовать две двоичные переменные; это означает, что данная схема содержит два триггера, представленных символами  $Q_1$  и  $Q_0$ . Опять имеется несколько вариантов выбора кода; в коде на рис. 5.6, а состоянию  $S_0$  присваивается кодовый набор 00, состоянию  $S_1$  — кодовый набор 10, состоянию  $S_2$  — кодовый набор 11 и состоянию  $S_3$  — кодовый набор 01. Эти кодовые наборы определяют значения  $Q_1$  и  $Q_0$ , где  $Q_1$  — старший разряд.

Коды, представленные на рис. 5.6, *а*, выбраны произвольно. Существуют методы определения кодов, которые по тому или иному критерию минимизируют сложность схемы. Мы не будем рассматривать этот аспект проектирования последовательностной схемы и ограничимся определением однозначных, но произвольных кодов для каждого состояния и условия.

На рис. 5.6, *б, в* показаны кодированные таблица состояний и диаграмма состояний, полученные из вариантов, приведенных на рис. 5.4. Эти кодированные варианты строятся путем замены каждого символа на кодовый набор, присвоенный ему. Например,  $I_0$  заменяется на 0,  $R_0$  — на 11,  $S_0$  — на 00.

#### 5.4. ПРЕОБРАЗОВАНИЕ ТАБЛИЦЫ СОСТОЯНИЙ В ЛОГИЧЕСКУЮ ДИАГРАММУ

В этом разделе будет описан метод преобразования кодированной таблицы состояний для последовательностной схемы в эффективную логическую диаграмму с использованием общей структурной модели, приведенной на рис. 5.7. В этой модели последовательностная схема разделяется на комбинационную и триггерную подсхемы. В ней также используются четыре различные группы переменных: на входах схемы  $X$ , на выходах схемы  $Z$ , на входах триггеров  $Y$ , на выходах триггеров  $Q$ . Среди этих сигналов  $X$  и  $Q$  являются входными сигналами комбинационной подсхемы, а  $Z$  и  $Y$  — выходными сигналами этой подсхемы.

Рассмотрим кодированную таблицу состояний, приведенную на рис. 5.8, *а*, и построим для нее логическую диаграмму, используя  $JK$ -триггеры. Согласно таблице состояний, имеем одну входную переменную  $X_0$ , одну выходную переменную  $Z_0$  и два выхода триггеров  $Q_1$  и  $Q_0$ . Поскольку используются  $JK$ -триггеры,  $Y$  должно содержать четыре переменные  $J_1$ ,  $K_1$ ,  $J_0$  и  $K_0$ . На рис. 5.8, *б* показана общая структурная модель, на кото-

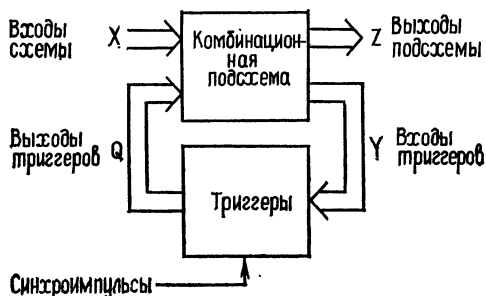
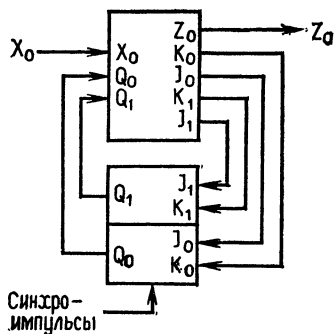


Рис. 5.7. Общая структурная модель последовательностной схемы.

Текущее состояние		$X_0$	
$Q_1$	$Q_0$	0	1
0	0	00/0	10/1
0	1	01/1	11/0
1	0	10/0	01/0
1	1	11/1	00/0

 $Q_1 Q_0 / Z_0$ 

а



б

Вход	Текущее состояние		Выход	Вход	Текущее состояние		Следующее состояние		Входы триггеров			
	$Q_1$	$Q_0$			$Q_1$	$Q_0$	$Q_1$	$Q_0$	$J_1$	$K_1$	$J_0$	$K_0$
$X_0$			$Z_0$	$X_0$								
0	0	0	0	0	0	0	0	0	0	d	0	d
0	0	1	1	0	0	1	0	1	0	d	d	0
0	1	0	0	0	1	0	1	0	d	0	0	d
0	1	1	1	0	1	1	1	1	d	0	d	0
1	0	0	1	1	0	0	1	0	1	d	0	d
1	0	1	0	1	0	1	1	1	1	d	d	0
1	1	0	1	1	1	0	0	1	d	1	1	d
1	1	1	0	1	1	1	0	0	d	1	d	1

в

г

Рис. 5.8. а — Таблица состояний. б — Структурная модель. в — Таблица истинности выхода. г — Расширенная таблица истинности для последовательностной схемы.

рой указаны эти переменные. Из рисунка видно, что построение логической диаграммы для данной последовательностной схемы сводится в основном к определению действительной логической диаграммы для комбинационной подсхемы. На первом шаге этой процедуры по информации, содержащейся в таблице состояний, получается таблица истинности для указанной комбинационной подсхемы.

В таблице состояний указаны значения  $Z_0$  для всех комбинаций  $X_0$ ,  $Q_1$  и  $Q_0$ . Таким образом, таблица истинности для  $Z_0$  получается просто посредством записи этой информации в более стандартной форме. На рис. 5.8, в приведена полученная **таблица истинности выхода**. Всем комбинациям  $X_0$ ,  $Q_1$  и  $Q_0$  присвоены упорядоченные двоичные значения. Каждая строка соответствует элементу таблицы состояний. Например, первая строка соответствует верхнему левому элементу, в котором

входной сигнал равен 0, а текущим состоянием является 00. В таблице состояний выходной сигнал для этой комбинации равен 0, который находится в первой строке таблицы истинности столбца  $Z_0$ . Вторая строка соответствует элементу, в котором входной сигнал равен 0, а текущее состояние есть 01. В таблице состояний выходной сигнал для данной комбинации равен 1; он также находится в таблице истинности. Эта процедура повторяется до тех пор, пока не будут определены все элементы таблицы истинности выхода.

Хотя в таблице состояний значения  $J_1$ ,  $K_1$ ,  $J_0$  и  $K_0$  явно не указываются, но в ней задано следующее состояние  $Q_1$  и  $Q_0$ , по которому можно определить значения входных сигналов триггеров. Для получения этой информации воспользуемся расширенной таблицей истинности с целью преобразования ее в желаемую форму.

В **расширенной таблице истинности** для этой задачи (рис. 5.8,2) указаны комбинации значений входных сигналов и текущих состояний и для каждой комбинации приведены значения соответствующих следующих состояний и входных сигналов триггеров, обеспечивающих данный переход. В этой таблице всем комбинациям  $X_0$ ,  $Q_1$  и  $Q_0$  присвоены упорядоченные двоичные значения. Элементы в столбцах следующих состояний взяты из таблиц состояний схемы с учетом значений  $Z_0$ , приведенных в таблице истинности выхода. Например, последняя строка в расширенной таблице истинности соответствует элементу таблицы состояний, в котором входной сигнал равен 1, а текущим состоянием является 11. Для этой комбинации следующим состоянием будет 00, которое заносится в последнюю строку расширенной таблицы истинности. Предпоследняя строка в расширенной таблице истинности соответствует элементу таблицы состояний, в котором входным сигналом является 1, текущее состояние есть 10. Для этой комбинации следующим состоянием будет 01. Эта процедура повторяется до тех пор, пока не будут определены все следующие состояния для расширенной таблицы истинности.

Теперь расширенная таблица истинности содержит текущие и следующие состояния для всех входных комбинаций. Следующим шагом будет нахождение значений входных сигналов триггеров для осуществления любого перехода состояний для каждого триггера. Для завершения построения этой таблицы воспользуемся таблицей переходов триггера (табл. 5.1), в которой приведены значения сигналов на входах  $J$  и  $K$  для всех возможных переходов. В первой строке расширенной таблицы истинности  $Q_1$  выполняет переход 0 в 0; из таблицы переходов  $JK$ -триггера видно, что этот переход осуществляется при  $J_1 = 0$  и  $K_1 = d$ . Поэтому данные значения заносятся в первую строку

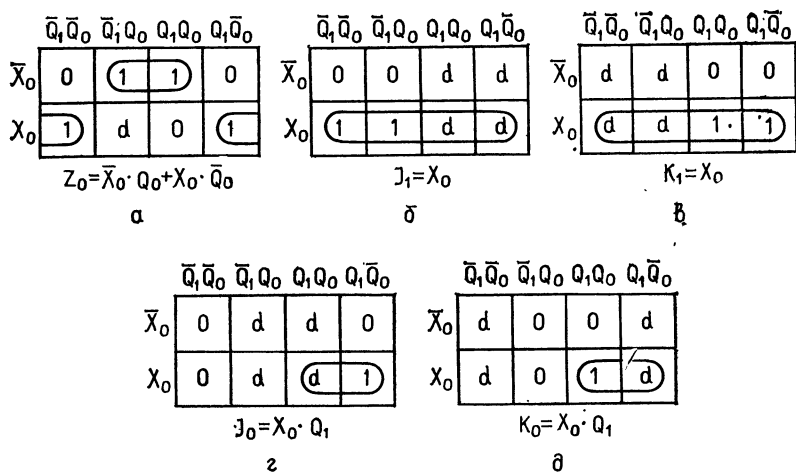


Рис. 5.9. Карты Карно для последовательной схемы, определенной на рис. 5.8.

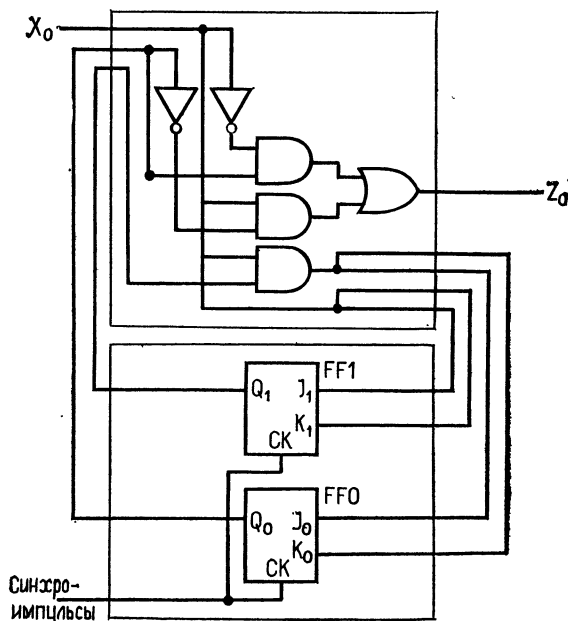


Рис. 5.10. Логическая диаграмма для последовательной схемы, определенной на рис. 5.8.



расширенной таблицы истинности. В последней строке  $Q_1$  выполняет переход 1 в 0 и поэтому  $J_1 = d$  и  $K_1 = 1$ . Эти значения заносятся в расширенную таблицу истинности. Процедура повторяется до тех пор, пока не будут определены  $J_1$  и  $K_1$  для всех строк расширенной таблицы истинности. Затем указанная процедура применяется для определения  $J_0$  и  $K_0$  на основании переходов  $Q_0$ .

Таблица истинности выхода (рис. 5.8, в) и расширенная таблица истинности (рис. 5.8, г) объединяются с целью определения поведения комбинационной подсистемы. Для получения минимизированной логической диаграммы используются карты

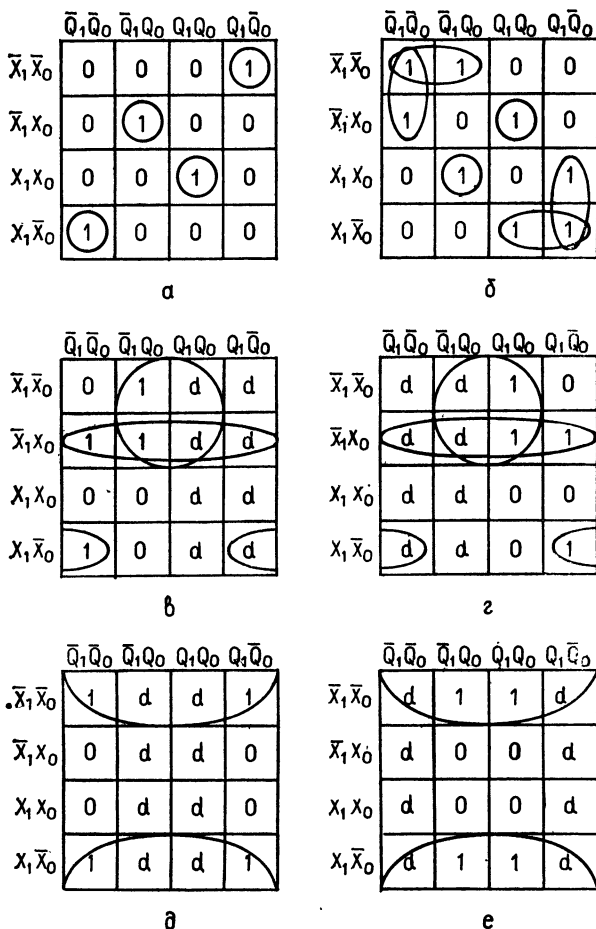


Рис. 5.11. Карты Карно для  $Z_1(a)$ ,  $Z_0(b)$ ,  $J_1(в)$ ,  $K_1(г)$ ,  $J_0(д)$ ,  $K_0(е)$ .

Карно для  $Z_0$ ,  $J_1$ ,  $K_1$ ,  $J_0$  и  $K_0$ ; в результате получается минимальная сумма произведений для каждой переменной. Эти карты и процесс минимизации иллюстрируются на рис. 5.9. На рис. 5.10 приведена логическая диаграмма с вентилями И, ИЛИ и НЕ, реализующая эти уравнения. Из этого рисунка видно, что  $X_0$  и  $Z_0$  являются лишь внешними сигналами, а входные и выходные сигналы триггеров — внутренними сигналами.

**Пример.** Для таблицы состояний, приведенной ниже, нужно получить все минимальные суммы произведений, необходимые для построения логической диаграммы с помощью  $JK$ -триггеров.

Текущее состояние $Q_1 Q_0$	$X_1 X_0$			
	00	01	10	11
00	01/01	10/01	11/10	00/00
01	10/01	11/10	00/00	01/01
10	11/10	00/00	01/01	10/01
11	00/00	01/01	10/01	11/10

**Решение.** Сначала построим стандартную и расширенную таблицы истинности. В дальнейшем используем таблицу переходов  $JK$ -триггера.

$X_1 X_0$		Текущее состояние $Q_1 Q_0$		$Z_1 Z_0$	
		$Q_1$	$Q_0$	$Z_1$	$Z_0$
0 0		0	0	0	1
		0	1	0	1
		1	0	1	0
		1	1	0	0
0 1		0	0	0	1
		0	1	1	0
		1	0	0	0
		1	1	0	1
1 0		0	0	1	0
		0	1	0	0
		1	0	0	1
		1	1	0	1
1 1		0	0	0	0
		0	1	0	1
		1	0	0	1
		1	1	1	0

$X_1 X_0$		Текущее состояние $Q_1 Q_0$		Следующее состояние $Q_1 Q_0$		$J_1 K_1 J_0 K_0$			
		$Q_1$	$Q_0$	$Q_1$	$Q_0$	$J_1$	$K_1$	$J_0$	$K_0$
0 0		0	0	0	1	0	$d$	1	$d$
		0	1	1	0	1	$d$	$d$	1
		1	0	1	1	$d$	0	1	$d$
		1	1	0	0	$d$	1	$d$	1
0 1		0	0	1	0	1	$d$	0	$d$
		0	1	1	1	1	$d$	$d$	0
		1	0	0	0	$d$	1	0	$d$
		1	1	0	1	$d$	1	$d$	0
1 0		0	0	1	1	1	$d$	1	$d$
		0	1	0	0	0	$d$	$d$	1
		1	0	0	1	$d$	1	1	$d$
		1	1	1	0	$d$	0	$d$	1
1 1		0	0	0	0	0	$d$	0	$d$
		0	1	0	1	0	$d$	$d$	0
		1	0	1	0	$d$	0	0	$d$
		1	1	1	1	$d$	0	$d$	0

Чтобы получить формулы для минимальных сумм произведений, строятся и используются карты Карно (рис. 5.11).

$$Z_1 = \bar{X}_1 \cdot \bar{X}_0 \cdot Q_1 \cdot \bar{Q}_0 + \bar{X}_1 \cdot X_0 \cdot \bar{Q}_1 \cdot Q_0 + X_1 \cdot X_0 \cdot Q_1 \cdot Q_0 + \\ + X_1 \cdot \bar{X}_0 \cdot \bar{Q}_1 \cdot \bar{Q}_0,$$

$$Z_0 = \bar{X}_1 \cdot \bar{X}_0 \cdot \bar{Q}_1 + \bar{X}_1 \cdot \bar{Q}_1 \cdot \bar{Q}_0 + X_1 \cdot \bar{X}_0 \cdot Q_1 + X_1 \cdot Q_1 \cdot \bar{Q}_0 + \\ + \bar{X}_1 \cdot X_0 \cdot Q_1 \cdot Q_0 + X_1 \cdot X_0 \cdot \bar{Q}_1 \cdot Q_0,$$

$$J_1 = \bar{X}_1 \cdot X_0 + \bar{X}_1 \cdot Q_0 + X_1 \cdot \bar{X}_0 \cdot \bar{Q}_0,$$

$$K_1 = \bar{X}_1 \cdot X_0 + \bar{X}_1 \cdot Q_0 + X_1 \cdot \bar{X}_0 \cdot \bar{Q}_0,$$

$$J_0 = \bar{X}_0,$$

$$K_0 = \bar{X}_0.$$

Логическая диаграмма может быть построена непосредственно по этим уравнениям.

### 5.5. ПРЕОБРАЗОВАНИЕ ЛОГИЧЕСКОЙ ДИАГРАММЫ В ТАБЛИЦУ СОСТОЯНИЙ

В данном разделе описывается метод преобразования логической диаграммы последовательностной схемы в таблицу состояний. Это преобразование, обратное по отношению к рассмотренному в предыдущем разделе, является важным шагом при анализе последовательностных схем.

На рис. 5.12 показана логическая диаграмма для последовательностной схемы. Она имеет одну входную переменную  $X_0$ , одну выходную переменную  $Z_0$  и два  $T$ -триггера FF0 и FF1. Комбинационная схема описывается посредством вентилей И, ИЛИ и НЕ. Выражения для  $Z_0$  и входных сигналов двух триггеров  $T_0$  и  $T_1$  можно записать через входные сигналы  $X_0$ ,  $Q_1$  и  $Q_0$  комбинационных подсхем:

$$Z_0 = \bar{X}_0 \cdot Q_1 \cdot Q_0 + X_0 \cdot \bar{Q}_1 \cdot \bar{Q}_0, \quad (5.1)$$

$$T_1 = X_0 \cdot \bar{Q}_0 + \bar{X}_0 \cdot Q_0, \quad (5.2)$$

$$T_0 = 1. \quad (5.3)$$

Таблица 5.2 содержит таблицу истинности выхода  $Z_0$ , построенную по формуле (5.1). Первые три столбца  $X_0$ ,  $Q_1$  и  $Q_0$  содержат всевозможные комбинации входных сигналов комбинационной подсхемы. Как и выше, входным сигналам присвоены упорядоченные двоичные значения. Значения  $Z_0$  получены путем представления выражения (5.1) в виде  $Z_0 = m_3 + m_4$  и подстановки 1 в строки, соответствующие минтермам  $m_3$  и  $m_4$ , и 0 во все остальные строки.

Таблица 5.2 включает также расширенную таблицу истинности для входных сигналов триггеров. Значения входного сигнала и текущего состояния пронумерованы так же, как и ранее.

Таблица 5.2. Таблица истинности выхода, расширенная таблица истинности и таблица состояний для последовательностной схемы, заданной на рис. 5.12

Таблица истинности выхода				
Вход $X_0$	Текущее состояние		Выход $Z_0$	
	$Q_1$	$Q_0$		
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	0	

Расширенная таблица истинности						
Вход $X_0$	Текущее состояние		Следующее состояние		Вход триггера	
	$Q_1$	$Q_0$	$Q_1$	$Q_0$	$T_1$	$T_0$
0	0	0	0	1	0	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	0	0	1	1
1	0	0	1	1	1	1
1	0	1	0	0	0	1
1	1	0	0	1	1	1
1	1	1	1	0	0	1

Таблица состояний			
$Q_1$	$Q_0$	$X_0$	
		0	1
0	0	01/0	11/1
0	1	10/0	00/0
1	0	11/0	01/0
1	1	00/1	10/0

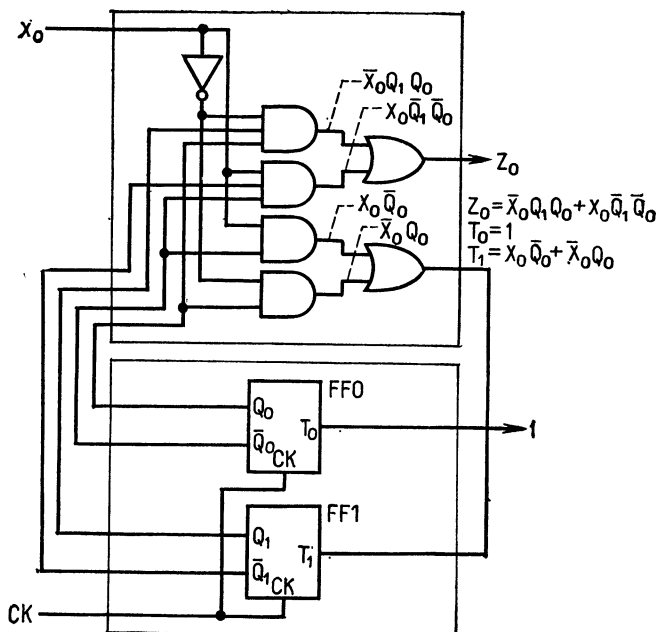


Рис. 5.12. Логическая диаграмма для последовательной схемы.

Значения входных сигналов триггеров получены по формулам (5.2) и (5.3). Выражение для  $T_1$  можно переписать в виде  $T_1 = m_1 + m_3 + m_4 + m_6$  для явного определения строк, где  $T_1$  равен 1. Согласно формуле (5.3),  $T_0$  всегда равен 1 и эти значения включены в расширенную таблицу истинности.

В этой таблице теперь содержатся все комбинации текущего состояния и входов триггеров. Затем определим следующее состояние каждого триггера, заданное каждым переходом. Эти данные будем записывать во втором и третьем столбцах расширенной таблицы истинности. Для получения таких данных используем таблицу состояний  $T$ -триггера (табл. 5.1) для каждого заданного перехода для каждого триггера в каждой строке расширенной таблицы истинности. Для иллюстрации этого метода рассмотрим триггер FF1. В первой строке  $Q_1 = 0$  и  $T_1 = 0$ , так что по таблице состояний FF1-триггера следующим состоянием будет  $Q_1 = 0$ . Во второй строке  $Q_1 = 0$  и  $T_1 = 1$ ; следовательно, следующим состоянием будет  $Q_1 = 1$ . В третьей строке  $Q_1 = 1$  и  $T_1 = 0$ , поэтому следующим состоянием будет  $Q_1 = 1$ . В четвертой строке  $Q_1 = 1$  и  $T_1 = 1$ , так что следующим состоянием будет  $Q_1 = 0$ , и т. д. для остальных четырех строк. Элементы для FF0-триггера обрабатываются таким же образом.

Из табл. 5.2 истинности видно, что имеются две переменные состояния ( $Q_1$  и  $Q_2$ ), откуда следует, что существуют  $4 = 2^2$  состояний, поэтому в таблице состояний четыре строки. Из того, что входная переменная одна, следует, что существуют  $2^1 = 2$  комбинации входов, поэтому таблица состояний должна иметь два столбца. Каждая строка таблиц истинности преобразуется в некоторый элемент таблицы состояний. Например, первая строка в таблицах истинности, представляющая текущее состояние 00 и вход 0, соответствует верхнему левому внутреннему элементу таблицы состояний. Согласно таблицам истинности, следующее состояние есть 01, а сигнал на выходе равен 0. Эта информация записана в виде 01/0 в указанном элементе. Следующая строка представляет текущее состояние 01 и сигнал на входе, равный 0. Для такого сигнала следующим состоянием является 10, а выходной сигнал равен 0. Этот процесс повторяется до тех пор, пока каждая строка таблиц истинности не будет представлена соответствующим элементом таблицы состояний.

**Пример.** Для схемы на рис. 5.13 требуется получить выходную последовательность, если ее начальное состояние  $Q_1 = 0$  и  $Q_0 = 1$ , а входная последовательность  $X = 1, 0, 1, 1, 0$ .

**Решение.** Для построения выходной последовательности необходимо получить таблицу состояний. Сначала запишем выражения для выходного сигнала  $Z$  и входных сигналов триггеров  $J_0$ ,  $K_0$ ,  $J_1$  и  $K_1$ :

$$Z = (\overline{\overline{X \cdot \overline{Q_1}}}) + \overline{Q_0} = \overline{X \cdot \overline{Q_1}} + \overline{Q_0},$$

$$J_0 = \overline{\overline{Q_1}} + \overline{Q_0} = Q_1 + \overline{Q_0},$$

$$K_0 = (\overline{\overline{X \cdot \overline{Q_1}}}) + \overline{Q_0} = \overline{X \cdot \overline{Q_1}} + Q_0,$$

$$J_1 = Q_0,$$

$$K_1 = (\overline{\overline{X \cdot \overline{Q_0}}}) + (\overline{\overline{X \cdot \overline{Q_1}}}) = X \cdot Q_0 + \overline{X \cdot \overline{Q_1}}.$$

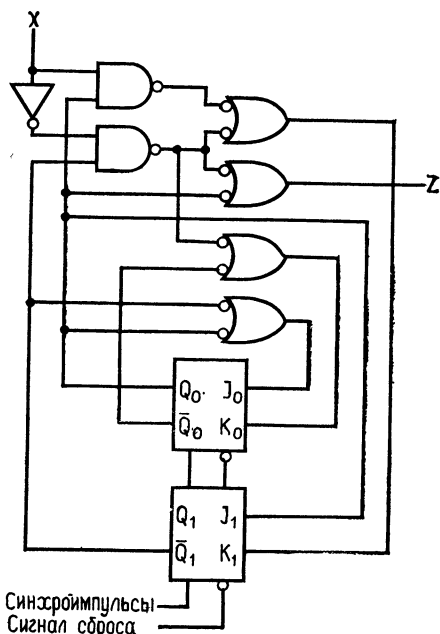


Рис. 5.13. Логическая диаграмма.

По этим выражениям строим таблицу истинности выхода и расширенную таблицу истинности, а также таблицу переходов для  $JK$ -триггера.

X	Текущее состояние		Z	X	Текущее состояние		Следующее состояние		$J_1$	$K_1$	$J_0$	$K_0$
	$Q_1$	$Q_0$			$Q_1$	$Q_0$	$Q_1$	$Q_0$				
0	0	0	1	0	0	0	0	1	0	1	1	1
0	0	1	1	0	0	1	1	0	1	1	0	1
0	1	0	1	0	1	0	1	1	0	0	1	0
0	1	1	0	0	1	1	1	0	1	0	1	1
1	0	0	1	1	0	0	0	1	0	0	1	0
1	0	1	0	1	0	1	1	0	1	1	0	1
1	1	0	1	1	1	0	1	1	0	0	1	0
1	1	1	0	1	1	1	0	0	1	1	1	1

На основе этих таблиц строим таблицу состояний

Текущее состояние		$X=0$	$X=1$
$Q_1$	$Q_0$		
0	0	01/1	01/1
0	1	10/1	10/0
1	0	11/1	11/1
1	1	10/0	00/0

При данных условиях имеем

Вход X	1	0	1	1	0
Текущее состояние $Q_1Q_0$	0 1	1 0	1 1	0 0	0 1
Выход Z	0	1	0	1	1

## 5.6. ПРИМЕРЫ ПРОЕКТИРОВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ

В этом разделе проиллюстрируем процесс построения простых последовательностных схем по их точным словесным спецификациям.

### Десятичный счетчик

Сначала построим десятичный счетчик с помощью  $T$ -триггеров. Эта схема содержит четыре триггера, выходные сигналы которых формируют кодовый набор, интерпретируемый как де-

сятничная величина в двоично-десятичном коде. Каждый синхронизирующий импульс увеличивает значение кода, которое, достигнув 9, переводится следующим синхронизирующим импульсом в 0. При переходе от 9 к 0 схема выдает выходной сигнал, который указывает на переполнение счетчика: этот сигнал часто называют сигналом переноса  $C$ .

Первый шаг в процессе проектирования — определение структурной модели и использование ее для идентификации входных и выходных переменных. На рис. 5.14 показана подобная модель для десятичного счетчика. В этой схеме, кроме синхронизирующего сигнала, нет других входных сигналов. Единственным выходным сигналом комбинационной подсхемы является сигнал переноса  $C$ .

Следующий шаг состоит в построении либо таблицы состояний, либо диаграммы состояний, описывающих поведение схемы. Диаграмма состояний для этого счетчика приведена на рис. 5.15, а. Согласно диаграмме, схема имеет 10 допустимых состояний, последовательность которых есть 0000, 0001, 0010 ... ..., 1000, 1001, затем следует повторение 0000, 0001 и т. д. Поскольку схема не имеет входов, на ориентированных линиях указан лишь текущий выходной сигнал, соответствующий каждому состоянию. Эти значения равны 0 для всех допустимых

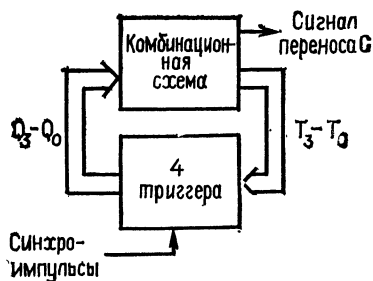
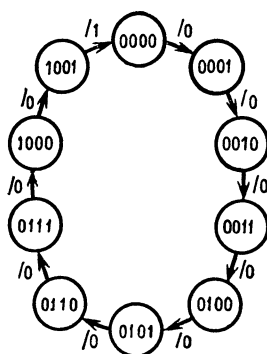


Рис. 5.14. Структурная модель десятичного счетчика.



а

Текущее состояние									
Q3	Q2	Q1	Q0		Q3	Q2	Q1	Q0	
0	0	0	0	0	0	0	0	0	1/0
0	0	0	1	0	0	0	0	1	0/0
0	0	1	0	0	0	0	1	0	1/0
0	0	1	1	0	0	1	0	0	0/0
0	1	0	0	0	0	1	0	1	1/0
0	1	0	1	0	1	0	1	1	0/0
0	1	1	0	0	1	0	1	1	1/0
0	1	1	1	1	1	0	0	0	0/0
1	0	0	0	1	0	0	0	0	1/0
1	0	0	1	0	0	0	0	0	0/1
1	0	1	0	0	d	d	d	d	d/d
1	0	1	1	0	d	d	d	d	d/d
1	1	0	0	0	d	d	d	d	d/d
1	1	0	1	0	d	d	d	d	d/d
1	1	1	0	0	d	d	d	d	d/d
1	1	1	1	1	d	d	d	d	d/d

б

Рис. 5.15. Диаграмма состояний (а) и таблица состояний десятичного счетчика (б).



	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	$Q_1\bar{Q}_0$	$Q_1Q_0$
$\bar{Q}_3\bar{Q}_2$	0	0	0	0
$\bar{Q}_3Q_2$	0	0	1	0
$Q_3\bar{Q}_2$	d	d	d	d
$Q_3Q_2$	0	1	d	d
$T_3 = Q_3 \cdot Q_0 + Q_2 \cdot Q_1 \cdot Q_0$				

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	$Q_1\bar{Q}_0$	$Q_1Q_0$
$\bar{Q}_3\bar{Q}_2$	0	0	1	0
$\bar{Q}_3Q_2$	0	0	1	0
$Q_3\bar{Q}_2$	d	d	d	d
$Q_3Q_2$	0	0	d	d
$T_2 = Q_1 \cdot Q_0$				

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	$Q_1\bar{Q}_0$	$Q_1Q_0$
$\bar{Q}_3\bar{Q}_2$	0	1	1	0
$\bar{Q}_3Q_2$	0	1	1	0
$Q_3\bar{Q}_2$	d	d	d	d
$Q_3Q_2$	0	0	d	d
$T_1 = \bar{Q}_3 \cdot Q_0$				

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	$Q_1\bar{Q}_0$	$Q_1Q_0$
$\bar{Q}_3\bar{Q}_2$	1	1	1	1
$\bar{Q}_3Q_2$	1	1	1	1
$Q_3\bar{Q}_2$	d	d	d	d
$Q_3Q_2$	1	1	d	d
$T_0 = 1$				

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	$Q_1\bar{Q}_0$	$Q_1Q_0$
$\bar{Q}_3\bar{Q}_2$	0	0	0	0
$\bar{Q}_3Q_2$	0	0	0	0
$Q_3\bar{Q}_2$	d	d	d	d
$Q_3Q_2$	0	1	d	d
$C = Q_3 \cdot Q_0$				

Рис. 5.16. Карты Карно десятичного счетчика, определенное на рис. 5.14 и рис. 5.15.

состояний, за исключением 1001. На рис. 5.15,б приведена соответствующая таблица состояний. Так как схема не имеет входов, таблица содержит только один внутренний столбец. Отметим, что для шести неиспользованных состояний от 1010 до 1111 следующие состояния не заданы, они обозначены как неопределенные состояния.

Таблица 5.3 состоит из таблицы истинности для выхода схемы  $C$  и расширенной таблицы истинности для входов триггеров  $T_3$ ,  $T_2$ ,  $T_1$  и  $T_0$ . Выходные значения таблицы истинности и

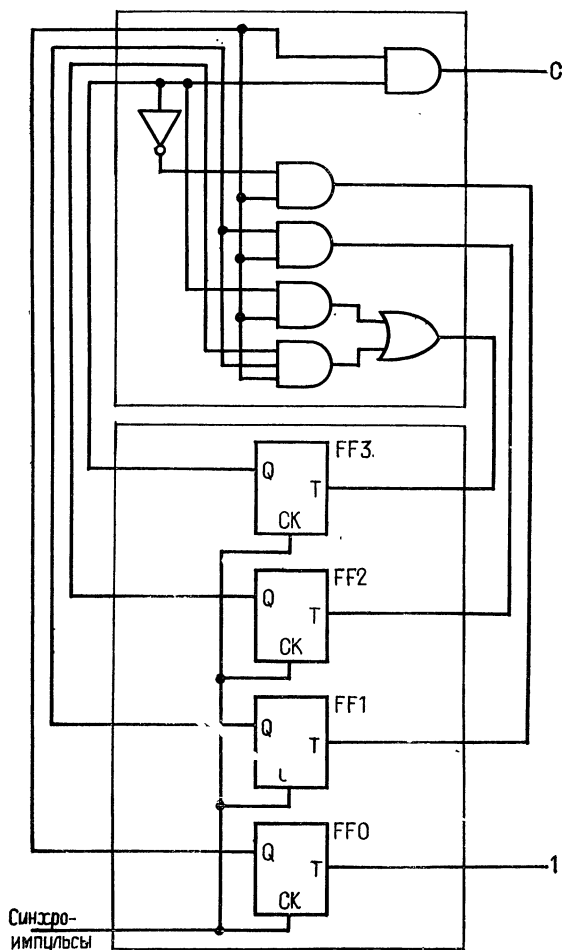


Рис. 5.17. Логическая диаграмма десятичного счетчика, определенного на рис. 5.14 и рис. 5.15.

следующие состояния расширенной таблицы истинности получены последовательно по строкам из соответствующих элементов таблицы состояний или соответствующих ориентированных линий на диаграмме состояний. Входные значения триггеров выбраны такими, чтобы мог осуществляться каждый заданный переход состояний. При определении этих значений была использована таблица переходов  $T$ -триггера (табл. 5.1).

Следующим шагом является получение выражений для выходного сигнала  $C$  и входных сигналов триггеров  $T_3, T_2, T_1$  и  $T_0$ .

Таблица 5.3. Таблица истинности выхода и расширенная таблица истинности для десятичного счетчика, определенного на рис. 5.14 и 5.15

Текущее состояние				Выход С	Текущее состояние				Следующее состояние				Вход триггера			
Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>		Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	T <sub>3</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	1	1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	d	1	0	1	0	d	d	d	d	d	d	d	d
1	0	1	1	d	1	0	1	1	d	d	d	d	d	d	d	d
1	1	0	0	d	1	1	0	0	d	d	d	d	d	d	d	d
1	1	0	1	d	1	1	0	1	d	d	d	d	d	d	d	d
1	1	1	0	d	1	1	1	0	d	d	d	d	d	d	d	d
1	1	1	1	d	1	1	1	1	d	d	d	d	d	d	d	d

На рис. 5.16 показаны карты, использованные при получении этих минимальных сумм произведений. Логическая диаграмма десятичного счетчика, реализованная в соответствии с этими выражениями посредством вентилей И, ИЛИ, НЕ, приведена на рис. 5.17.

### Трехразрядный реверсивный двоичный счетчик

В качестве второго примера рассмотрим трехразрядный реверсивный двоичный счетчик с одним входным сигналом для управления направлением счета и двумя выходными сигналами, один из которых указывает на переполнение счетчика, другой — на ситуацию противоположную переполнению. Эта схема содержит три триггера, выходные сигналы которых формируют кодовый набор, интерпретируемый как двоичная величина. При поступлении каждого синхронизирующего импульса содержимое счетчика увеличивается при наличии входного сигнала и уменьшается при его отсутствии. Выходной сигнал, указывающий на переполнение счетчика и иногда называемый сигналом переноса, появляется при переходе от 111 к 000. Второй выходной сигнал, указывающий на ситуацию противоположную переполнению, иногда называемый сигналом заема, появляется при переходе от 000 к 111. Поскольку наша задача состоит в построении таблицы состояний, т. е. ограничена функциональным

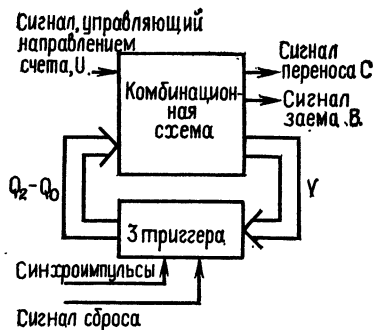
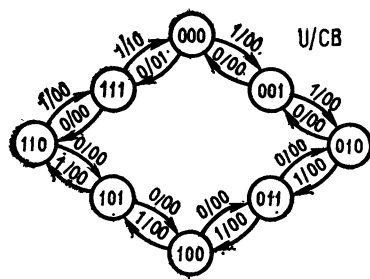


Рис. 5.18. Структурная модель трехразрядного нисходящего двоичного счетчика.



Текущее состояние $Q_2 Q_1 Q_0$	Входы					
	0			1		
0 0 0	1	1	1/0	1	0	0
0 0 1	0	0	0/0	0	0	1
0 1 0	0	0	1/0	0	1	0
0 1 1	0	1	0/0	0	1	0
1 0 0	0	1	1/0	0	1	0
1 0 1	1	0	0/0	0	1	0
1 1 0	1	0	1/0	1	1	0
1 1 1	1	1	0/0	0	0	0

Рис. 5.19. Диаграмма состояний (a) и таблица состояний трехразрядного нисходящего двоичного счетчика (б).

описанием трехразрядного счетчика, нет необходимости в уточнении конкретных типов используемых триггеров.

Структурная модель трехразрядного реверсивного двоичного счетчика показана на рис. 5.18. Входной сигнал, управляющий направлением счета, обозначен через  $U$ , сигнал переноса — через  $C$ , а сигнал заема — через  $B$ .

На рис. 5.19 приведены диаграмма и таблица состояний этой схемы в случае соглашения о положительной логике. Когда входной сигнал  $U$  имеет значение 0, следующее состояние получается путем вычитания единицы из двоичной величины, определенной текущим состоянием. Это справедливо даже для состояния 000, которое при вычитании единицы переходит в 111. При этом переходе возникает сигнал заема, который устанавливается в 1. Когда входной сигнал равен 1, следующее состояние находится путем прибавления единицы к двоичной величине, определенной текущим состоянием. Это также верно для состояния 111, которое при прибавлении единицы переходит в 000. При таком переходе возникает сигнал переноса  $C$ , который устанавливается в 1.

Таблица истинности для  $C$  и  $B$  и расширенная таблица истинности для входов триггеров могут быть построены либо

исходя из таблицы состояний, либо исходя из диаграммы с использованием таблицы переходов для выбранного типа триггера. По этим таблицам истинности можно получить выражения для минимальных сумм произведений, которые используются при построении логической диаграммы.

### Двухразрядный сдвиговый регистр

В качестве третьего примера рассмотрим двухразрядный сдвиговый регистр, последовательностную схему, в которой содержимое каждого триггера сдвигается в соседний с ним триггер. В этом примере направление передачи управляется входным сигналом, содержимое сдвигается вправо при наличии указанного сигнала и влево, если он отсутствует. Сдвинутые данные поступают на линию вывода данных, и величина на линии ввода данных сдвигается в регистр.

Структурная модель двухразрядного сдвигового регистра показана на рис. 5.20. Сигнал, управляющий направлением сдвига, обозначен через  $R$ , а сигналы ввода и вывода данных представлены символами  $DI$  и  $DO$  соответственно.

Поведение этой схемы описывается при помощи таблицы состояний, содержащейся в табл. 5.4. Так как схема имеет два входа  $R$  и  $DI$ , возможны четыре входные комбинации. Поскольку все они допустимы, в таблице состояний для них имеются четыре столбца. В схеме два триггера, что приводит к четырем текущим состояниям, и поэтому таблица имеет четыре строки.

Чтобы найти элементы таблицы, сначала по  $R$  определяется направление сдвига, а затем оцениваются выходные данные и влияние сдвига на содержимое регистра при рассматриваемых входных данных. В качестве примера обратимся к элементам второй строки, в которой текущим состоянием является 01. Для определения соответствующего следующего состояния и выходного сигнала воспользуемся рис. 5.21.

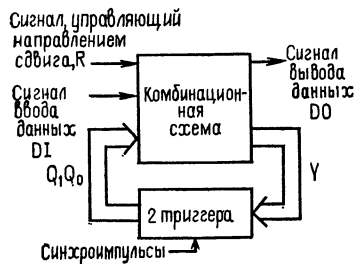


Рис. 5.20. Структурная модель для двухразрядного сдвигового регистра.

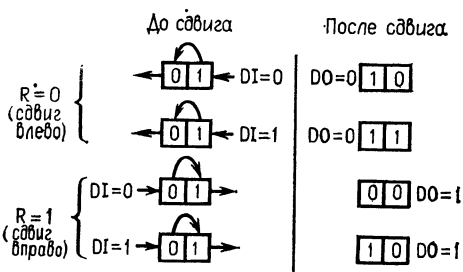


Рис. 5.21. Функционирование двухразрядного сдвигового регистра при текущем состоянии 01.

Таблица 5.4. Таблица состояний для двухразрядного сдвигового регистра

Текущее состояние $Q_1Q_0$	Входы R DI			
	00	01	10	11
00	00/0	01/0	00/0	10/0
01	10/0	11/0	00/1	10/1
10	00/1	01/1	01/0	11/0
11	10/1	11/1	01/1	11/1

В двух верхних примерах на рис. 5.21, где  $R = 0$ , происходит сдвиг влево. Таким образом, 0 в левой позиции сдвигается в линию выходных данных, 1 в правой позиции сдвигается в левую позицию и значение на входе независимо от того, равно ли оно 0 или 1, сдвигается в правую позицию. В двух нижних примерах на этом рисунке, где  $R = 1$ , выполняется сдвиг вправо. Таким образом, 1 в правой позиции сдвигается в линию выходных данных, 0 в левой позиции сдвигается вправо, и значение на входе сдвигается в левую позицию. В первом примере для входной комбинации 00 и текущего состояния 01 следующим состоянием является 10, а выходной сигнал равен 0. Эти данные записываются как 10/0 в первом столбце второй строки табл. 5.4. Аналогично из второго примера видно, что для входной комбинации 01 и текущего состояния 01 следующим состоянием является 11, а выходной сигнал равен 0. В третьем примере для входной комбинации 10 и текущего состояния 01 следующим состоянием является 00, а выходной сигнал равен 1. В четвертом примере для входной комбинации 11 и текущего состояния 01 следующим состоянием является 10, а выход равен 1.

Элементы в остальных трех строках находятся таким же способом. Эта таблица состояний может быть преобразована в таблицы истинности, а затем — в логическую диаграмму, представляющую реализацию минимальной суммы произведений.

### Восьмиразрядный сдвиговый регистр

Хотя подход, основанный на таблице состояний, оказывается эффективным для простых схем, его применение становится слишком громоздким по мере увеличения числа триггеров. Часто схему можно разбить на меньшие, но одинаковые блоки, которые могут быть построены с помощью таблицы состояний. Полная схема проектируется путем спецификации взаимосвязей.

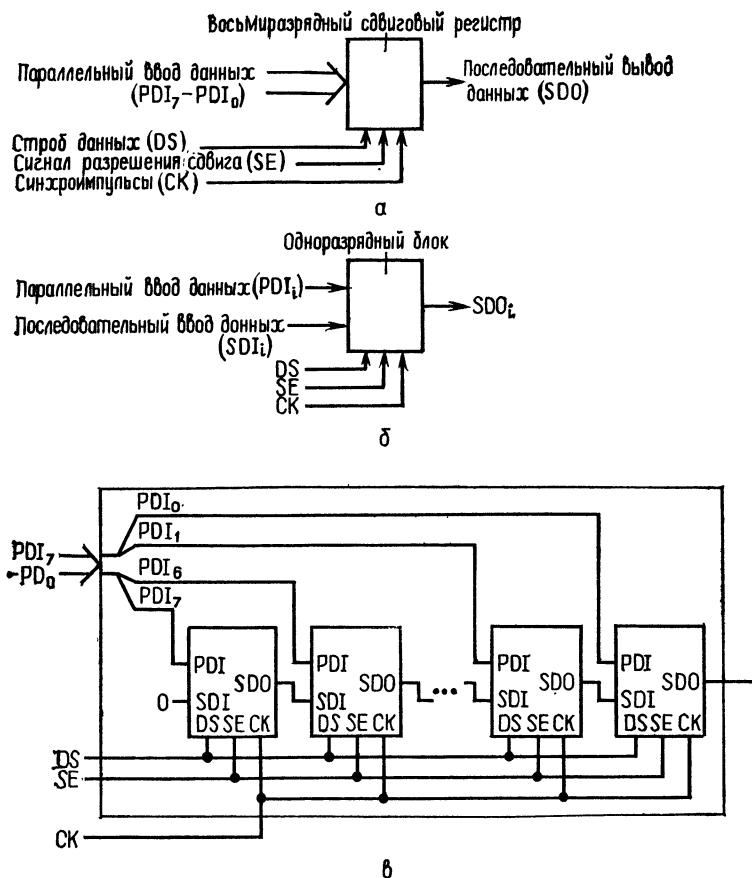


Рис. 5.22. Восьюмиразрядный сдвиговый регистр.

*а* — модель ввода-вывода; *б* — модель ввода-вывода одноразрядного блока; *в* — диаграмма внутренних соединений.

Чтобы пояснить идею разбиения, спроектируем регистр, который принимает 8 бит параллельных данных и затем последовательно выдвигает их из регистра, начиная с младшего разряда. Эта схема называется **преобразователем параллельного кода в последовательный**. Схема имеет два управляющих входных сигнала: сигнал стробирования данных  $(DS)$ , который равен 1, когда ввод параллельных данных допустим, и сигнал разрешения сдвига  $(SE)$ , который должен быть равен 1, когда передаются последовательные данные. Сигнал  $DS$  равен 1 в течение одного синхронизирующего импульса при загрузке параллельных данных в регистр, сигнал  $SE$  должен быть равен 1 в течение восьми синхронизирующих импульсов для сдвига из ре-

гистра всех 8 бит. На рис. 5.22, *а* показана модель ввода-вывода для этой схемы. Модель ввода-вывода для одноразрядного блока иллюстрируется на рис. 5.22, *б*, а взаимосвязь восьми одноразрядных блоков — на рис. 5.22, *в*.

Теперь построим таблицу состояний для одноразрядного блока. Поскольку он содержит только один триггер, у него лишь два текущих состояния. Кроме того, он имеет четыре различных входных сигнала, так что существуют 16 входных комбинаций. Табл. 5.5 является таблицей состояний для одноразрядного

Таблица 5.5. Таблица состояний одного каскада восьмиразрядного сдвигового регистра

Текущее состояние Q	Входы DS SE PDI SDI			
	0000	0001	0010	0011
0	0/d	0/d	0/d	0/d
1	1/d	1/d	1/d	1/d
	0100	0101	0110	0111
	0/0 0/1	1/0 1/1	0/0 0/1	1/0 1/1
	1000	1001	1010	1011
	0/d 0/d	0/d 0/d	1/d 1/d	1/d 1/d
	1100	1101	1110	1111
	d/d d/d	d/d d/d	d/d d/d	d/d d/d

блока. В первых четырех столбцах состояния не изменяются, а выход соответствует неопределенному значению, так как оба управляющих входных сигнала равны 0. Во второй четверке столбцов  $DS = 0$  и  $SE = 1$ , поэтому данные на входе SDI будут вводиться в триггер и текущее состояние будет выдвинуто из триггера. Таким образом, следующее состояние определяется по входному значению SDI, а выходное — по текущему состоянию. В следующих четырех столбцах  $DS = 1$  и  $SE = 0$ , поэтому данные на входе PDI должны быть загружены в триггер, а значение выхода не представляет интереса. Таким образом, следующее состояние определяется входом PDI, а выход соответствует неопределенному значению. В последних четырех столбцах  $DS = 1$  и  $SE = 1$ . Поскольку эта входная комбинация неправильная, следующее состояние и выход соответствуют неопределенным значениям.



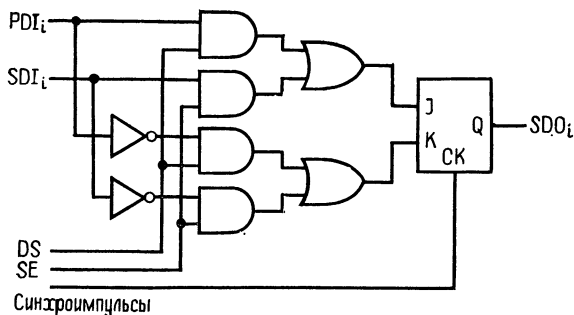


Рис. 5.23. Логическая диаграмма одноразрядного блока восьмиразрядного сдвигового регистра.

Таблица состояний (табл. 5.5) может быть преобразована в логическую диаграмму с помощью методов, описанных в предыдущих примерах. То обстоятельство, что мы имеем четыре входные переменные и одну переменную состояния, немного усложняет ситуацию, так как мы не рассматривали карту Карно для пяти переменных. Вместо этого опишем более интуитивный подход к проектированию указанного одноразрядного блока. Отметим, что его триггер должен быть установлен в 1, если поступили сигнал стробирования данных  $DS$  и входной сигнал параллельных данных  $PDI$  или если имеются как сигнал разрешения сдвига  $SE$ , так и входной сигнал последовательных данных  $SDI$ . Эти условия объединяются в виде

$$SET = DS \cdot PDI_i + SE \cdot SDI_i, \quad (5.4)$$

где  $SET$  — сигнал для установки триггера в 1. Аналогичное соотношение может быть выведено для сигнала  $RESET$ :

$$RESET = DS \cdot \overline{PDI_i} + SE \cdot \overline{SDI_i}. \quad (5.5)$$

Отметим также, что  $SDO_i$  просто представляет выходной сигнал триггера. Исходя из этих соотношений, можно построить логическую диаграмму для одного разряда. На рис. 5.23 приведена эта диаграмма, использующая  $JK$ -триггер, где логические вентильные схемы, реализующие уравнение (5.4), соединены с входом  $J$ , тогда как схемы, реализующие уравнение (5.5), соединены с входом  $K$ .

## 5.7. ПРИМЕРЫ НЕКОТОРЫХ ВАЖНЫХ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ

В данном разделе приведены примеры некоторых важных последовательностных схем, изготовленных в виде одной интегральной схемы. В общем их можно разбить на три класса:

счетчики, сдвиговые регистры и элементы памяти. Для описания поведения этих схем вместо подробных моделей, представленных ранее в этой главе, мы воспользуемся сокращенными таблицами напряжений.

### Счетчики

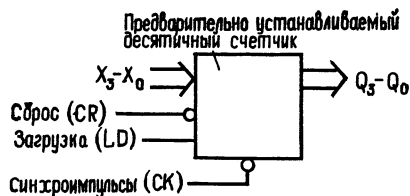
**Счетчики** являются последовательностными схемами, которые проходят через последовательность состояний в соответствии со схемой счета. Существуют два типа счетчиков: десятичные и двоичные. **Десятичные счетчики** содержат четыре триггера и порождают последовательность, соответствующую 0, 1, ..., 8, 9, 0, 1 и т. д. **Двоичные счетчики** могут содержать три, четыре или большее число триггеров.  $N$ -разрядный двоичный счетчик порождает последовательность, соответствующую 0, 1, ...,  $2^N - 1$ , 0, 1 и т. д.

Кроме входа синхронизации счетчики имеют ряд других входов, которые могут оказаться полезными в различных приложениях. Примером является вход, управляющий направлением последовательности счета. Этот вход называется **линией управления направлением счета**. Многие счетчики имеют вход сброса, подача сигнала на который устанавливает все триггеры в счетчике в 0. Некоторая группа входов допускает выборочную установку в 1 или 0 всех триггеров в счетчике. Эта группа содержит управляющий вход, называемый **линией загрузки** или **управляющей линией предварительной установки**, по которой передаются данные в группу входных линий предварительной установки триггеров. Таким образом, счетчик может быть предварительно установлен в любое значение путем подбора соответствующей комбинации входных сигналов предварительной установки и при активизации линии загрузки.

Обычно на входах сброса и линии загрузки используются правила отрицательной логики, так что сигналы на этих входах появляются, когда на линиях устанавливается низкий уровень напряжения. Даже в синхронных счетчиках эти сигналы могут быть асинхронными, поэтому результирующие переходы осуществляются, как только на входе появляется сигнал. Часто этим входам присваивают приоритеты для определения доминирующего входа, если сигналы появляются одновременно на двух входах.

В дополнение к выходам триггера могут быть **выход переноса** и **выход заема**.

Рассмотрим в качестве примера счетчика предварительно устанавливаемый десятичный счетчик с асинхронными входами сброса и предварительной установки. На рис. 5.24 показана модель ввода-вывода этой схемы и приведена сокращенная табли-



а

Входы			Следующее состояние	Комментарии
Сброс	Загрузка	Синхронизация		
L	<i>d</i>	<i>d</i>	LLLL	Все триггеры очищены асинхронно
H	H	<i>d</i>	$X_3X_2X_1X_0$	Все триггеры предварительно установлены асинхронно
H	L	L или H	$(Q_3Q_2Q_1Q_0)$	Нет изменений
H	L	↓	$(Q_3Q_2Q_1Q_0) + 1$	Увеличивается на 1 на активной фазе

б

Рис. 5.24. Предварительно устанавливаемый десятичный счетчик: Н-уровень низкого напряжения, L-уровень высокого напряжения, *d*-неопределенное условие, ↓-запуск по заднему фронту.

ца напряжений, которая описывает ее поведение. Выходы  $Q_3, \dots, Q_0$  представляют истинные выходы четырех триггеров. Кружок на входе СБРОС соответствует отрицательной логике, так что сигнал на этом входе появляется при низком уровне напряжения. Для входа ЗАГРУЗКА используется соглашение о положительной логике, так что сигнал на этом входе появляется при высоком уровне напряжения. Входы предварительной установки  $X_3, \dots, X_0$  соединены с соответствующими триггерами. Отрицательный фронт представляет активную фазу сигнала синхронизации, что отмечено кружком на данном входе в модели. Активная фаза синхронизирующего импульса вызывает переходы состояний, которые соответствуют прибавлению единицы к хранимой в триггерах величине. Поскольку схема является десятичным счетчиком, последовательность будет иметь вид 0, 1, ..., 8, 9, 0, 1 и т. д.

Первая строка в сокращенной таблице напряжений указывает на то, что счетчик будет устанавливаться в исходное состояние, если на вход СБРОС поступает сигнал, соответствующий низкому уровню напряжения независимо от сигналов на других входах. Таким образом, сигнал СБРОС имеет наибольший при-

оритет. Из второй строки видно, что если сигнал СБРОС отсутствует, т. е. на этом входе высокий уровень напряжения и на входе ЗАГРУЗКА также высокий уровень напряжения, то триггеры должны быть загружены или предварительно установлены в исходные состояния, соответствующие сигналам на входах предварительной установки  $X_3, \dots, X_0$  независимо от состояния сигнала синхронизации. Поэтому сигнал ЗАГРУЗКА имеет второй приоритет. Третья строка указывает на то, что если сигналы СБРОС и ЗАГРУЗКА отсутствуют и сигнал синхронизации не изменяет своего значения (т. е. сохраняет либо высокий, либо низкий уровень), то состояния триггеров неизменны. Четвертая строка показывает, что, если сигналы СБРОС и ЗАГРУЗКА отсутствуют, счетчик будет увеличивать свое содержимое на 1 при каждом отрицательном фронте синхронизирующего импульса.

Эта схема (или элемент интегральной схемы) может иметь многочисленные применения. На рис. 5.25 приведена схема из двух таких элементов, которая выполняет функцию делителя синхронизирующей частоты на 100; это означает, что она вырабатывает один синхронизирующий импульс на выходе на каждые 100 входных импульсов. Сигнал УСТАНОВКА В НАЧАЛЬНОЕ СОСТОЯНИЕ, который обычно подается в начале работы схемы, обеспечивает установку триггеров в 0. Первые 10 входных синхронизирующих импульсов вызовут переполнение левого десятичного счетчика, который называется младшим десятичным счетчиком. При поступлении десятого импульса сигнал на выходе  $Q_3$  изменится от высокого уровня напряжения к низкому и этот перепад приведет к увеличению содержимого десятичного счетчика справа, который является следующим по значимости десятичным счетчиком. Так как содержимое старшего десятичного счетчика увеличивается на единицу один раз на каждые 10 входных синхронизирующих импульсов, он переполнится после 100 входных синхронизирующих импульсов и вызовет переход с высокого уровня напряжения на низкий на выходе  $Q_3$ . Когда это произойдет, все триггеры в двух десятичных счетчиках будут установлены в 0 и готовы к повторению процесса. Таким образом, выходная синхронизирующая частота будет составлять одну сотую часть входной частоты.

Рис. 5.26 иллюстрирует второе применение счетчика, описанного на рис. 5.24. Эта схема, которая производит деление входной синхронизирующей частоты на 360, использует возможность предварительной установки устройства. Трехкаскадный десятичный счетчик при отсутствии предварительной установки будет производить деление частоты на 1000. Это означает, что после каждых 1000 входных синхронизирующих импульсов будет иметь место переход с высокого уровня напряжения на низ-

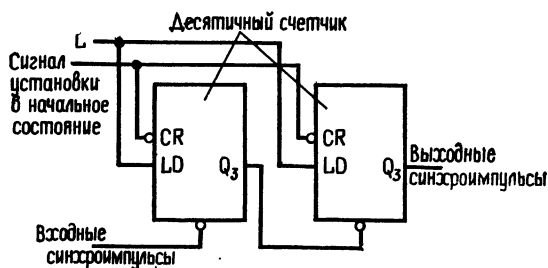


Рис. 5.25. Схема деления на 100, использующая предварительно устанавливаемый десятичный счетчик на рис. 5.24.

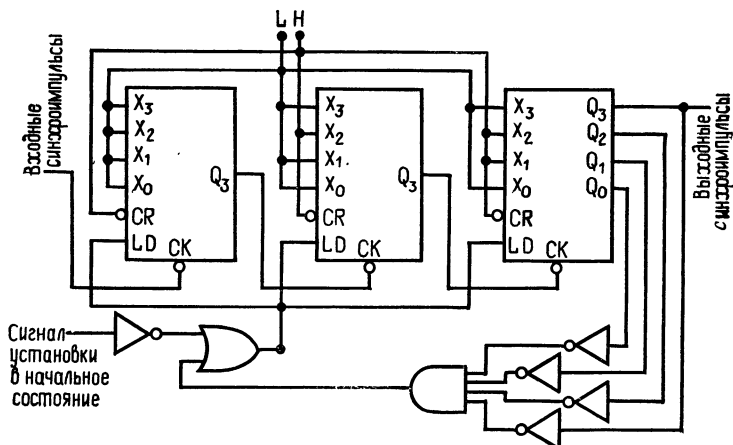


Рис. 5.26. Схема деления на 360, использующая предварительно устанавливаемый десятичный счетчик на рис. 5.24.

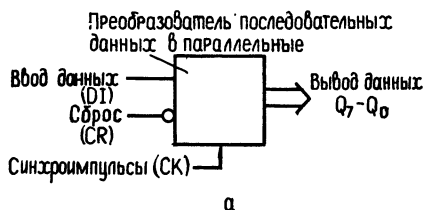
кий на выходе старшего десятичного счетчика. Если же в счетчик предварительно устанавливается 640, то для изменения выхода старшего десятичного счетчика с высокого уровня на низкий потребуется только 360 импульсов. Сигналы, подаваемые на входы предварительной установки левого младшего десятичного счетчика, обеспечивают установку 0; подключение входов среднего десятичного счетчика к линии предварительной установки производится таким образом, чтобы обеспечить в нем установку 4, в правом старшем десятичном счетчике устанавливается предварительная установка 6. Сигнал разрешения предварительной установки, ЗАГРУЗКА, может формироваться сигналом УСТАНОВКА В НАЧАЛЬНОЕ СОСТОЯНИЕ, который предполагается «активным» на низком уровне, или же с помощью вентиля И, который выделяет его, когда содержимое старшего десятичного счетчика имеет нулевое значение. Послед-

нее соединение необходимо для того, чтобы вся схема повторно генерировала выходной импульс на каждые 360 входных импульсов.

### Сдвиговые регистры

**Сдвиговый регистр** представляет собой последовательностную схему, в которой содержимое каждого триггера передается в соседний с ним триггер. Величина, содержащаяся в последнем триггере, выдвигается и значение на входной линии вводится в первый триггер регистра. Часто сдвиговые регистры предусматривают дополнительный вход, управляющий направлением сдвига (влево или вправо). Во многих сдвиговых регистрах имеется также шина управления для параллельной загрузки триггеров через их отдельные входы. Все входы, выходы и шины управления редко удается разместить в одном блоке, однако существуют различные их комбинации.

Две важные конфигурации сдвиговых регистров представляют преобразователь параллельного кода в последовательный и преобразователь последовательного кода в параллельный. Если все разряды двоичного слова передаются одновременно в течение одного интервала времени по своей линии, осуществляется **параллельная передача**. Если же многоразрядное двоичное слово передается разряд за разрядом по одной линии, имеет место **последовательная передача**. Термины параллельный и последовательный используются также для описания двоичных слов и информации, переданной соответствующим способом. Сдвиго-



Входы		Следующее состояние								Комментарии
Сброс	Синхронизация	Q <sub>7</sub>	Q <sub>6</sub>	Q <sub>5</sub>	Q <sub>4</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	
L	d	L	L	L	L	L	L	L	L	Сброс Нет изменений Сдвиг
H	H или L	Q <sub>7</sub>	Q <sub>6</sub>	Q <sub>5</sub>	Q <sub>4</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	
H	↑	DI	Q <sub>7</sub>	Q <sub>6</sub>	Q <sub>5</sub>	Q <sub>4</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	

б

Рис. 5.27. Восьмиразрядный преобразователь последовательных данных в параллельные.

вый регистр, используемый в качестве преобразователя параллельных данных в последовательные, имеет линию управления, которая загружает все триггеры одновременно параллельным словом, и вторую линию управления, сигналы которой последовательно сдвигают данные в выходную шину. В случае преобразователя последовательных данных в параллельные последовательные данные сдвигаются в сдвиговый регистр и затем выводятся как параллельное слово. На рис. 5.27 приведена модель ввода-вывода и сокращенная таблица напряжений для восьмиразрядного преобразователя последовательных данных в параллельные. Схема имеет три входных сигнала. Сигнал СБРОС, который является активным при низком напряжении и имеет наивысший приоритет, устанавливает все восемь триггеров в 0. Остальные два входных сигнала, сигнал синхронизации и сигнал ВВОД ДАННЫХ, управляют последовательной загрузкой схемы. Во время активной фазы цикла синхронизации, который в данном случае определяется переходом с низкого уровня напряжения на высокий, содержимое всех триггеров передается направо, а сигнал из линии ВВОД ДАННЫХ передается на  $Q_7$ . После восьми циклов синхронизации восьмиразрядное последовательное слово будет введено в регистр и окажется доступным в параллельной форме на выходах  $Q_7, \dots, Q_0$ .

### Элементы памяти

Простейшим элементом памяти является один регистр, часто называемый фиксатором (latch) или буфером данных. Он слу-

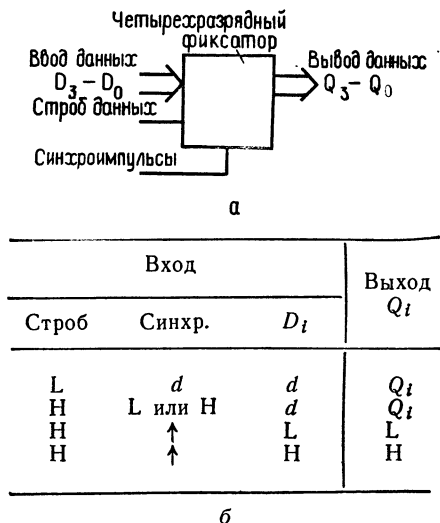


Рис. 5.28. Четырехразрядный буфер данных.

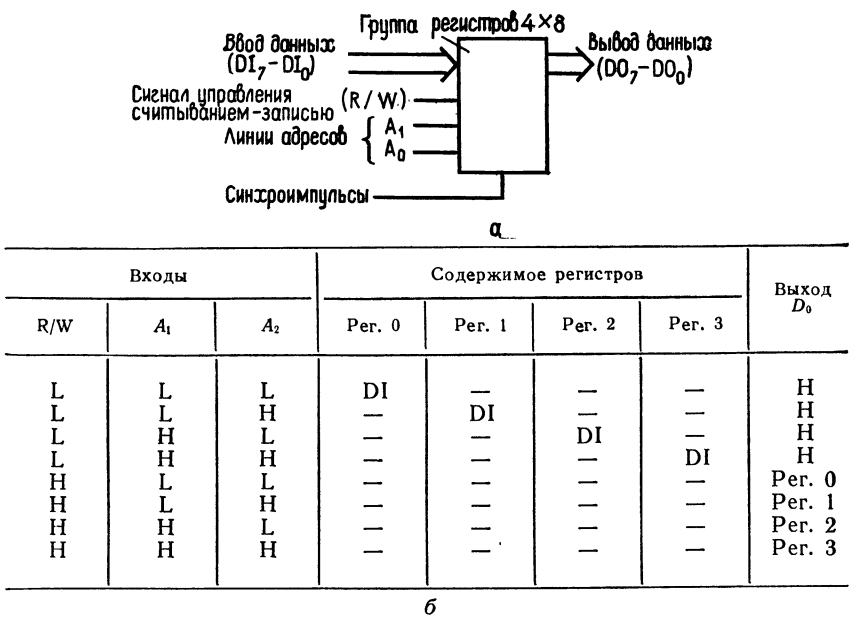


Рис. 5.29. Группа регистров размера  $4 \times 8$ . (Прочерк указывает на отсутствие изменений.)

жит в качестве временной памяти для параллельного слова, которое может быть доступным лишь в определенном интервале времени. Схемы этого типа используются при сопряжении цифровых систем. На рис. 5.28 показаны модель ввода-вывода и сокращенная таблица напряжений для синхронного четырехрядного буфера данных. Схема имеет четыре входа  $D_3, \dots, D_0$ , каждый из которых соединен с триггером. Уровни напряжения на этих линиях передаются в триггеры во время активной фазы сигнала синхронизации, когда сигнал стробирования имеет высокий уровень. Данные, которые затем хранятся в регистре, могут быть считаны с выходов  $Q_3, \dots, Q_0$ .

Существуют интегральные схемы, содержащие более одного регистра на кристалле. На рис. 5.29 показана модель ввода-вывода для **группы регистров** размера  $4 \times 8$ , которая содержит 32 триггера, организованные как четыре восьмиразрядных слова. Имеется восемь линий ввода данных  $DI_7, \dots, DI_0$ ; сигналы с этих линий могут быть введены (или записаны) в любой из четырех восьмиразрядных регистров. Существуют восемь линий вывода данных  $DO_7, \dots, DO_0$ ; содержимое любого из четырех регистров может быть выведено на эти линии. Направление потока данных, т. е. производится ли запись данных в регистре или они считываются из регистра, определяется управляющей



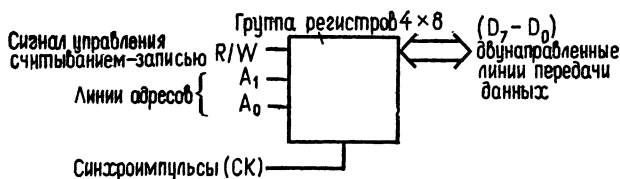


Рис. 5.30. Группа регистров размера  $4 \times 8$  с двунаправленными линиями передачи данных.

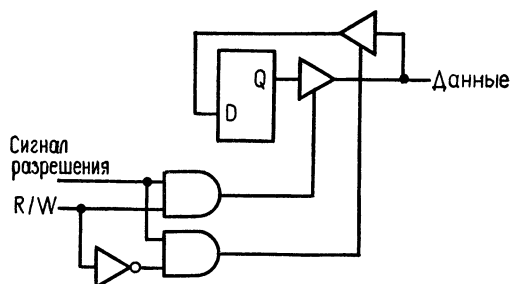


Рис. 5.31. Ячейка триггера с двунаправленными линиями передачи данных.

линией считывание-запись  $R/W$ . Когда на этой линии установлен низкий уровень напряжения, данные должны быть приняты или записаны в схему; когда на ней установлен высокий уровень напряжения, данные должны быть выданы или считаны из схемы. Регистр, участвующий при передаче, определяется двумя адресными линиями. Эти две линии могут иметь четыре состояния LL, LH, HL и HH. Каждое состояние используется для обозначения одного из четырех регистров. В качестве примера функционирования данной схемы рассмотрим комбинацию состояний  $R/W = L$ ,  $A_1 = H$  и  $A_0 = L$ , при которой запоминаются или записываются сигналы с линий ввода данных  $DI_7, \dots, DI_0$  в регистр 2 и на линиях вывода  $DO_7, \dots, DO_0$  сохраняются высокие уровни напряжения. Аналогично при  $R/W = H$ ,  $A_1 = L$ ,  $A_0 = H$  содержимое регистра 1 поступает на линии вывода, а данные на линиях  $DI_7, \dots, DI_0$  игнорируются.

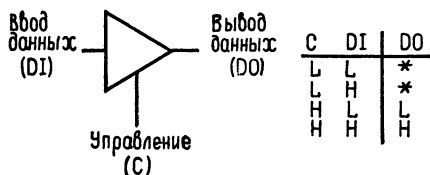


Рис. 5.32. Буфер с тремя состояниями: L-уровень низкого напряжения, H-уровень высокого напряжения, \* — состояние высокого импеданса.

По мере усовершенствования полупроводниковой технологии размер регистровых запоминающих устройств увеличивается. С ростом сложности запоминающих устройств требуется значительно большее число входных и выходных соединений. Поскольку число входных и выходных соеди-

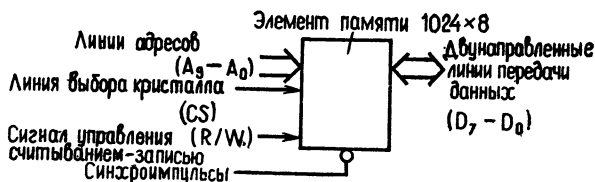


Рис. 5.33. Модель ввода-вывода элемента памяти.

нений является фактором, усложняющим как производство, так и эксплуатацию этих устройств, были разработаны методы уменьшения числа соединений. Значительный прогресс достигнут с использованием двунаправленных линий передачи данных. На рис. 5.30 приведена модель ввода-вывода для регистрового запоминающего устройства размера  $4 \times 8$ , использующая двунаправленные линии передачи данных.

Соединение двунаправленной линии с ячейкой триггера показано на рис. 5.31. Кроме двунаправленной линии передачи данных и линии управления считыванием и записи имеется вход, разрешающий обращение к конкретной ячейке. Сигнал, поступающий на этот вход, генерируется где-то в интегральной схеме из сигналов входных адресных линий. На рис. 5.32 показано новое графическое обозначение **буфера с тремя состояниями**. Поведение этого элемента описано на рисунке. Отметим, что выход рассматриваемого элемента имеет три состояния, а именно стандартные состояния низкого и высокого напряжения и состояние высокого импеданса. Когда схема находится в последнем состоянии, ее выход представляет собой разомкнутый контур. Если на входе управления имеется высокий уровень, то на выходе будет также высокий уровень. Однако, если на входе управления низкий уровень, выход будет находиться в состоянии высокого импеданса. Такая схема необходима по двум причинам: 1) она используется для согласования одной линии передачи данных с входом и выходом триггера и 2) что более важно устраняет ряд проблем согласования, которые могут возникать при соединении общей линии передачи данных с большим числом ячеек триггеров.

В настоящее время имеются регистровые запоминающие устройства, содержащие сотни и даже тысячи регистров, которые размещены в одной интегральной схеме. Поскольку они обладают большой емкостью, их называют **элементами памяти**. На рис. 5.33 показана модель ввода-вывода для элемента памяти размера  $1024 \times 8$ , требующего 10 адресных линий для однозначного выбора одного из 1024 восьмиразрядных регистров. Кроме того, имеется линия выбора кристалла, на которую при любой передаче данных также должен быть подан сигнал.

*Т. Барти*

### 6.1. ВВЕДЕНИЕ

Арифметико-логическое устройство (АЛУ) является узлом ЭВМ, который выполняет арифметические и логические операции над данными, обрабатываемыми ЭВМ. Этот узел может быть относительно малых размеров и состоять из одного или нескольких кристаллов больших интегральных схем (БИС), а в случае переработки большого объема числовых данных (в ЭВМ, предназначенных для решения научно-исследовательских задач) он может состоять из значительного числа быстродействующих логических компонент. Несмотря на различия по размерам и сложности, малые машины при выполнении арифметических и логических операций используют в общем те же принципы, что и большие машины. Различия касаются скорости действия логических вентилях и триггеров, содержащихся в них; кроме того, в больших ЭВМ применяются специальные методы для ускорения выполнения операций и обеспечения параллелизма некоторых операций.

Хотя АЛУ современных машин могут осуществлять многие функции, основные арифметические операции — сложение, вычитание, умножение и деление — остаются главными. При описании новых машин интервалы времени, требуемые для сложения и умножения, всегда указываются в качестве важных характеристик. Поэтому в этой главе сначала описываются средства, с помощью которых ЭВМ производит сложение, вычитание, умножение и деление чисел, а уже затем рассматриваются другие основные операции, такие, как сдвиг, логическое умножение и логическое сложение.

Следует иметь в виду, что устройство управления направляет работу АЛУ, которое выполняет суммирование, вычитание, сдвиг и т. д., когда обеспечивается правильная последовательность входных сигналов. Функцией управляющего элемента яв-

ляется подача этих сигналов, а функцией элементов памяти — снабжение арифметического элемента информацией, которая должна быть использована. Поэтому будем предполагать, что устройства управления и памяти могут выдавать правильные сигналы управления и что мы располагаем данными, над которыми должны выполняться операции. Таким образом, АЛУ предназначается для сложения, вычитания или для выполнения любой операции, указываемой устройством управления.

## 6.2. КОНСТРУКЦИЯ АЛУ

В общем случае информация, обрабатываемая ЭВМ, разделяется на «слова», состоящие из фиксированного числа двоичных разрядов. Например, слова, обрабатываемые данной двоичной машиной, могут иметь длину 32 двоичных разряда. В этом случае АЛУ должно иметь возможность производить сложение, вычитание и т. д. над словами, состоящими из 32 двоичных разрядов. Используемые в дальнейшем операнды поступают из памяти вычислительной машины и управляющий элемент указывает операции, которые должны выполняться. Если необходимо провести операцию сложения, следует ввести первое и второе слагаемые в АЛУ, которое должно сложить эти числа и затем по крайней мере временно сохранить результат (сумму).

Чтобы ввести ряд понятий, рассмотрим конструкцию АЛУ типичной ЭВМ. Устройства памяти содержат набор триггерных **регистров**, каждый из которых состоит из одного или нескольких триггеров. **Длина** любого регистра определяется как максимальное количество информации, которое может хранить регистр. Длина двоичного регистра равна максимальному числу двоичных разрядов, которые он может хранить. В случае двоично-десятичного регистра длина регистра равна числу десятичных разрядов, которые могут храниться в регистре.

Для удобства различным регистрам АЛУ, как правило, присваиваются имена, такие, как *X*-регистр, *B*-регистр, *MQ*-регистр и т. д., а затем триггерам приписываются те же имена, так что *X*-регистр будет содержать триггеры  $X_1$ ,  $X_2$ ,  $X_3$  и т. д.

Большинство вычислительных машин (особенно микропроцессоры) имеют регистр, называемый накапливающим сумматором (аккумулятор), который является основным регистром для арифметических и логических операций. Этот регистр хранит результат каждой арифметической или логической операции; к нему подсоединяются вентильные схемы для выполнения необходимых операций над его содержимым и содержимым других регистров.

Накапливающий сумматор служит основным запоминающим регистром арифметического элемента. Если в машину введена команда **загрузить** накапливающий сумматор, то сначала управляющий элемент должен очистить накапливающий сумматор от данных, которые могли в нем храниться, а затем ввести в него выбранный из памяти операнд. Если вычислительная машина получила команду сложить числа, то число, хранимое в накапливающем сумматоре, будет представлять первое слагаемое. Второе слагаемое находится в это время в памяти, и соответствующие схемы ЭВМ прибавят это число (второе слагаемое) к числу (первому слагаемому), уже содержащемуся в накапливающем сумматоре, и сохраняют в нем полученную сумму. Отметим, что исходное первое слагаемое хранится в накапливающем сумматоре лишь до сложения чисел. Более того, в дальнейшем сумма или может остаться в накапливающем сумматоре, или же будет передана в устройство памяти в зависимости от типа ЭВМ. В этой главе мы рассмотрим лишь процессы сложения, вычитания и т. д., а процессы размещения числа, вводимого в память, или передачи чисел в память не будем затрагивать.

Некоторые ЭВМ имеют не один, а два или большее число накапливающих сумматоров; их обозначают, например, как *A*-накапливающий сумматор, *B*-накапливающий сумматор (как в микропроцессоре 6800) или ACC1, ACC2 и т. д. (как в ЭВМ фирмы Data General). Если число регистров, предназначенных для хранения операндов, больше 4, то эти регистры часто называют **регистрами общего типа**, и тогда отдельным регистрам присваивают такие имена, как регистр общего типа 4, регистр общего типа 8 и т. д.

### 6.3. ПРЕДСТАВЛЕНИЕ ЦЕЛЫХ ЧИСЕЛ

Числа, используемые в цифровых вычислительных машинах, должны быть представлены с помощью таких устройств памяти, как триггеры. Наиболее простой системой представления для двоичных запоминающих устройств является **система представления целых чисел**. На рис. 6.1, *a* показан регистр, состоящий из четырех триггеров  $X_1$ ,  $X_2$ ,  $X_3$  и  $X_4$ , используемых для хранения чисел. Запись величин или состояний триггеров соответствует целому числу, так что  $X_1 = 1$ ,  $X_2 = 1$ ,  $X_3 = 0$ ,  $X_4 = 0$  дает 1100 или десятичное число 12, а  $X_1 = 0$ ,  $X_2 = 1$ ,  $X_3 = 0$ ,  $X_4 = 1$  представляет 0101 или десятичное число 5.

В общем случае необходимо представлять как положительные, так и отрицательные числа, поэтому требуется дополнительный разряд, называемый **разрядом знака**. Обычно он размещается слева от разрядов величины числа. На рис. 6.1, *b*

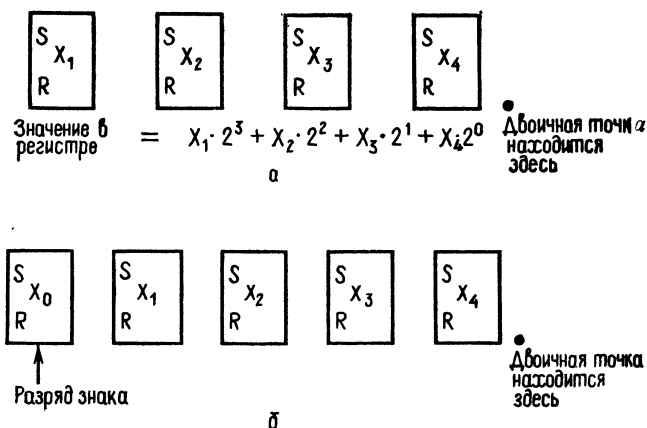


Рис. 6.1. Системы представления.

$a$  — представление целых чисел,  $b$  — система представления величин со знаком.

$X_0$  выбран в качестве разряда знака, а  $X_1, X_2, X_3, X_4$  определяют величину числа. Если  $X_0 = 0$ , то это означает, что число положительно,  $X_0 = 1$  означает, что число отрицательно (это соглашение является общепринятым), так что  $X_0 = 0, X_1 = 1, X_2 = 1, X_3 = 0$  и  $X_4 = 1$  дает положительное число 1101 или 13 в десятичной системе, а  $X_0 = 1, X_1 = 1, X_2 = 1, X_3 = 0$  и  $X_4 = 1$  представляет отрицательное число 1101 или  $-13$  в десятичной форме.

Описанная система называется **целочисленной двоичной системой со знаком**, или **двоичной целочисленной системой величин со знаком**.

Если регистр содержит восемь триггеров, то двоичное число со знаком в рассматриваемой системе будет иметь семь двоичных разрядов для представления величины, или целого числа, и один разряд знака. Поэтому 00001111 соответствует  $+15$ , а 10001111 — числу  $-15$ , так как первые разряды 0 и 1 указывают только знаки  $+$  и  $-$ .

Величины чисел, которые могут быть запомнены в этих двух системах представления на рис. 6.1, таковы:

1. При двоичном целочисленном представлении  $n$ -разрядный триггерный регистр может хранить (десятичные числа) от 0 до  $2^n - 1$ . Таким образом, в шестirazрядном регистре можно хранить числа от 000000 до 111111, где 111111 является двоичным представлением  $63 = 2^6 - 1$ , или  $64 - 1$ .

2. Целочисленная двоичная система представления со знаком имеет диапазон от  $-(2^{n-1} - 1)$  до  $+(2^{n-1} - 1)$  для двоичного регистра. Например, семиразрядный триггерный регистр может хранить числа от  $-111111$  до  $+111111$ , что соот-

ветствует диапазону от  $-63$  до  $+63$  [от  $-(2^6 - 1)$  до  $+(2^6 - 1)$ ].

В следующих разделах будут описаны способы выполнения различных арифметических и логических операций на регистрах.

#### 6.4. ДВОИЧНЫЙ ПОЛУСУММАТОР

Основной элемент, используемый в двоичных арифметических элементах, называется **полусумматором**. Функция полусумматора заключается в сложении двух двоичных цифр, в результате чего образуются сумма и перенос в соответствии с правилами двоичного сложения из табл. 6.1.

Таблица 6.1.

Входные сигналы		Сумма
0 + 0		0
0 + 1		1
1 + 0		1
1 + 1		0 с переносом 1

На рис. 6.2 показана схема полусумматора, который имеет два входа  $X$  и  $Y$  и два выхода  $S$  и  $C$ . Полусумматор выполняет операцию двоичного сложения двух двоичных сигналов, указанных в табл. 6.1. Здесь имеется в виду **арифметическое**, а не логическое, или булево **сложение**.

Как видно из рис. 6.2, полусумматор имеет два входа и два выхода. Если только на одном из входов сигнал равен 1, то выходной сигнал на линии  $S$  будет равен 1. Если оба входных сигнала равны 1, то выходной сигнал на линии  $C$  (перенос) будет равен 1.

Для всех остальных комбинаций входных сигналов перенос равен 0. Эти соотношения можно записать в булевой

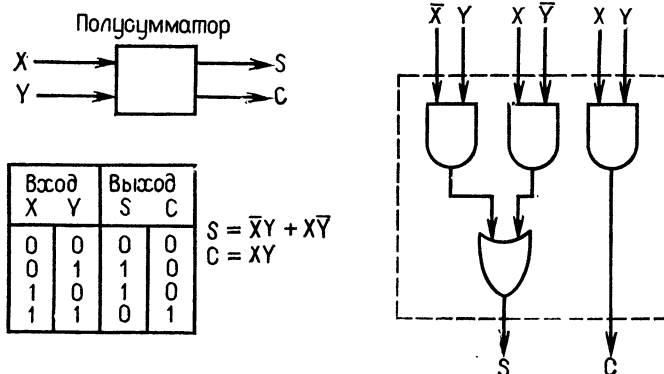


Рис. 6.2. Полусумматор.  $S = \bar{X}Y + X\bar{Y}$ ,  $C = XY$ .

форме:

$$S = X\bar{Y} + \bar{X}Y, \quad C = XY.$$

Четвертьсумматор имеет два входа как полусумматор и только один выход  $S$ . Логическое выражение для этой схемы записывается как  $S = X\bar{Y} + \bar{X}Y$ . Оно соответствует операции ИСКЛЮЧАЮЩЕЕ ИЛИ булевой алгебры.

## 6.5. ОДНОРАЗРЯДНЫЙ СУММАТОР

При сложении более двух двоичных цифр недостаточно использовать несколько полусумматоров, так как полусумматор не имеет входа для учета переносов из других разрядов.

Рассмотрим сложение двух следующих двоичных чисел:

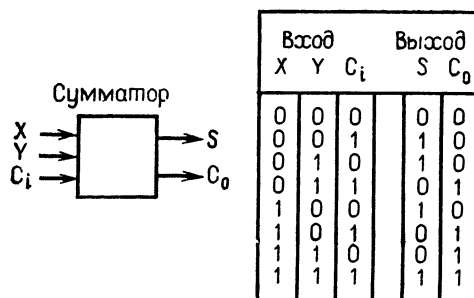
$$\begin{array}{r} + 1011 \\ + 1110 \\ \hline 11001 = \text{сумма} \end{array} \quad \begin{array}{r} + 1011 \\ + 1110 \\ \hline 0101 = \text{частичная сумма} \\ 11 = \text{разряды переноса} \\ \hline 11001 = \text{полная сумма} \end{array}$$

Как видно, цифры переноса, полученные в каждом столбце, должны быть учтены в процессе сложения. Поэтому схема сумматора, способного сложить содержимое двух регистров, должна включать средства для обработки цифр переносов, а также цифр первого и второго слагаемых. Следовательно, каждый разряд многоразрядного сумматора должен иметь три входа, за исключением разряда, соответствующего младшей цифре,— по одному входу для каждого из складываемых чисел и один вход для сигнала переноса, который может возникнуть в предыдущем разряде или передаваться через него.

Условное обозначение **двоичного сумматора**, обрабатывающего эти сигналы переноса, приведено на рис. 6.3, где помещена также таблица входных и выходных сигналов сумматора. Сумматор имеет три входа: входы  $X$  и  $Y$  от соответствующих разрядов регистров, содержимое которых складывается, и вход  $C_i$  для сигнала переноса, возникающего в предыдущем разряде. Двумя его выходами являются  $S$  — для выходного значения поразрядной суммы и  $C_0$  — выход сигнала переноса, который суммируется в следующем разряде. На рис. 6.3 также представлены соотношения между входными и выходными сигналами в булевой форме и булево выражение для выхода  $C_0$  в сокращенном виде.

Как поясняется на рис. 6.4, сумматор может быть построен из двух полусумматоров. Построение сумматора из двух полусумматоров не обязательно оказывается наиболее экономичным методом, и сумматоры проектируются, как правило, на основе соотношений вход-выход, приведенных на рис. 6.3.





$$S = \bar{X}\bar{Y}C_i + \bar{X}Y\bar{C}_i + X\bar{Y}\bar{C}_i + XYC_i.$$

$$C_o = \bar{X}YC_i + X\bar{Y}C_i + XY\bar{C}_i + XYC_i \text{ или}$$

$$C_o = XC_i + XY + YC_i.$$

Рис. 6.3. Сумматор.

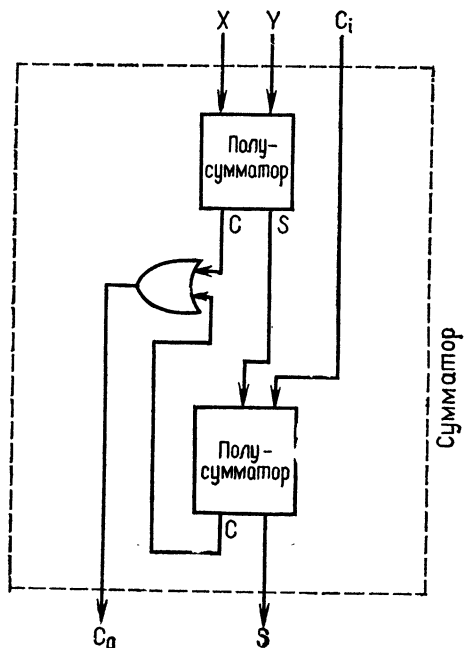


Рис. 6.4 Связь между полусумматором и сумматором.

## 6.6. ПАРАЛЛЕЛЬНЫЙ ДВОИЧНЫЙ СУММАТОР

На рис. 6.5 показан четырехразрядный параллельный двоичный сумматор. Он предназначен для сложения двух четырехразрядных целых двоичных чисел. Входы для первого слагаемого

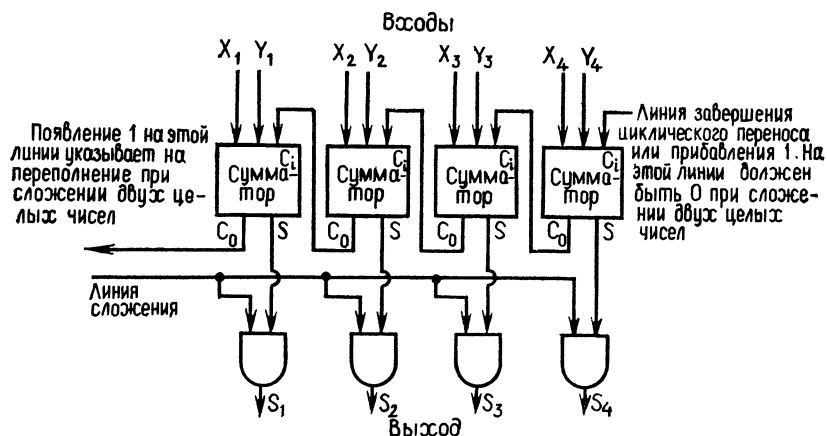


Рис. 6.5. Параллельный сумматор.

го обозначены через  $X_1, \dots, X_4$ , а разряды второго слагаемого представлены входами  $Y_1, \dots, Y_4$ . Этот сумматор не формирует знака суммы двоичных слов и суммирует лишь их абсолютные величины. Для обработки разрядов знака необходимо дополнительное устройство, схема которого зависит от того, представлены ли отрицательные числа в прямом, обратном (система дополнения до единицы) или дополнительном (система дополнения до двух) коде. Эти вопросы будут рассмотрены позже.

Обсудим сложение следующих двух четырехразрядных двоичных чисел:

$$0111, \text{ где } X_1=0, X_2=1, X_3=1, X_4=1,$$

$$0011, \text{ где } Y_1=0, Y_2=0, Y_3=1, Y_4=1.$$

$$\text{Сумма} = 1010$$

Следовательно, поразрядные суммы будут  $S_1=1, S_2=0, S_3=1, S_4=0$ .

Работу сумматора можно проверить следующим образом. Поскольку  $X_4$  и  $Y_4$  — младшие разряды, перенос из предыдущего разряда в них не поступает. В данном примере  $X_4$  и  $Y_4$  оба равны 1, поэтому их сумма равна 0, при этом возникает перенос, который суммируется в сумматоре вместе с разрядами  $X_3$  и  $Y_3$ . Обе цифры  $X_3$  и  $Y_3$  также равны 1, как и цифра переноса в этот разряд. Поэтому поразрядная сумма  $S_3$  принимает значение 1, а на линии ПЕРЕНОС в следующий разряд также появится 1. Так как  $X_2=1$  и  $Y_2=0$ , а на входе переноса будет 1, на линии выхода поразрядной суммы  $S_2$  будет 0 и перенос в следующий разряд будет равен 1. На обоих входах  $X_1$  и  $Y_1$  будет 0, а на входе ПЕРЕНОС в этот разряд сумматора

будет 1. Поэтому выходная линия поразрядной суммы должна представлять 1 и на выходной линии ПЕРЕНОС, обозначенной как «переполнение» на рис. 6.5, будет 0.

Основная конфигурация, приведенная на рис. 6.5, может расширяться до любого числа разрядов. Для построения семиразрядного сумматора можно использовать 7 сумматоров, а для построения двадцатиразрядного сумматора — 20 сумматоров.

Следует отметить, что в четырехразрядном сумматоре линия ПЕРЕПОЛНЕНИЕ может быть использована для получения сигнала пятого разряда. Однако, как правило, этого не делают, поскольку и второе, и первое слагаемые поступают из запоминающего устройства, поэтому их длина равна длине основного слова вычислительной машины. Непосредственное хранение более длинного слова в вычислительной машине невозможно. Ранее уже указывалось, что машина с  $n$ -разрядным словом [включающим разряд знака и  $(n - 1)$  разрядов для обозначения абсолютной величины] может оперировать двоичными числами в пределах от  $(-2^{n-1} + 1)$  до  $(2^{n-1} - 1)$ . Число из этого диапазона называется представимым числом. Так как четырехразрядный простой сумматор на рис. 6.5 не имеет разряда знака, он может представлять лишь числа от 0 до 15. Если в сумматоре сложить 1100 и 1100, то на выходе линии ПЕРЕПОЛНЕНИЕ будет 1, поскольку сумма этих двух чисел равна 11000. Указанное число в десятичной записи есть 24 и поэтому оно не может быть представлено в данной системе. Для этого конкретного регистра весьма малой емкости подобное число называется непредставимым. Если получаемая при сложении двух чисел сумма оказывается непредставимой (т. е. содержит слишком много разрядов), то говорят, что сумма **переполняется** или происходит **переполнение**. Это обстоятельство отмечается цифрой 1 на линии ПЕРЕНОС сумматора старшего разряда. В подобном случае информация о переполнении передается в схему, которая сигнализирует ЭВМ о возникновении переполнения. Обнаружение случаев переполнения является одной из функций арифметического элемента.

Вентили И, соединенные с линиями выходов S четырех сумматоров, используются для передачи суммы в соответствующий регистр.

## 6.7. ПОЛОЖИТЕЛЬНЫЕ И ОТРИЦАТЕЛЬНЫЕ ЧИСЛА

Когда числа записываются в десятичной системе, часто число представляется в виде абсолютной величины, которой предшествует знак + или —, указывающий на то, является оно положительным или отрицательным. Поэтому +125 является положительным числом 125, а —125 — отрицательным числом

125. Обычно также обозначают двоичные числа:  $+111$  — положительное число 7, а  $-110$  — отрицательное число 6. Для обработки как положительных, так и отрицательных чисел ЭВМ должна иметь некоторые средства различения положительных и отрицательных чисел. Как уже отмечалось, машинное слово содержит разряд знака, как правило предшествующий старшему разряду в машинном слове. В описываемых ниже системах 1 в разряде знака соответствует отрицательному числу, а 0 в разряде знака — положительному числу.

Выше было рассмотрено представление числа в виде знака и абсолютной величины. Однако чаще используются системы записи в обратном и дополнительном кодах. В настоящее время дополнительный код используется наиболее часто. Преимущество такого представления отрицательных чисел состоит в том, что сумма и разность как положительных, так и отрицательных чисел могут вычисляться с помощью только одного уже описанного типа сумматора.

Существуют три основные системы записи:

1. Отрицательные числа могут храниться в прямом коде. Поэтому двоичное число  $-0011$  будет храниться в виде  $1.0011$ , где 1 указывает на то, что это число отрицательное, а  $0011$  обозначает его абсолютную величину.

2. Для записи отрицательного числа можно воспользоваться обратным кодом. При этом двоичное число  $-0111$  будет представлено в виде  $1.1000$ , где 1 указывает на то, что число отрицательное.  $1000$  является дополнением его абсолютной величины до 1. (Дополнение абсолютной величины до 1 получается путем отрицания каждого ее бита.)

3. Для представления отрицательных двоичных чисел можно использовать дополнительный код. Например, число  $-0111$  будет храниться как  $1.1001$ , где 1 в разряде знака показывает, что число отрицательное,  $1001$  — дополнение его абсолютной величины до двух. (Дополнение до двух формируется путем отрицания каждого бита абсолютной величины  $0111$ , что дает  $1000$ , и прибавлением 1 к младшему биту, в результате чего получается  $1001$ .)

## 6.8. СЛОЖЕНИЕ В СИСТЕМЕ С ОБРАТНЫМ КОДОМ

Система с обратным кодом для представления отрицательных чисел часто применяется в параллельных двоичных машинах. В основном это объясняется простотой получения обратного кода, так как требуется только инвертирование каждого бита двоичного числа, хранящегося в триггерном регистре. Перед тем как рассмотреть вопросы реализации сумматора для системы с обратным кодом, отметим четыре основных возможных

случая, которые могут иметь место при сложении различных комбинаций положительных и отрицательных чисел.

1. Когда одно положительное число суммируется с другим положительным числом, складываются все разряды, включая разряд знака. Так как оба разряда знака будут нулевыми, разряд знака сумматора остается в состоянии 0. Приведем пример сложения двух четырехразрядных положительных чисел.

Обычное обозначение	Машинное слово
$\begin{array}{r} +0011 \\ +0100 \\ \hline +0111 \end{array}$	$\begin{array}{r} 0.0011 \\ 0.0100 \\ \hline 0.0111 \end{array}$

2. Сумма положительного и отрицательного чисел может оказаться либо положительной, либо отрицательной. Если положительное число по абсолютной величине больше, то сумма будет положительной; если же отрицательное число по абсолютной величине больше, то сумма будет отрицательной. В системе с обратным кодом результат будет правильным и тогда, когда сумма этих двух чисел отрицательна. В этом случае при их сложении переполнения не возникнет. Например,

$\begin{array}{r} +0011 \\ -1100 \\ \hline -1001 \end{array}$	$\begin{array}{r} 0.0011 \\ 1.0011 \\ \hline 1.0110 \end{array}$
---	--

На выходе сумматора будет 10110, причем последние 4 бита составляют дополнение 1001 до 1. Разряд знака, равный 1, также верный, он указывает на то, что число отрицательно.

3. Если положительное число по модулю больше отрицательного, то сумма, полученная до циклического переноса, будет неправильной. Прибавление к ней циклического переноса исправит эту сумму. Разряд знака будет равен 0, что соответствует положительному значению суммы.

$\begin{array}{r} +1001 = 0.1001 \\ -0100 = 1.1011 \\ \hline +0101 \quad \underline{0.0100} \\ \quad \quad \quad \rightarrow 1 \\ \quad \quad \quad \hline 0.0101 \end{array}$	$\begin{array}{r} +0011 = 0.0011 \\ -0010 = 1.1101 \\ \hline +0001 \quad \underline{0.0000} \\ \quad \quad \quad \rightarrow 1 \\ \quad \quad \quad \hline 0.0001 \end{array}$
--	--

Покажем, что произойдет, если сложить два числа с равными абсолютными величинами, но с противоположными знаками:

$\begin{array}{r} +1011 = 0.1011 \\ -1011 = 1.0100 \\ \hline 0000 \quad 1.1111 \end{array}$	$\begin{array}{r} +0000 = 0.0000 \\ -0000 = 1.1111 \\ \hline 0000 \quad 1.1111 \end{array}$
---	---

В этих случаях результатом будет отрицательный 0, что верно.

4. Когда суммируются два отрицательных числа, всегда производится циклический перенос, а также перенос из старшего разряда числа в знаковый разряд. Поэтому разряд знака будет равен 1.

$$\begin{array}{rcl}
 -0011 & = & 1.1100 \\
 -1011 & = & 1.0100 \\
 \hline
 -1110 & \longrightarrow & 1.0000 \\
 & \longrightarrow & 1 \\
 \hline
 & & 1.0001
 \end{array}
 \qquad
 \begin{array}{rcl}
 -0100 & = & 1.1011 \\
 -0111 & = & 1.1000 \\
 \hline
 -1011 & \longrightarrow & 1.0011 \\
 & \longrightarrow & 1 \\
 \hline
 & & 1.0100
 \end{array}$$

В каждом рассмотренном случае значение на выходе сумматора будет представлено в обратном коде, разряд знака которого равен 1.

Из изложенного выше следует, что для реализации сумматора, оперирующего четырехразрядными числами со знаком в системе с обратным кодом, к конфигурации на рис. 6.5 достаточно присоединить еще один сумматор. Входы для знака числа будут обозначены через  $X_0$  и  $Y_0$ , и выход  $C_0$  сумматора со входами  $X_1$  и  $Y_1$  будет соединен со входом  $C_i$  нового сумматора для  $X_0$  и  $Y_0$ . Выход  $C_0$  сумматора для  $X_0$  и  $Y_0$  будет соединен со входом  $C_i$  сумматора для  $X_4$  и  $Y_4$ . Значение на выходе  $S_0$  нового сумматора будет определять разряд знака суммы. (Для выявления переполнения потребуются дополнительные вентили, так как в этом сумматоре оно не обнаруживается.)

## 6.9. СЛОЖЕНИЕ В СИСТЕМЕ С ДОПОЛНИТЕЛЬНЫМ КОДОМ

Когда отрицательные числа представляются в системе с дополнительным кодом, операция сложения подобна сложению в системе с обратным кодом. В параллельных ЭВМ число в дополнительном коде, которое хранится в регистре, может быть получено сначала образованием обратного кода содержимого регистра, а затем прибавлением 1 к младшему разряду регистра. Этот процесс состоит из двух шагов и поэтому на него затрачивается больше времени, чем для преобразования числа в обратный код. Однако преимуществом системы с дополнительным кодом является то, что в ней нет циклических переносов при сложении.

Когда складываются два числа, записанные в системе с дополнительным кодом, возможны следующие четыре случая.

1. Если оба числа положительные, этот случай полностью совпадает со случаем 1, рассмотренным для системы с обратным кодом.

2. Когда одно из чисел положительное, а другое отрицательное, причем положительное число имеет большую абсолютную

величину, возникнет перенос в разряде знака. Этот перенос следует отбросить и на выходе сумматора будет получен правильный результат:

$$\begin{array}{r}
 + 0111 = 0.0111 \\
 - 0011 = + 1.1101 \\
 \hline
 + 0100 \quad \quad 0.0100 \\
 \quad \quad \quad \rightarrow \text{перенос} \\
 \quad \quad \quad \text{отбрасы-} \\
 \quad \quad \quad \text{вается}
 \end{array}$$

$$\begin{array}{r}
 + 1000 = 0.1000 \\
 - 0111 = + 1.1001 \\
 \hline
 + 0001 \quad \quad 0.0001 \\
 \quad \quad \quad \rightarrow \text{перенос} \\
 \quad \quad \quad \text{отбрасы-} \\
 \quad \quad \quad \text{вается}
 \end{array}$$

3. Когда суммируются положительное и отрицательное числа и отрицательное число имеет большую абсолютную величину, в разряде знака не будет переноса и результат окажется правильным:

$$\begin{array}{r}
 + 0011 = 0.0011 \\
 - 0100 = 1.1100 \\
 \hline
 - 0001 \quad 1.1111
 \end{array}
 \qquad
 \begin{array}{r}
 + 0100 = 0.0100 \\
 - 1000 = 1.1000 \\
 \hline
 - 0100 \quad 1.1100
 \end{array}$$

*Замечание.* При преобразовании отрицательного числа в дополнительный код со знаком к его младшему разряду следует прибавить 1. Например,

$$\begin{array}{r}
 1.0011 = 1100 \text{ формирование обратного кода} \\
 \quad \quad 0001 \text{ прибавление 1} \\
 \hline
 - 1101
 \end{array}$$

Если оба числа имеют одинаковые абсолютные величины, то получим результат:

$$\begin{array}{r}
 + 0011 = 0.0011 \\
 - 0011 = 1.1101 \\
 \hline
 0000 \quad 0.0000
 \end{array}$$

Когда складываются положительное и отрицательное числа с одинаковыми абсолютными величинами, то результатом будет положительный 0.

4. Когда складываются два отрицательных числа, в разряде знака, а также в разряде справа от разряда знака образуется перенос. По этой причине разряд знака станет равным 1 и перенос в разряде знака следует отбросить

$$\begin{array}{r}
 - 0011 = 1.1101 \\
 - 0100 = 1.1100 \\
 \hline
 - 0111 \quad 1.1001 \\
 \quad \quad \quad \rightarrow \text{перенос от-} \\
 \quad \quad \quad \text{брасывается}
 \end{array}
 \qquad
 \begin{array}{r}
 - 0011 = 1.1101 \\
 - 1011 = 1.0101 \\
 \hline
 1110 \quad 1.0010
 \end{array}$$

Для параллельных машин операция сложения положительных и отрицательных чисел выполняется довольно просто, по-

скольку переполнение от разряда знака отбрасывается. Таким образом, к параллельному сумматору на рис. 6.5 мы просто присоединяем дополнительный сумматор со входами  $X_0$  и  $Y_0$  и с линией ПЕРЕНОСА  $C_0$  из сумматора для сложения  $X_1$  и  $Y_1$ , соединенной со входом переноса  $C_i$  сумматора для  $X_0$  и  $Y_0$ . Значение на входе  $C_i$  сумматора с входами  $X_4$  и  $Y_4$  полагается равным 0.

Отмеченная простота выполнения операций сложения и вычитания чисел послужила причиной того, что система с дополнительным кодом стала наиболее распространенной для параллельных машин. Если для представления отрицательных чисел используются системы с прямым кодом, эти числа обычно преобразуются в дополнительный код до выполнения операций сложения или вычитания. Затем они преобразуются обратно в прямой код.

## 6.10. СЛОЖЕНИЕ И ВЫЧИТАНИЕ ЧИСЕЛ В ПАРАЛЛЕЛЬНОМ АРИФМЕТИЧЕСКОМ БЛОКЕ

Теперь рассмотрим вентильную схему, предназначенную для сложения или вычитания двух чисел. Эта схема должна иметь входную линию СЛОЖЕНИЕ, входную линию ВЫЧИТАНИЕ, а также линии для ввода чисел, которые следует сложить или вычесть. Когда на линию СЛОЖЕНИЕ подается 1, на линиях вывода должна появиться сумма чисел, когда же на линию ВЫЧИТАНИЕ подается 1, на линиях вывода должна появиться разность чисел. Если на обеих линиях будет 0, то на линиях вывода должны быть 0.

Прежде всего отметим, что если машина способна суммировать как положительные, так и отрицательные числа, то вычитание можно осуществить путем преобразования вычитаемого в обратный или дополнительный код и сложения его с уменьшаемым. Например, результаты выполнения действий  $8 - 4$  и  $8 + (-4)$  одинаковы и результаты действий  $6 - (-2)$  и  $6 + 2$  тоже одинаковы. Поэтому вычитание можно выполнить с помощью арифметического блока, способного складывать числа, посредством преобразования вычитаемого в обратный или дополнительный код и сложения его с уменьшаемым. Например, в системе с обратным кодом могут иметь место четыре случая.

Два положительных числа

$\begin{array}{r} 0.0011 \\ -0.0001 \\ \hline \end{array}$	преобразование вычитаемого в обратный код и сложение его с уменьшаемым	$\begin{array}{r} 0.0011 \\ 1.1110 \\ \hline 0.0001 \end{array}$
		$\begin{array}{r} \phantom{0.0001} \\ \phantom{0.0001} \\ \hline \rightarrow \text{перенос } 1 \end{array}$
		$\begin{array}{r} 0.0010 \end{array}$



## Два отрицательных числа

$\begin{array}{r} 1.1101 \\ -1.1011 \\ \hline \end{array}$	образование обратного кода	$\begin{array}{r} 1.1101 \\ 0.0100 \\ \hline \end{array}$
		$\begin{array}{r} 0.0001 \\ \hline \end{array}$
		→ перенос 1
		$\begin{array}{r} 0.0010 \\ \hline \end{array}$
Положительное уменьшаемое $\begin{array}{r} 0.0010 \\ -1.1101 \\ \hline \end{array}$	=	Отрицательное уменьшаемое $\begin{array}{r} 1.0101 \\ -0.0010 \\ \hline \end{array}$
		$\begin{array}{r} 0.0010 \\ \hline \end{array}$
		→ перенос 1
		$\begin{array}{r} 1.0010 \\ \hline \end{array}$
		1.0011

Те же самые основные правила применимы к вычитанию в системе с дополнительным кодом, за исключением того, что перенос, возникший в сумматорах разрядов знака, просто отбрасывается. В этом случае образуется дополнительный код вычитаемого и затем вычитаемое в дополнительном коде складывается с уменьшаемым без циклического переноса.

Теперь рассмотрим вопрос реализации комбинированной схемы сумматора и вычитателя. Сначала следует представить число, которое вычитается в обратном или дополнительном коде. Это можно сделать несколькими способами. Если запоминающий регистр состоит из триггеров, то обратный код числа может быть получен простым соединением, обеспечивающим подачу инверсного значения каждого входа в сумматор. Когда складываются два числа, для образования дополнительного кода к младшему разряду следует прибавить 1, что осуществляется соединением, обеспечивающим подачу 1 на вход ПЕРЕНОС младшего разряда сумматора.

На рис. 6.6 показана полная логическая схема для сложения или вычитания двух чисел, представленных в дополнительном коде. Одно число представлено разрядами  $X_0, X_1, X_2, X_3$  и  $X_4$ , а другое — разрядами  $Y_0, Y_1, Y_2, Y_3$  и  $Y_4$ . Имеются два управляющих сигнала: сигнал СЛОЖЕНИЕ и сигнал ВЫЧИТАНИЕ. Если ни один из управляющих сигналов не равен 1 (т. е. оба они равны 0), то на всех выходах  $S_0, S_1, S_2, S_3, S_4$  пяти сумматоров будут 0. Если управляющий сигнал СЛОЖЕНИЕ равен 1, то сумма  $X$  и  $Y$  появится на выходах  $S_0, S_1, S_2, S_3, S_4$ . Если управляющий сигнал ВЫЧИТАНИЕ равен 1, то разность между  $X$  и  $Y$  появится на выходах  $S_0, S_1, S_2, S_3, S_4$ .

Отметим, что цепочка вентилях И — ИЛИ, связанная с каждым из входов  $Y$ , производит выбор либо  $Y$  либо  $\bar{Y}$ , так что

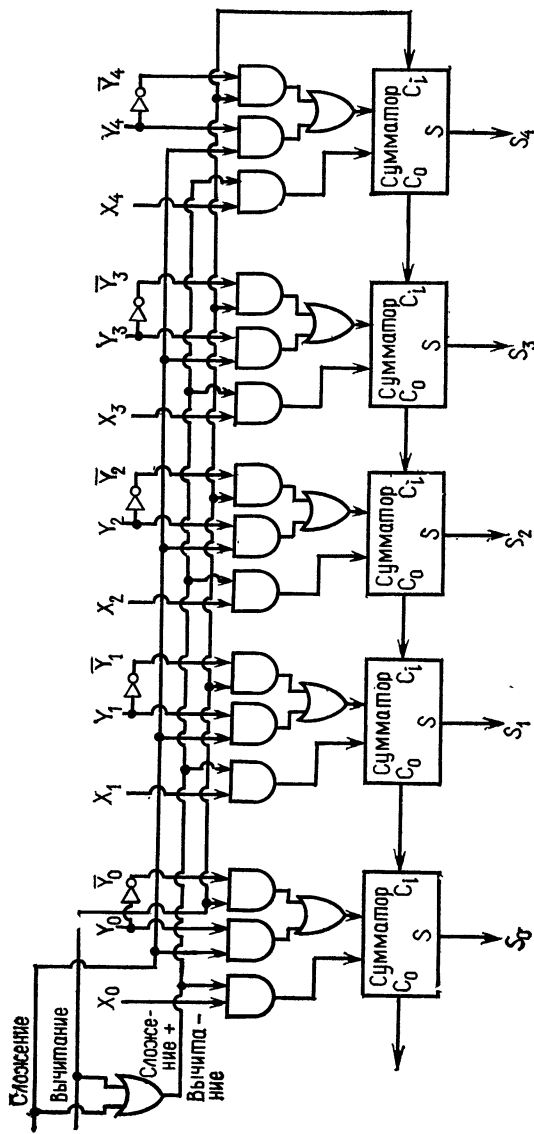


Рис. 6.6. Параллельное сложение и вычитание

Чтобы сложить, сигнал на линии СЛОЖЕНИЕ должен равняться 1. Чтобы вычесть, сигнал на линии ВЫЧИТАНИЕ должен равняться 1. Числа должны быть представлены в дополнительном коде.

сигнал СЛОЖЕНИЕ приводит к тому, что на входе соответствующего сумматора появится сигнал  $Y_1$ , а сигнал ВЫЧИТАНИЕ приводит к тому, что на входе сумматора будет  $\bar{Y}_1$ .

Для выполнения операции сложения или вычитания каждый вход  $X$  соединяется с соответствующим сумматором. Когда требуется выполнить операцию вычитания, в сумматор подается инвертированное значение с каждого выхода триггера  $Y$  и прибавляется 1 при подаче сигнала ВЫЧИТАНИЕ на вход  $C_i$  сумматора для младших разрядов  $X_4$  и  $Y_4$ . Так как при сложении сигнал на линии ВЫЧИТАНИЕ будет равен 0, то и перенос на этой линии будет равен 0. Простота работы схемы на рис. 6.6 дает основание считать систему с дополнительным кодом, в которой выполняются сложение и вычитание, очень удобной; в настоящее время она используется наиболее часто.

Конфигурация на рис. 6.6 наиболее часто применяется для сложения и вычитания, так как она позволяет довольно просто производить сложение и вычитание положительных и отрицательных чисел. Часто линии  $S_0, S_1, \dots, S_4$  соединяются при помощи вентилях И со входами триггеров  $X$ , так что исходное значение  $X$  замещается суммой или разностью чисел  $X$  и  $Y$ .

Важным вопросом, требующим рассмотрения, является **переполнение**. В ЭВМ переполнение возникает тогда, когда результат выполнения какой-либо операции не уместается в регистре (или регистре памяти). Поскольку регистры на рис. 6.6 имеют один разряд для знака числа и четыре разряда для представления абсолютной величины числа, они могут хранить числа от  $+15$  до  $-16$  в дополнительном коде. Поэтому если результат сложения или вычитания больше чем  $+15$  или меньше чем  $-16$ , то говорят, что возникло переполнение. Пусть требуется сложить  $+8$  с  $+12$ , в результате получится 20; это число нельзя (точно) представить в дополнительном коде на линиях  $S_0, S_1, S_2, S_3$  и  $S_4$ . То же самое будет иметь место, если сложить  $-13$  с  $-7$  или вычесть из  $+12$  число  $-8$ . В каждом случае для обнаружения переполнения и оповещения управляющего блока ЭВМ используется логическая схема. Возможные варианты осуществления этой схемы зависят от типа выполняемой команды. (Иногда в программах с удвоенной точностью используются преднамеренные переполнения. Это встречается также при умножении и делении.)

## 6.11. КОНСТРУКЦИИ СУММАТОРОВ

Сумматор является основным элементом арифметического устройства. На рис. 6.3 приведены условное обозначение сумматора, а также таблица комбинаций входных и выходных значений и логические выражения выходных сигналов линий СУМ-

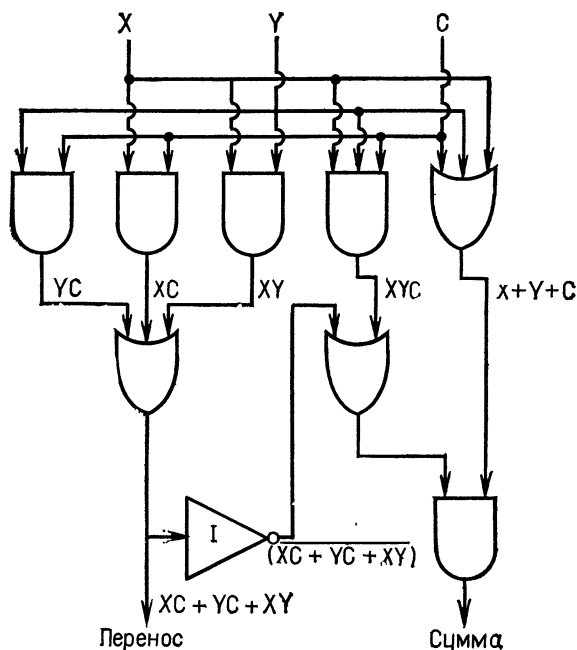


Рис. 6.7. Сумматор, используемый в ЭВМ фирмы IBM.

$$[(XC + YC + XY) + XYC] (X + Y + C) = \bar{X}\bar{Y}C + \bar{X}Y\bar{C} + X\bar{Y}\bar{C} + XYC.$$

МА и ПЕРЕНОС. Заметим, что в системе параллельного сложения для каждого бита основного слова требуется по одному сумматору.

Конечно, имеется множество различных схемных реализаций двоичных сумматоров. Ниже описаны сумматор IBM и микросхема со средним уровнем интеграции, содержащая два сумматора.

**1. Двоичный сумматор.** На рис. 6.7 показана конфигурация двоичного сумматора, используемого в некоторых ЭВМ общего назначения фирмы IBM. Схема имеет три входа:  $X$  — от одного из запоминающих элементов накапливающего сумматора,  $Y$  — от соответствующего запоминающего элемента в регистре, содержимое которого следует сложить с содержимым регистра накапливающего сумматора, вход ПЕРЕНОС от сумматора для следующего младшего бита. Выходами схемы являются выход СУММА и выход ПЕРЕНОС. На выходе СУММА будет значение суммы для рассматриваемого конкретного разряда (поразрядная сумма). Выход ПЕРЕНОС будет соединен со входом ПЕРЕНОС в следующем младшем разряде сумматора (рис. 6.5).

Сигналы выхода трех вентилей И, непосредственно соединенных со входами  $X$ ,  $Y$  и  $C$ , логически суммируются с помощью вентиля ИЛИ, расположенного ниже. Если на входных линиях  $X$  и  $Y$ , либо  $X$  и  $C$ , либо  $Y$  и  $C$  будут 1, то на выходе ПЕРЕНОС также будет 1. Значение на выходе указанной схемы, записанное в виде логического выражения, показано на рисунке. Его можно сравнить с выражением, приведенным на рис. 6.3.

Получение выражения для сигнала на выходе СУММА несколько сложнее. Выражение для сигнала на выходе ПЕРЕНОС  $XY + XC + YC$  сначала инвертируется (применяется операция отрицания), что дает  $\overline{(XY + XC + YC)}$ . К этому выражению логически прибавляется логическое произведение  $X$ ,  $Y$  и  $C$ , образованное вентилем И:

$$\overline{(XY + XC + YC)} + XYC.$$

Логическое умножение суммы  $X$ ,  $Y$ ,  $C$  на полученное выражение дает

$$[\overline{(XY + XC + YC)} + XYC](X + Y + C).$$

После раскрытия скобок и упрощения получается выражение  $X\bar{Y}C + X\bar{Y}\bar{C} + X\bar{Y}C + XYC$ , которое приведено на рис. 6.3. Анализируя это логическое выражение для различных значений на входах, установим, что значение на выходе СУММА будет равно 1 только тогда, когда одно из входных значений равно 1 или если все три входных значения равны 1. Для остальных комбинаций входных значений выходное значение будет равно 0.

**2. Микросхема с двумя сумматорами.** На рис. 6.8 показаны два сумматора. В этой интегральной схеме используется транзисторно-транзисторная логика (ТТЛ). Вся схема конструктивно реализована в виде одной микросхемы. Максимальная задержка от момента изменения сигнала на входе до момента изменения сигнала на выходе  $S$  имеет порядок 8 нс. Максимальная задержка между входом и выходом  $C2$  приблизительно составляет 8 нс.

Величина задержки, связанная с каждым переносом, является важным показателем при оценке сумматора для параллельной системы, так как полное время, требуемое для сложения двух чисел, определяется максимальным временем распространения сигнала переноса через сумматоры. Например, если 01111 суммируется с 10001 в системе с дополнительным кодом, то сигнал переноса, порожденный единицами в младших разрядах каждого числа, должен распространиться через четыре разряда переноса и разряд суммы, прежде чем сумму можно

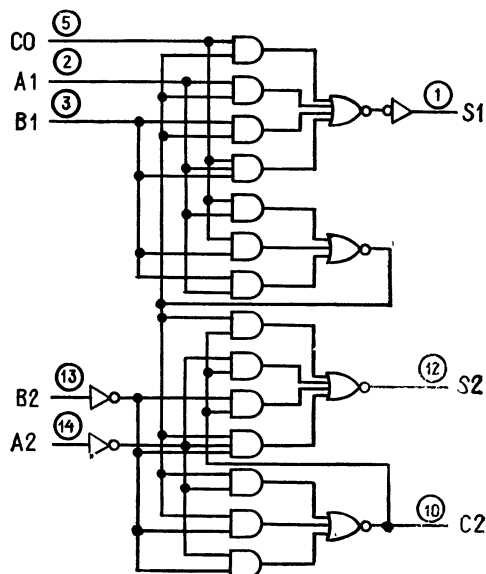


Рис. 6.8. Два сумматора в микросхеме (предоставлено фирмой Texas Instruments).

будет надежно ввести в накапливающий сумматор. Это станет ясно при рассмотрении сложения указанных чисел с помощью сумматора, показанного на рис. 6.5. Данная проблема называется **проблемой сквозного переноса**.

Имеется ряд методов, используемых в быстродействующих машинах, которые облегчают решение этой проблемы. Наиболее часто используется схема, которая прослеживает перенос, появляющийся в ряде разрядов одновременно и затем подает этот перенос в следующий разряд.

## 6.12. ДВОИЧНО-ДЕСЯТИЧНЫЙ СУММАТОР

Арифметические блоки, выполняющие операции над числами, хранящимися в двоично-десятичной форме, должны иметь возможность суммировать четыре бита, которые представляют десятичную цифру. С этой целью используется двоично-десятичный сумматор. На рис. 6.9 показано символическое обозначение сумматора. Он имеет вход разряда второго слагаемого, состоящий из четырех линий, вход разряда первого слагаемого, также состоящий из четырех линий, вход десятичного переноса, выход переноса и выход поразрядной суммы, содержащий четыре линии. Разряды второго и первого слагаемых, а также пораз-

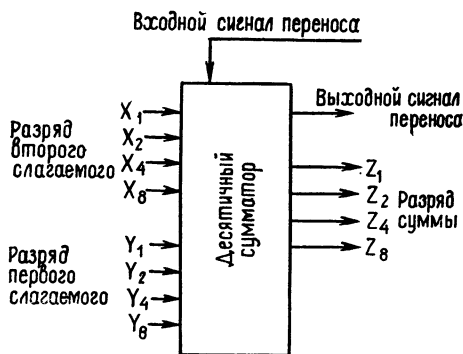


Рис. 6.9. Последовательно-параллельное сложение

рядные суммы представлены в двоично-десятичном коде (8, 4, 2, 1).

Двоично - десятичный сумматор на рис. 6.9 предназначен для сложения цифр первого и второго слагаемых и значения на входе десятичного переноса, с тем чтобы вычислить поразрядную сумму и десятичный перенос разряда. Такой сумматор можно построить, используя сумматоры и вентили И или ИЛИ (рис. 6.10).

Двоично-десятичный сумматор имеет восемь входов: четыре входа  $X_i$  для второго слагаемого и четыре входа  $Y_i$  для первого слагаемого. На каждом из этих входов будет 0 или 1 при выполнении операции сложения. Если складываются 3 (0011) и 2 (0010), то  $X_8 = 0$ ,  $X_4 = 0$ ,  $X_2 = 1$  и  $X_1 = 1$ ;  $Y_8 = 0$ ,  $Y_4 = 0$ ,  $Y_2 = 1$ ,  $Y_1 = 0$ .

Основной сумматор на рис. 6.10 состоит из четырех двоичных сумматоров, представленных в верхней части рисунка, и выполняет сложение по основанию 16, когда требуется выполнить сложение по основанию 10. Поэтому следует учитывать особенности 1) образования переносов и 2) исправления поразрядных сумм, больших 9. Например, если  $3_{10}$  (0011) складывается с  $8_{10}$  (1000), то результатом будет  $1_{10}$  (0001) с образованием переноса.

Схема, определяющая, когда именно перенос должен быть передан в следующие старшие разряды слагаемых, состоит из группы вентилей, соединенных с выходами  $S$  сумматоров 8, 4, 2, и выходом  $C$  сумматора 8.

Изучение процесса сложения приводит к выводу, что перенос должен возникать тогда, когда на выходах суммы 8 И 4, или 8 И 2, или 8 И 4 И 2 сумматора по основанию 16 будут единицы, или же когда на выходе ПЕРЕНОС из сумматора, соответствующего позиции восьмерки, будет 1. (Этот случай имеет место при сложении цифр 8 или цифр 9.)

Всегда, когда сумма двух десятичных цифр превосходит 9, на линии ПЕРЕНОС В СЛЕДУЮЩИЙ СТАРШИЙ СУММАТОР будет 1 для сумматора, изображенного на рис. 6.10.

Новые трудности возникают при образовании переноса. Если сложить  $7_{10}$  (0111) с  $6_{10}$  (0110), возникает перенос, но на

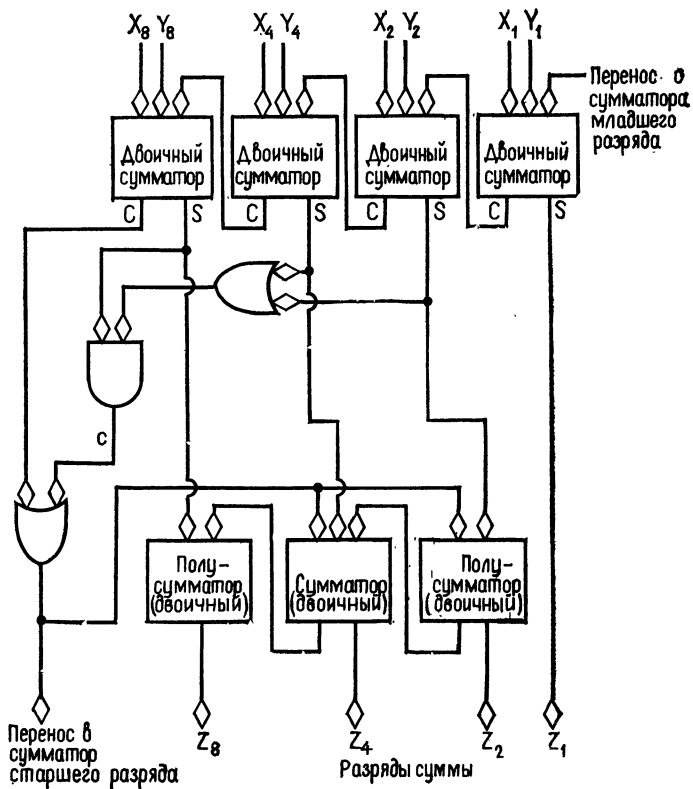


Рис. 6.10. Двоично-десятичный сумматор.

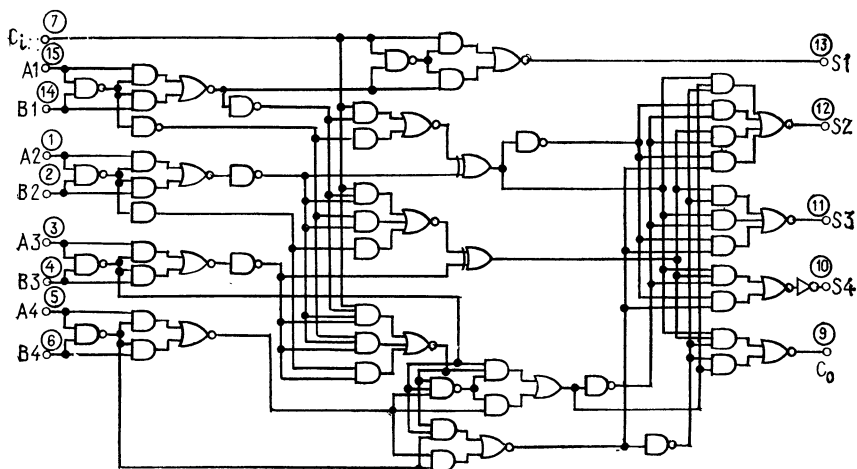


Рис. 6.11. ИС двоично-десятичного сумматора.



выходе сумматора по основанию 16 будет 1101. Этот результат не соответствует десятичным цифрам, допустимым в системе (8, 4, 2, 1)-кодов, и должен быть исправлен. Метод его исправления заключается в прибавлении числа  $6_{10}(0110)$  к сумме, получаемой в сумматорах по основанию 16 всякий раз, когда возникает перенос. Сложение при образовании переноса выполняется путем прибавления единиц на выходных линиях сумматора по основанию 16, позиции которых имеют веса 4 и 2. Эта операция осуществляется двумя полусумматорами и сумматором, расположенными в нижней части рис. 6.10. По существу в этом случае сумматор выполняет сложение по основанию 16 и исправляет сумму, если она больше 9, путем прибавления к ней 6. Ниже приведен ряд примеров

$$\begin{array}{r}
 \begin{array}{l}
 8 + 7 = 15 \quad 1000 + 0111 = \\
 \\
 9 + 5 = 14
 \end{array}
 \qquad
 \begin{array}{r}
 \begin{array}{cccc}
 (8) & (4) & (2) & (1) \\
 1 & 1 & 1 & 1 \\
 +0 & 1 & 1 & 0 \\
 \hline
 1 & 0 & 1 & 0 & 1 = 5 \\
 \uparrow & \text{с образованием переноса} \\
 \begin{array}{cccc}
 (8) & (4) & (2) & (1) \\
 1 & 0 & 0 & 1 \\
 0 & 1 & 0 & 1 \\
 \hline
 1 & 1 & 1 & 0 \\
 +0 & 1 & 1 & 0 \\
 \hline
 1 & 0 & 1 & 0 & 0 = 4 \\
 \uparrow & \text{с образованием переноса}
 \end{array}
 \end{array}
 \end{array}$$

На рис. 6.11 показана микросхема двоично-десятичного сумматора. Его входами являются разряды  $A$  и  $B$ , выходами —  $S$ . В него включены линии ввода и вывода переноса. Микросхема реализована на КМОП-транзисторах.

### 6.13. ПОЛОЖИТЕЛЬНЫЕ И ОТРИЦАТЕЛЬНЫЕ ЧИСЛА В ДВОИЧНО-ДЕСЯТИЧНОЙ ФОРМЕ

Методы обработки чисел в двоично-десятичной форме очень похожи на методы обработки двоичных чисел. Для указания того, является ли число положительным или отрицательным, используется разряд знака. Существуют три метода представления отрицательных чисел. Безусловно, самым простым из них является метод записи отрицательного числа в прямом коде, т. е. в виде абсолютной величины и знака, так что  $-645$  записывается как 1.645. Двумя другими методами являются методы представления в обратном (дополнениями до 9) и дополнительном (дополнениями до 10) кодах, которые подобны обратному и дополнительному кодам в двоичной системе.



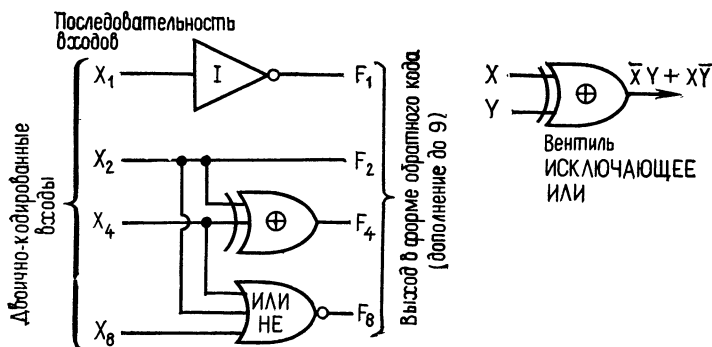


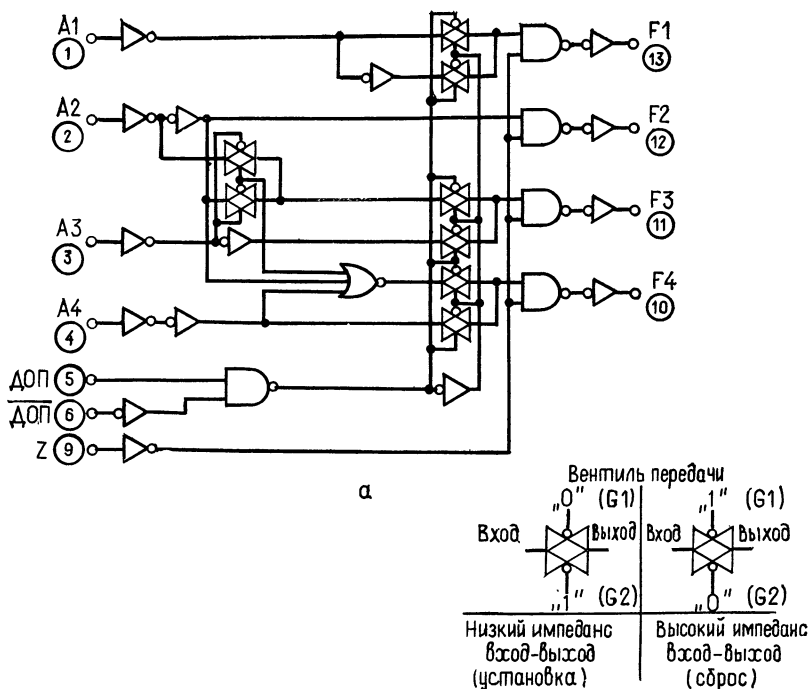
Рис. 6.12. Логическая схема для образования обратного кода (дополнения до 9) в двоично-десятичном коде (8, 4, 2, 1).

нии числа. Поэтому необходимо воспользоваться вентильным блоком, называемым **схемой образования обратного кода**.

Для иллюстрации типичной схемы, которая может использоваться при образовании обратных кодовых наборов для чисел в двоично-десятичной системе, на рис. 6.12 показана логическая схема, которая формирует дополнение до 9 кодового набора, представляющего десятичное число в двоично-десятичном коде (8, 4, 2, 1). У этой схемы четыре входа:  $X_1$ ,  $X_2$ ,  $X_4$  и  $X_8$ . Каждому входу приписан вес, отличный от весов остальных:  $X_1$  имеет вес 1,  $X_2$  — вес 2,  $X_4$  — вес 4,  $X_8$  — вес 8. Если цифры на входах представляют десятичный разряд числа, для которого нужно образовать дополнение, цифры на выходах будут представлять дополнения до 9 входных цифр. Например, если на входе будет 0010 (десятичное 2), то на выходе будет 0111 (десятичное 7), являющееся дополнением входного значения до 9.

На рис. 6.13 показана интегральная схема, предназначенная для образования обратного кода (дополнения числа до 9). В ней использованы КМОП-транзисторы и поэтому она содержит как вентили передачи, так и обычные вентили. (Это не должно беспокоить пользователя схемы, так как он в первую очередь заинтересован в функционировании схемы.) Когда на входе ДОП будет 1, на выходах  $F1$ - $F4$  будет обратный код цифры десятичного разряда, поданного на входы  $A1$ - $A4$ , если же на входе ДОП будет 0, то значения со входов  $A1$ - $A4$  без изменения передаются на  $F1$ - $F4$ .

На рис. 6.14 показано двоично-десятичное суммирующе-вычитающее устройство, которое может быть построено посредством соединения интегральных схем, представленных на рис. 6.11 и 6.13. Из рис. 6.14 видно, что микросхема двухразряд-



Десятичное представление значений на выходе	Входы				Десятичное представление значений на выходе	Выходы			
	A4	A3	A2	A1		F1	F2	F3	F4
0	0	0	0	0	9	1	0	0	1
1	0	0	0	1	8	1	0	0	0
2	0	0	1	0	7	0	1	1	1
3	0	0	1	1	6	0	1	1	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	4	0	1	0	0
6	0	1	1	0	3	0	0	1	1
7	0	1	1	1	2	0	0	1	0
8	1	0	0	0	1	0	0	0	1
9	1	0	0	1	0	0	0	0	0
10	1	0	1	0	9	0	1	1	1
11	1	0	1	1	8	0	1	1	0
12	1	1	0	0	7	0	1	0	1
13	1	1	0	1	6	0	1	0	0
14	1	1	1	0	5	0	0	1	1
15	1	1	1	1	4	0	0	1	0
					3	0	0	1	1
					2	0	0	1	0

б

Рис. 6.13. ИС образования дополнения до 9.  
а — логическая диаграмма; б — таблица истинности.

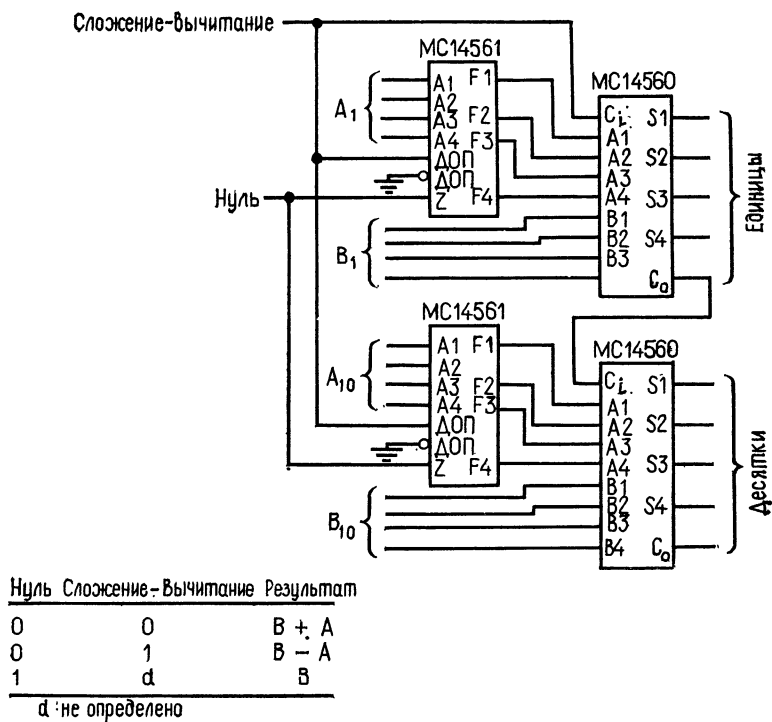


Рис. 6.14. Параллельная суммирующе-вычитающая схема (дополнительный код).

ного суммирующе-вычитающего устройства может использоваться для большого количества разрядов. При сложении входной сигнал СЛОЖЕНИЕ-ВЫЧИТАНИЕ устанавливается равным 1, а при вычитании этот сигнал должен быть равен 0. (Если на вход НУЛЬ подать 1, то значение  $B$  пройдет через схему без изменения.) Как уже было показано, двоично-десятичные числа могут быть также представлены в параллельной форме, но часто используется так называемый **последовательно-параллельный** режим работы. Если десятичное число записано в двоичном коде, результирующее число состоит из группы кодовых наборов, каждый из которых представляет один десятичный разряд. Например, десятичное число 463 в двоично-десятичном коде (8, 4, 2, 1) записывается как 0100 0110 0011. Каждый набор из четырех битов представляет один десятичный разряд. Удобно оперировать кодовыми наборами, представляющими десятичные разряды, как блоками, и обрабатывать их параллельно. В то же время, когда слова для десятич-

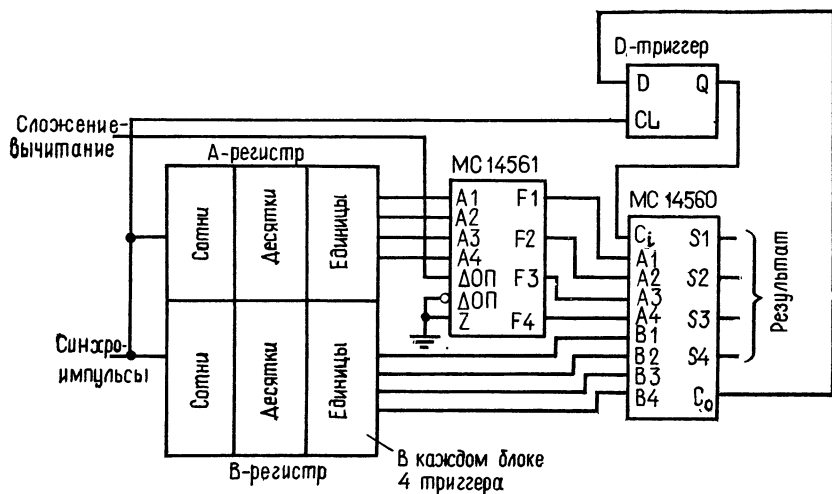


Рис. 6.15. Последовательно-параллельное двоично-десятичное суммирующе-вычитающее устройство, использующее сдвиговый регистр.

ных ЭВМ становятся достаточно длинными, желательно экономно использовать оборудование.

**Последовательно-параллельная** система обеспечивает компромисс, когда кодовые наборы обрабатываются параллельно, а десятичные разряды — последовательно. Для этого требуются четыре линии для каждой цифры, записанной в двоично-десятичном коде (8, 4, 2, 1), причем каждая входная линия имеет вес, отличный от весов других линий. На рис. 6.15 показана блок-схема сумматора, функционирующего в этой системе. Сумматор имеет две группы входов; одна группа состоит из четырех входных линий, по которым подается кодированная цифра второго слагаемого, по другим четырем входным линиям — кодированная цифра первого слагаемого. Эти значения поступают последовательно из регистров *A* и *B*, каждый из которых состоит из четырех сдвиговых регистров; содержимое младших разрядов первого и второго слагаемых в двоично-десятичной форме поступает в первую очередь, а за ним следует содержимое следующих старших десятичных разрядов.

Пусть 324 — первое, а 238 — второе слагаемые и выбран (8, 4, 2, 1)-код. Сигнал СЛОЖЕНИЕ равен 0. Сумматор сначала получит 0100 по линиям первого слагаемого и в то же самое время получит 1000 по линиям второго слагаемого. После первого синхронизирующего импульса значения на этих входах будут заменены на 0010 по линиям первого слагаемого и на 0011 по линиям второго слагаемого. Перед первым

синхронизирующим сигналом на линии суммы будет 0010, а перед вторым синхронизирующим сигналом — 0110. При сложении первых двух цифр десятичных разрядов образуется перенос, который будет задержан и прибавлен с помощью *D*-триггера. Этот процесс продолжается до тех пор, пока не будет проведено сложение цифр каждого из трех десятичных разрядов. Чтобы вычесть *B* из *A*, достаточно подать на вход СЛОЖЕНИЕ-ВЫЧИТАНИЕ сигнал, равный 1, а затем использовать синхронизирующие импульсы.

### 6.15. ОПЕРАЦИЯ СДВИГА

Операция сдвига заключается в перемещении содержимого разрядов, хранимых в регистре, на новые позиции в том же регистре. Существуют две различные операции сдвига: операция сдвига влево и операция сдвига вправо. При операции сдвига влево каждый бит информации, хранящейся в регистре, передвигается влево на некоторое заданное количество разрядов. Пусть следующие шесть двоичных разрядов 000110 хранятся в параллельном двоичном регистре. Если содержимое регистра сдвигается влево на один разряд, то после сдвига регистр будет содержать 001100. Если производится сдвиг вправо на один разряд слова 000110, то после сдвига регистр будет содержать 000011. Процесс сдвига в десятичном регистре осуществляется аналогичным образом: если регистр содержит 0.01234, то после сдвига вправо на один разряд он будет содержать 0.00123 или после сдвига влево на один разряд содержимым регистра будет 0.12340. Операция сдвига используется в командах УМНОЖЕНИЕ и ДЕЛЕНИЕ большинства

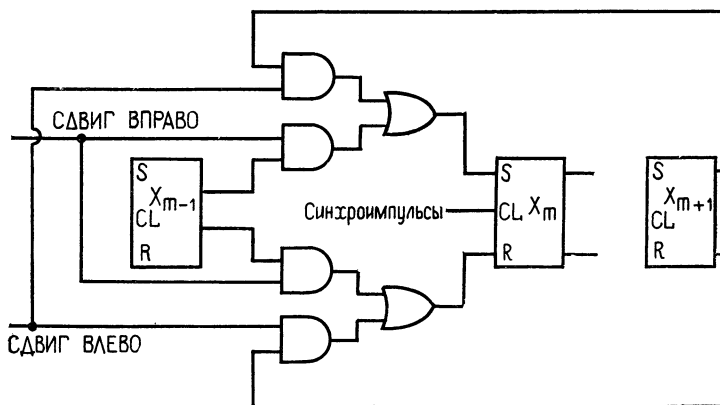


Рис. 6.16. Каскады регистра, осуществляющего сдвиг вправо и сдвиг влево.

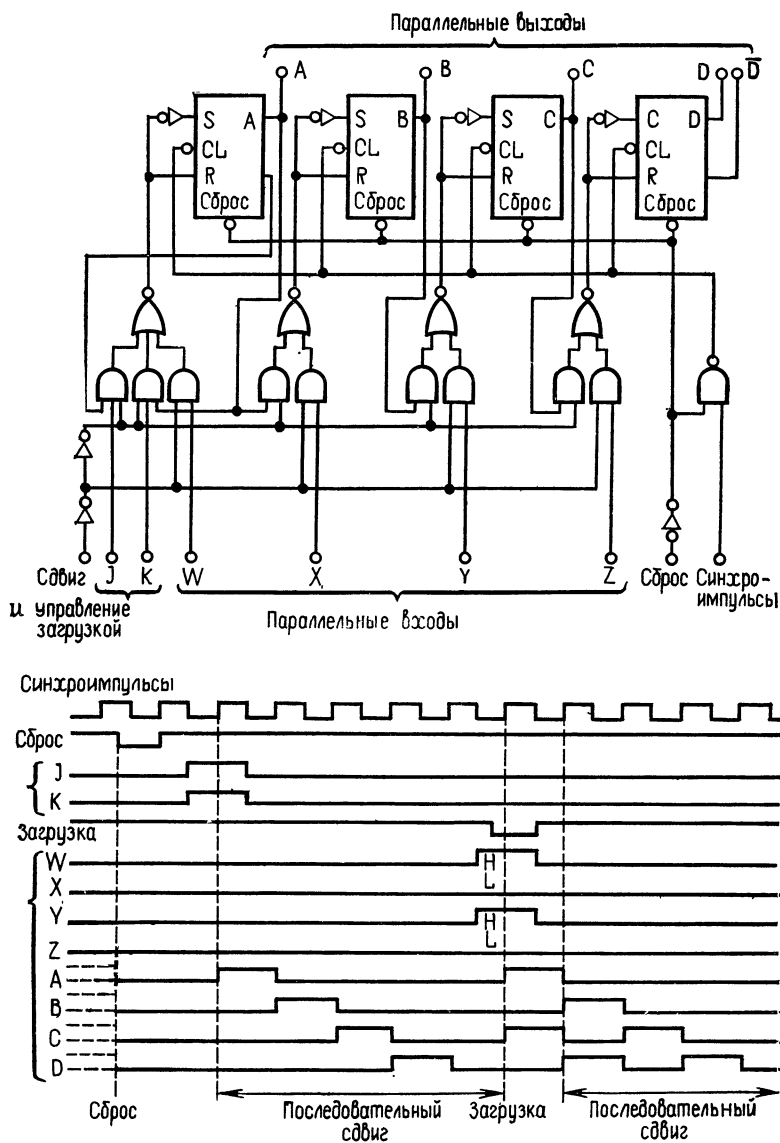


Рис 6.17. Сдвиговый регистр (модель SN74195) с возможностью параллельной загрузки (предоставлено фирмой Texas Instruments).



ЭВМ; кроме того, она может применяться программистами как отдельная команда. Например, машина может иметь команды СДВП и СДВЛ, где буквы представляют мнемоническое обозначение команд СДВИГ-ВПРАВО и СДВИГ-ВЛЕВО.

Блок-диаграмма логической схемы для одного разряда (триггера) в регистре, содержимое которого может быть сдвинуто влево или вправо, показана на рис. 6.16. Из рисунка видно, что, когда сигнал СДВИГ-ВПРАВО равен 1, бит слева сдвигается в  $X_m$ , если сигнал СДВИГ-ВЛЕВО равен 1, то бит справа сдвигается в  $X_m$ .

На рис. 6.17 показана интегральная микросхема с четырьмя триггерами и вентильными схемами; содержимое регистра может сдвигаться как вправо, так и влево, и, кроме того, четыре триггера могут параллельно загружаться с четырех входных линий  $W$ ,  $X$ ,  $Y$  и  $Z$ . Указанные элементы являются транзисторно-транзисторными логическими схемами, которые синхронизируются параллельно. Из интегральных схем этого типа можно получить регистр выбранной длины, содержимое которого может сдвигаться вправо или влево. Этот регистр может также загружаться параллельно.

## 6.16. ОСНОВНЫЕ ОПЕРАЦИИ

Арифметико-логическое устройство ЭВМ состоит из нескольких регистров, в которых может храниться информация, и логических схем, обеспечивающих выполнение определенных операций между регистрами.

Как было показано, данные, хранящиеся в триггерном регистре, могут обрабатываться следующим образом.

1. Регистр может быть сброшен (все триггеры установлены в состояние 0).

2. Содержимое регистра может быть преобразовано в обратный код (дополнение до единицы) или в дополнительный код (дополнение до двух) для двоичных данных, или в дополнение до 9 или до 10 для десятичных данных.

3. Содержимое регистра может быть сдвинуто вправо или влево.

4. Содержимое регистра может быть увеличено или уменьшено на единицу.

Кроме того, были описаны некоторые операции между регистрами, к числу которых относятся:

- 1) передача содержимого одного регистра в другой регистр;

- 2) сложение или вычитание данных, составляющих содержимое двух регистров.

В АЛУ чаще всего выполняются операции или последовательные группы этих двух типов операций. Сложные команды, такие, как умножение и деление, могут потребовать большого числа указанных операций, однако эти команды могут выполняться с использованием лишь последовательности уже рассмотренных простых операций.

Следует подчеркнуть еще один важный момент. Некоторые операции, встречающиеся в командах, являются **условными**: это означает, что данная операция может выполняться или не выполняться в зависимости от значения определенных битов чисел, хранящихся в ЭВМ. Например, может оказаться желательным выполнять умножение, используя только прямой код числа. В этом случае разряды знаков двух сомножителей должны быть проверены управляющей схемой, и если какой-либо из них равен 1, то соответствующее число должно быть преобразовано в дополнительный код до начала умножения. Указанная операция образования дополнительного кода содержимого регистра является условной.

Многие различные последовательности операций могут привести к одному и тому же результату. Например, умножение двух чисел можно заменить на сложение одного из множителей с самим собой столько раз, сколько единиц в другом множителе. Если вручную нужно вычислить  $369 \times 12$ , то это можно сделать путем сложения 369 с самим собой 12 раз. Этот процесс может оказаться трудоемким по сравнению с более простым алгоритмом, который используется при умножении, однако при этом будет получен тот же результат. Аналогичный принцип применим к машинному умножению чисел. Произведение двух чисел может быть вычислено посредством пересылки одного из множителей в счетчик, из которого вычитается единица каждый раз, когда выполняется сложение другого множителя с самим собой до тех пор, пока содержимое счетчика не достигнет 0. Помимо этого, применяются другие более быстрые методы. Они будут описаны ниже.

Для умножения и деления чисел в цифровых машинах используется множество алгоритмов. Особенно сложным процессом является деление, и для его выполнения, в частности в десятичных машинах, применяются разнообразные методы. Выбор метода, используемого в машине, главным образом определяется стоимостью машины и выигрышем в быстроте действия. Почти для всех операций ускорение их выполнения приводит к удорожанию, и, как правило, чем быстрее осуществляется процесс деления, тем дороже машина.

Для пояснения операций двоичных умножения и деления воспользуемся блок-схемой обобщенной двоичной машины. На рис. 6.18 приведена блок-схема регистров АЛУ. Машина имеет три ос-

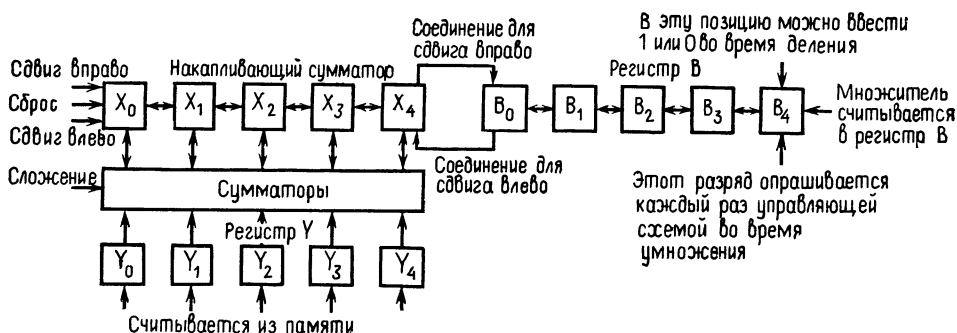


Рис. 6.18. Обобщенный параллельный арифметический элемент.

новых регистра, накапливающий сумматор, регистр  $Y$  и регистр  $B$ . Опишем операции, которые могут выполняться на ней.

1. Очищается накапливающий сумматор. (Установка в нулевое состояние.)

2. Содержимое накапливающего сумматора сдвигается вправо или влево. Накапливающий сумматор и регистр  $B$  могут быть соединены в один длинный сдвиговый регистр. Если содержимое этого регистра сдвинуть вправо на два разряда, то два младших разряда этого сумматора будут сдвинуты в первые две позиции регистра  $B$ . Несколько левых сдвигов сдвинут старшие разряды регистра  $B$  в накапливающий сумматор. Поскольку основное машинное слово состоит из пяти битов, в каждом регистре содержатся пять двоичных запоминающих устройств. Сдвиг вправо на пять позиций приведет к передаче содержимого накапливающего сумматора в регистр  $B$ , а результатом сдвига влево на пять позиций будет передача содержимого регистра  $B$  в этот сумматор.

3. Над содержимым регистра  $Y$  и накапливающего сумматора могут производиться операции сложения или вычитания. Тогда сумма или разность будет храниться в регистре накапливающего сумматора.

4. В регистр  $Y$  считываются слова из устройства памяти. Для передачи слова в накапливающий сумматор необходимо сначала очистить его, затем считать слово из устройства памяти в регистр  $Y$ , а после этого сложить содержимое регистра  $Y$  с содержимым накапливающего сумматора.

Арифметическое устройство, способное выполнять эти операции на своих регистрах, может быть использовано для реализации всех арифметических операций. Можно построить машину, использующую меньше числа операций, однако большинство ЭВМ общего назначения обычно содержат арифметическое устройство, которое может выполнять по крайней мере все эти операции.

### 6.17. ДВОИЧНОЕ УМНОЖЕНИЕ

Процесс умножения одного двоичного числа на другое лучше всего пояснить на примере перемножения двух двоичных чисел;

$$\begin{array}{r}
 1001 = \text{множимое} \\
 1101 = \text{множитель} \\
 \hline
 1001 \\
 0000 \\
 1001 \\
 1001 \\
 \hline
 1110101 = \text{произведение}
 \end{array}
 \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \text{частичные произведения}$$

Важно отметить, что в этом процессе в действительности применяются только два правила умножения одного двоичного числа на двоичную цифру: 1) если эта двоичная цифра (множитель) равна 1, то двоичное число (множимое) просто копируется; 2) если этот множитель равен 0, то частичное произведение равно 0. В приведенном примере эти правила иллюстрируются следующим образом: в первом справа разряде множителя стоит 1, поэтому в качестве частичного произведения копируется значение множимого. В следующем слева разряде множителя стоит 0, и, следовательно, частичное произведение равно 0. Заметим, что при образовании каждого нового частичного произведения оно сдвигается влево на один разряд по отношению к предыдущему частичному произведению. Следующее частичное произведение сдвигается влево на один разряд от него и в том случае, когда частичное произведение равно 0. Этот процесс продолжается до тех пор, пока не будут использованы все разряды множителя; затем полученные частичные произведения суммируются.

Поэтому для реализации умножения чисел рассмотренным способом ЭВМ должна обладать следующими способностями: 1) определять, находится ли в данном разряде множитель 1 или 0, 2) осуществлять сдвиг частичных произведений, 3) выполнять сложение частичных произведений.

Сложение можно начинать, не ожидая формирования всех частичных произведений; они одновременно могут суммироваться парами. Для приведенного выше примера начнем с первых двух частичных произведений

$$\begin{array}{r}
 1001 \\
 0000 \\
 \hline
 01001
 \end{array}$$

К этой сумме нужно затем прибавить следующее частичное произведение, предварительно сдвинув его влево на одну по-

зицию:

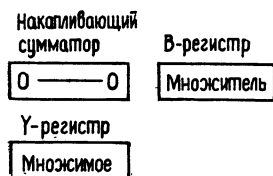
$$\begin{array}{r} 01001 \\ 1001 \\ \hline 101101 \end{array}$$

и, наконец,

$$\begin{array}{r} 101101 \\ 1001 \\ \hline 1110101 \end{array}$$

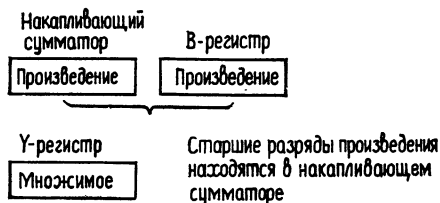
Устройство умножения может быть построено с использованием именно такого способа. По очереди выбирается каждый из разрядов множителя и прибавляется множимое к содержанию некоторого регистра, а затем сдвигается множимое влево при выборе нового разряда множителя; в результате формируется произведение как суммы частичных произведений.

Для рассмотрения типичного метода умножения воспользуемся обобщенным арифметическим устройством, показанным на рис. 6.18. Пусть, как показано ниже, множитель хранится в регистре *B*, множимое — в регистре *Y*, а накапливающий сумматор содержит все нули.



Предположим также, что оба сомножителя положительны. Если хотя бы один из них отрицательный, то до умножения он должен быть преобразован в прямой код. Пусть каждый операнд состоит из  $n$  битов. На рис. 6.18  $n = 4$ .

Необходимый формат результата выполнения операции умножения показан ниже; произведение находится в накапливающем сумматоре и регистре *B*.



Умножение требует выполнения  $n$  элементарных шагов, где  $n$  — число разрядов значащей части чисел, которые перемножаются, и заключительного сдвига произведения вправо.

Каждый элементарный шаг инициируется управляющей схемой, проверяющей правый крайний бит в регистре  $B$ . Он заключается в следующем.

**Элементарный шаг.** Если правый крайний бит в регистре  $B$  равен 0, то содержимое накапливающего сумматора и регистра  $B$  сдвигается вправо на один разряд. Если правый крайний бит в регистре  $B$  равен 1, то число в регистре  $Y$  складывается с содержимым накапливающего сумматора и объединенное содержимое накапливающего сумматора и регистра  $B$  сдвигается вправо на одну позицию.

После каждого элементарного шага опять проверяется новый крайний бит регистра  $B$  и начинается выполнение следующего из  $n$  шагов.

Рассмотрим умножение  $1101 \times 1001$ , которое уже использовалось в предыдущем примере. Здесь  $1101$  — множитель. В данном случае накапливающий сумматор сначала содержит 0.0000, регистр  $B$  — 0.1101, а регистр  $Y$  — 0.1001. Следует выполнить четыре шага и произвести заключительный сдвиг.

**Шаг 1.** Поскольку в крайнем правом разряде регистра  $B$  содержится 1 (в младшем разряде множителя), на первом шаге содержимое регистра  $Y$  прибавляется к содержимому накапливающего сумматора и затем содержимое соединенных вместе накапливающего сумматора и регистра  $B$  сдвигается вправо. Второй младший бит множителя займет правый крайний разряд регистра  $B$  и будет управлять следующей операцией. Регистр  $Y$  все еще будет содержать множимое 0.1001, содержимым накапливающего сумматора будет 0.0100, а регистра  $B$  — 1.0110.

**Шаг 2.** Правый крайний бит регистра  $B$  равен 0, и, поскольку он управляет следующей операцией, включится сигнал СДВИГ-ВПРАВО и содержимое накапливающего сумматора и регистра  $B$  будет сдвинуто вправо, в результате чего в накапливающем сумматоре будет содержаться 0.0010, а в регистре  $B$  — 0.1011.

**Шаг 3.** Теперь правый крайний бит регистра  $B$  равен 1. Поэтому содержимое регистра  $Y$  опять будет суммироваться с содержимым накапливающего сумматора и содержимое соединенных вместе накапливающего сумматора и регистра  $B$  сдвинется вправо. Эти регистры будут содержать 0.0101 и 1.0101 соответственно.

**Шаг 4.** Наименьшим значащим битом регистра  $B$  будет другая единица, так что содержимое регистра  $Y$  опять будет суммироваться с содержимым накапливающего сумматора и содержимое накапливающего сумматора сдвинется вправо. После этого сдвига вправо содержимым соединенных вместе накапливающего сумматора и регистра  $B$  будет 0.011101010.

Результат заключительного сдвига вправо соответствует правильной записи произведения в виде 0.001110101 в рассматриваемой целочисленной системе счисления. Его старшие разряды будут храниться в накапливающем сумматоре, а младшие — в регистре *B*.

Накапливающий сумматор	Регистр <i>B</i>	
0.0000	0.1101	В начале
0.0100	1.0110	После шага 1
0.0010	0.1011	После шага 2
0.0101	1.0101	После шага 3
0.0111	0.1010	После шага 4
0.0011	1.0101	После сдвига вправо

Теперь можно пояснить причину объединения содержимого накапливающего сумматора и содержимого регистра *B*. Произведение двух пятиразрядных чисел со знаком может иметь до девяти значащих разрядов (включая разряд знака), и поэтому потребуется два (а не один) пятиразрядных регистра для хранения произведения. Окончательная запись произведения рассматривается как десятиразрядное число, содержащееся в двух регистрах, причем левые крайние биты (старшие разряды) хранятся в левом регистре, а правые крайние биты (младшие разряды) — в правом регистре, включая наименьший двоичный разряд произведения. Таким образом, в двух соединенных регистрах будет содержаться 0.001110101, которое представляет десятичное число  $+117$ .

Для выполнения опроса разрядов множителя разработана управляющая схема, которая либо сдвигает, либо складывает и сдвигает необходимое число раз, а потом останавливается. В этом случае длина множителя (т. е. содержимое регистра *Y*) равна 4 разрядам плюс заряд знака, поэтому выполняются четыре шага указанного типа. В свою очередь каждый разряд машинного слова, кроме разряда знака, опрашивается в соответствии с общепринятой практикой. Например, если основное машинное слово состоит из 25 разрядов, т. е. 24 разряда представляют величину запоминаемого числа и один разряд указывает знак, то при каждом выполнении умножения машина должна по очереди опросить 24 разряда множителя, выполняя сложение и сдвиг или просто операцию сдвига 24 раза. Ясно, что по этой причине для выполнения операции умножения требуется больше времени, чем для таких операций, как сложение или вычитание. В некоторых параллельных машинах предусмотрено удвоение скорости нормальной работы во время

выполнения операции умножения: если машина выполняет такие операции, как сложение, дополнение, передача и т. д. со скоростью 4 МГц/с для обычных команд, то для операций сложение-и-сдвиг, выполняемых при умножении чисел, скорость увеличивается до 8 МГц/с. В некоторых машинах сдвиг вправо осуществляется во время сложения; это означает, что сумма содержимого накапливающего сумматора с содержимым регистра  $Y$  формируется и сдвигается вправо на одну позицию каждый раз и поэтому отпадает необходимость в выполнении операции сдвига вправо после каждого сложения.

Разряды знаков обоих сомножителей могут обрабатываться несколькими способами. Например, знак произведения можно определить с помощью управляющей схемы до начала процесса умножения. Эта информация о знаке хранится во время выполнения умножения, после чего она помещается в разряд знака накапливающего сумматора и в случае необходимости осуществляется преобразование содержимого накапливающего сумматора в обратный или дополнительный код. Поэтому разряды знака обоих сомножителей определяются в первую очередь; если они оба содержат 0 или 1, то в разряде знака произведения должен быть 0, если только один из них содержит 1, то в разряде знака произведения должна быть 1. Эта информация, хранящаяся в триггере во время умножения, может быть передана в разряд знака позже. Если машина оперирует числами, заданными в обратном и дополнительном коде, то во время выполнения операции умножения оба сомножителя могут обрабатываться как положительные числа, и если какое-либо из них отрицательно, то это число следует преобразовать в прямой код (с положительной абсолютной величиной путем выполнения операции дополнения) до начала умножения. Иногда умножение чисел в обратном или дополнительном коде выполняется с помощью более сложных алгоритмов.

## 6.18. ДЕСЯТИЧНОЕ УМНОЖЕНИЕ

Десятичное умножение является более сложным процессом по сравнению с двоичным умножением. В то время как произведение двоичной цифры на двоичное число либо совпадает с этим числом, либо равно 0, произведение десятичной цифры на десятичное число вычисляется с помощью таблицы умножения с применением операций переноса и сложения. Например,

$$7 \times 24 = 7 \times 4 + 7 \times 20 = 28 + 140 = 168.$$

Даже произведение двух десятичных цифр может состоять из двух десятичных цифр, например  $7 \times 8$  равно 56. В дальнейшем две цифры, которые могут получиться при перемножении



двух десятичных цифр, будем называть **левым** и **правым разрядами**. Таким образом, при перемножении 3 на 6 левым разрядом будет 1, правым разрядом — 8. Для  $2 \times 3$  левый разряд равен 0, правый — 6.

Хотя процесс сложения множимого с самим собой столько раз, сколько единиц в множителе, простой, он требует больших затрат времени. Простейший метод десятичного умножения включает передачу крайнего правого разряда множителя в счетчик, считающий в обратном направлении, сложение множимого с самим собой и одновременное вычитание единицы из содержимого счетчика до тех пор, пока оно не станет равным 0. Частичное произведение, полученное таким образом, должно быть сдвинуто вправо на один десятичный разряд, затем в счетчик вводится следующий разряд множителя, и этот процесс повторяется до тех пор, пока не будут использованы все разряды множителя. Хотя этот метод относительно медленный, но он является ясным и простым.

Указанный процесс можно ускорить путем образования с помощью множимого и правого крайнего разряда множителя, так же как в описанной схеме, за исключением того, что фактически отдельно записываются левые и правые разряды частичных произведений, полученных при умножении десятичной цифры на число, которые затем суммируются. Например, для произведения  $6 \times 7164$  правые разряды частичных произведений образуют число 2664, левые их разряды — число 4032. Их сумма равна

$$\begin{array}{r} 2664 \\ + 4032 \\ \hline 42984 \end{array}$$

Вообще выполнение десятичного умножения на ЭВМ с большой скоростью является сложным процессом. Существует столько же методов умножения двоично-десятичных чисел, сколько и типов имеющихся машин. Производятся микросхемы, содержащие вентиляльную схему, значения на двух входах которой представлены в двоично-десятичной форме, а значение на выходе — двухразрядное десятичное число.

## 6.19. ДЕЛЕНИЕ

Операция деления — наиболее трудная и продолжительная по времени операция, выполняемая АЛУ большинства ЭВМ общего назначения. Хотя на первый взгляд деление не сложнее, чем умножение, возникает ряд проблем, связанных с процессом деления, которые требуют дополнительных шагов с определенными затратами времени.

Деление вручную — это процесс проб и ошибок. Например, если нужно разделить 4610 на 77, мы сначала замечаем, что 77 не «входит» в 46, поэтому пытаемся разделить 461 на 77. Можно предположить, что число 461 в шесть раз больше делителя 77, однако проверка показывает ошибочность этого предположения:

$$\begin{array}{r|l} 4610 & 77 \\ 462 & 6 \\ \hline -1 & \end{array}$$

Следовательно, мы зависили значение частного и необходимо уменьшить цифру первого разряда частного, которую мы пытаемся определить, до значения 5.

Та же проблема возникает, когда ЭВМ пытается разделить одно число на другое этим способом. Она должна пробовать выполнять вычитания на каждом шаге процесса деления и затем проверять, является ли остаток отрицательным. Рассмотрим пример деления 1111 на 11:

$$\begin{array}{r|l} 1111 & 11 \\ 11 & 101 \\ \hline 0011 & \\ & 11 \\ & \hline & 00 \end{array}$$

На любом шаге деления легко определить визуально, равно ли частное 1 или 0, но ЭВМ не обладает этой способностью и должна каждый раз выполнять пробное вычитание. Если после предположения о значении пробного разряда частного и вычитания делителя результат окажется отрицательным, то текущее значение делимого должно быть «восстановлено», или же следует применить какой-либо другой метод деления.

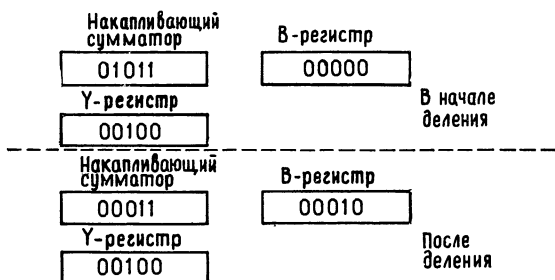
Заслуживает внимания ряд вопросов, касающихся деления двоичных целых чисел с фиксированной точкой. Обычно деление выполняется над двумя двоичными целыми числами со знаком, имеющими одинаковую длину. Результат деления, или частное, хранится как число такой же длины, что делитель или делимое. Остаток также хранится как число такой же длины.

Используя регистры, представленные на рис. 6.18, покажем, как осуществляется деление числа, которое хранится в накапливающем сумматоре, на число, хранящееся в регистре  $Y$ . Частное будет храниться в регистре  $B$ , а остаток — в накапливающем сумматоре. Это наиболее часто используемая структура выполнения операции деления.

Предположим, что регистры  $B$ ,  $Y$ , а также накапливающий сумматор имеют длину в 5 бит (4 бита для представления

величины и один бит знака). До начала операции деления делимое находится в накапливающем сумматоре, делитель — в регистре  $Y$ . После деления частное будет находиться в регистре  $B$ , остаток — в накапливающем сумматоре. Как делитель, так и делимое должны быть положительными числами.

Рассмотрим следующий пример. Сначала содержимое накапливающего сумматора (делимое) равно десятичному числу 11, содержимое регистра  $Y$  (делитель) — четырем. В результате деления частное, равное 2, будет содержаться в регистре  $B$ , остаток, равный 3 — в накапливающем сумматоре.



В двоичных машинах применяются два основных метода деления: метод с восстановлением остатка и метод без восстановления остатка. В первом примере применяется метод с восстановлением остатка.

Как и в случае умножения, при использовании метода деления с восстановлением остатка **элементарный шаг** должен выполняться повторно (в данном случае столько раз, сколько значащих разрядов в вычитаемом).

**Элементарный шаг.** В методе с восстановлением остатка элементарный шаг состоит в выполнении «пробного деления», которое начинается с вычитания содержимого регистра  $Y$  из содержимого накапливающего сумматора. После вычитания выполняется одно из следующих действий.

1. Если получен отрицательный результат, то делитель «не подходит». Поэтому в правый крайний разряд регистра  $B$  помещается 0 и делимое (содержимое накапливающего сумматора) восстанавливается путем сложения делителя с полученной разностью. Затем содержимое регистра  $B$  и содержимое накапливающего сумматора одновременно сдвигаются влево.

2. Если разность неотрицательна, то отпадает необходимость в восстановлении частичного делимого в накапливающем сумматоре, поскольку пробное деление было успешным. Содержимое как накапливающего сумматора, так и регистра  $B$  сдвигается влево и после этого в крайний правый разряд регистра  $B$  помещается 1.

Путем проверки разряда знака в накапливающем сумматоре после каждого вычитания ЭВМ определяет, является ли результат пробного деления положительным или отрицательным.

Чтобы описать всю процедуру, необходимо сначала пояснить, как начинается деление и каковы правила начала и остановки при выполнении элементарных шагов. К сожалению, сложность этих процедур такая же, как сложность определения начала, остановки и установления положения десятичной точки «обычного деления».

1. Как уже отмечалось, если делитель больше делимого, то частное должно равняться 0, а значение остатка — делимому. (Например, если 7 разделить на 17, то частное равно 0, остаток — семи.) Для проверки этого из содержимого накапливающего сумматора следует вычесть делимое, находящееся в регистре  $Y$ , и если их разность отрицательна, то необходимо восстановить содержимое накапливающего сумматора путем сложения содержимого регистра  $Y$  с содержимым накапливающего сумматора. Теперь содержимое регистра  $B$  равно 0, т. е. правильному значению частного; содержимое накапливающего сумматора имеет исходное значение, равное остатку.

2. После этой проверки необходимо выровнять левый крайний бит делителя по отношению к левому крайнему биту делимого путем сдвига влево содержимого делителя и записать число сдвигов, требуемых для указанного выравнивания. Если число сдвигов равно  $M$ , то элементарный шаг следует повторить  $(M + 1)$  раз.

3. Основной (элементарный) шаг повторяется ровно  $(M + 1)$  раз.

4. Наконец, для исправления остатка содержимое накапливающего сумматора должно быть сдвинуто вправо  $(M + 1)$  раз после выполнения последнего элементарного шага. Соответствующие примеры приведены в табл. 6.2 и 6.3. В этих двух примерах отсутствует проверка равенства остатка нулю, т. е. пропущен шаг 1.

На рис. 6.19 приведена блок-схема алгоритма. В более подробной блок-схеме некоторые шаги должны быть разделены, например шаг «сдвиг вправо содержимого накапливающего сумматора  $(M + 1)$  раз» должен быть представлен в виде единичных сдвигов, выполняемых в цикле, который управляется счетчиком. В тех случаях, когда рассматриваются достаточно сложные алгоритмы, как этот алгоритм, удобно составить блок-схему алгоритма до попытки реализации схемы управления.

При делении разряды знака обрабатываются точно так же, как и при умножении. На первом шаге как делитель, так и де-

Таблица 6.2.

Регистр <i>B</i>	Накапливающий сумматор	Регистр <i>Y</i>	Замечания
0.0000	0.0110	0.0011	Разделить 6 на 3
0.0000	0.0110	0.0110	Содержимое регистра <i>Y</i> сдвигается влево один раз для выравнивания единиц в накапливающем сумматоре и регистре <i>Y</i> . Элементарный шаг должен выполняться два раза
0.0000	0.0000	0.0110	Содержимое регистра <i>Y</i> вычитается из содержимого накапливающего сумматора. Результат равен 0, поэтому содержимое регистра <i>B</i> и содержимое накапливающего сумматора сдвигаются влево и в правый крайний разряд регистра <i>B</i> помещается 1
0.0001	0.0000	0.0110	Содержимое регистра <i>Y</i> вычитается из содержимого накапливающего сумматора
0.0001	1.1010	0.0110	
0.0010	0.0000	0.0110	Содержимое регистра <i>Y</i> складывается с содержимым накапливающего сумматора, и содержимое регистра <i>B</i> и накапливающего сумматора сдвигаются влево на одну позицию. В последний разряд регистра <i>B</i> помещается 0
0.0010	0.0000		Теперь содержимое накапливающего сумматора следует сдвинуть вправо два раза, но оно равно 0, поэтому результат не изменится. Частное в регистре <i>B</i> равно 2, а остаток в накапливающем сумматоре — 0

лимое должны быть преобразованы к виду величины с положительным знаком. Значение разряда знака частного должно храниться во время деления. Оно определяется следующим правилом: если разряды знака делимого и делителя оба содержат 0 или 1, то частное будет положительным. Если только один из них содержит 1, то частное будет отрицательным. Поэтому соотношение между значением разряда знака *S* частного и значениями разрядов знака *X* и *Y* делителя и делимого определяется логикой работы четвертьсумматора или операцией ИСКЛЮЧАЮЩЕЕ ИЛИ,  $S = XY + \bar{X}\bar{Y}$ . При делении значение правильного разряда знака может считываться в триггер и затем засылаться в разряд знака регистра, содержащего частное от деления величин.

Таблица 6.3

Регистр <i>B</i>	Накапли- вающий сумматор	Регистр <i>Y</i>	Замечания
0.0000	0.1101	0.0011	Разделить 13 на 3
0.0000	0.1101	0.0110	Содержимое регистра <i>Y</i> сдвигается влево
0.0000	0.1101	0.1100	Содержимое регистра <i>Y</i> сдвигается влево. Левые крайние единицы в накапливающем сумматоре и регистре <i>Y</i> выравниваются. Элементарный шаг должен выполняться три раза
0.0000	0.0001	0.1100	Содержимое регистра <i>Y</i> вычитается из содержимого накапливающего сумматора. Результат положительный
0.0001	0.0010	0.1100	Содержимое регистра <i>B</i> и содержимое накапливающего сумматора сдвигаются влево, и в регистр <i>B</i> помещается 1
0.0001	1.0110	0.1100	Содержимое регистра <i>Y</i> вычитается из содержимого накапливающего сумматора. Результат отрицательный
0.0010	0.0100	0.1100	Содержимое регистра <i>Y</i> складывается с содержимым накапливающего сумматора. Содержимое регистра <i>B</i> и содержимое накапливающего регистра сдвигаются влево, и в регистр <i>B</i> помещается 0
0.0010	1.1000	0.1100	Содержимое регистра <i>Y</i> вычитается из содержимого накапливающего сумматора. Результат отрицательный
0.0100	0.1000	0.1100	Содержимое регистра <i>Y</i> складывается с содержимым накапливающего сумматора. Содержимое накапливающего сумматора и содержимое регистра <i>B</i> сдвигаются влево, и в соответствующий разряд регистра <i>B</i> помещается 0
0.0100	0.0001	0.1100	Содержимое накапливающего сумматора сдвигается вправо три раза. Частное равно 4, а остаток — 1.

Для деления без восстановления остатка имеется несколько методов. Один из распространенных алгоритмов включает процедуру, при которой поочередно производится сложение и вычитание делителя. Другой алгоритм использует метод, в котором делитель сравнивается с делимым при каждом пробном делении.

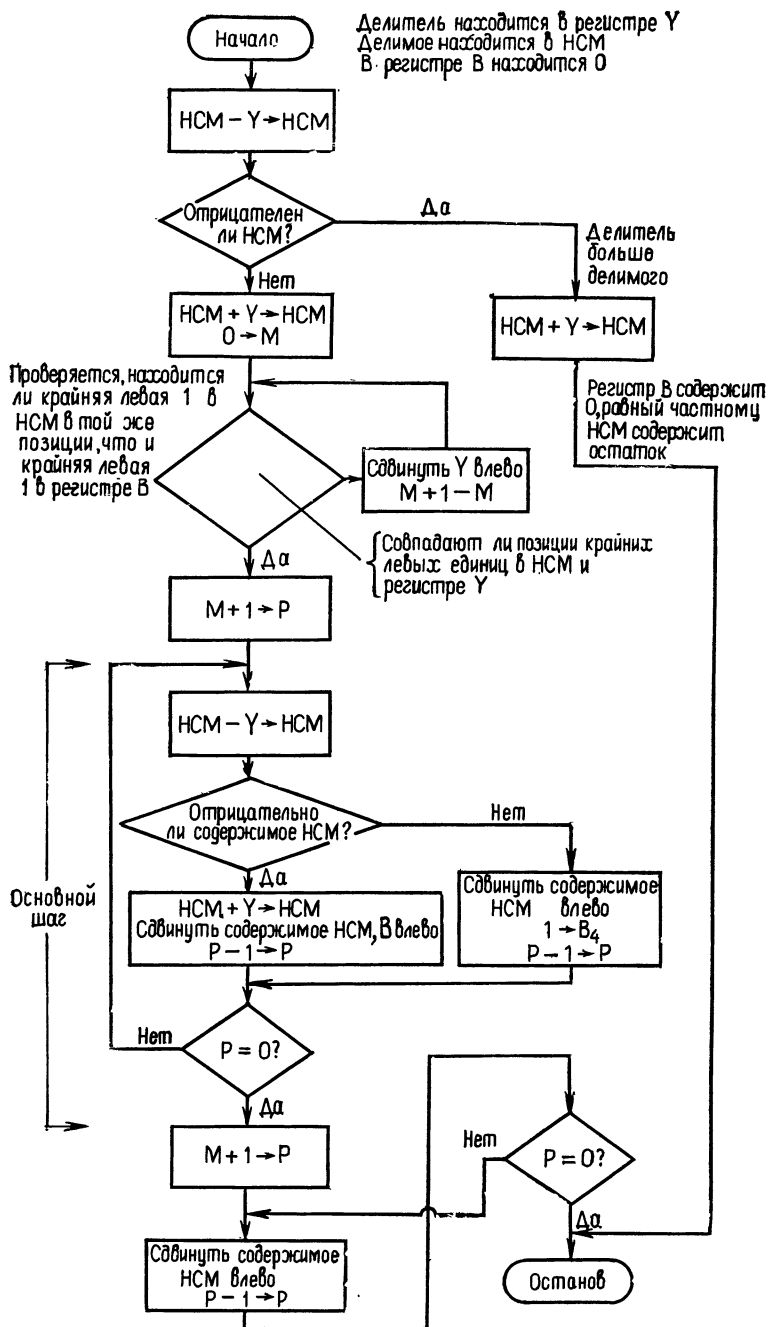


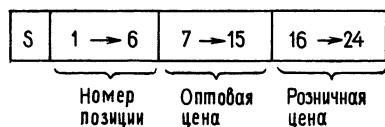
Рис. 6.19. Блок-схема алгоритма деления.

## 6.20. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Кроме арифметических операций с помощью АЛУ выполняются многие логические операции. В данном разделе будут описаны три логические операции: логическое умножение, логическое сложение и **сложение по модулю 2** (операция **ИСКЛЮЧАЮЩЕЕ ИЛИ**). Каждая из них является операцией между регистрами, выполняемой над каждым из соответствующих разрядов в двух регистрах. Результат операции хранится в одном из этих регистров.

Первая операция, логическое умножение, часто называется операцией **выделения**, или **маскирования**, или же операцией **И**. Правила логического умножения определяются соотношениями  $0 \cdot 0 = 0$ ;  $0 \cdot 1 = 0$ ;  $1 \cdot 0 = 0$  и  $1 \cdot 1 = 1$ . Предположим, что содержимое регистра накапливающего сумматора «логически умножается» на содержимое другого регистра. Допустим, что каждый регистр может содержать по 5 двоичных разрядов. Если накапливающий сумматор содержит 01101, а другой регистр — 00111, содержимым накапливающего сумматора после выполнения этой операции будет 00101.

Операция маскирования, или выделения, оказывается полезной при «упаковке» машинных слов. Для экономии памяти и совместного хранения данных, связанных друг с другом, отдельные виды информации могут храниться в одном и том же машинном слове. Например, машинное слово может содержать номер позиции, оптовую и розничную цены, которые упакованы следующим образом:



Для выделения розничной цены программист должен просто логически умножить это слово на слово, которое содержит нули во всех разрядах, начиная с разряда знака и кончая 15-м разрядом, и единицы в разрядах с 16-го по 24-й. После выполнения этой операции в слове останется только розничная цена.



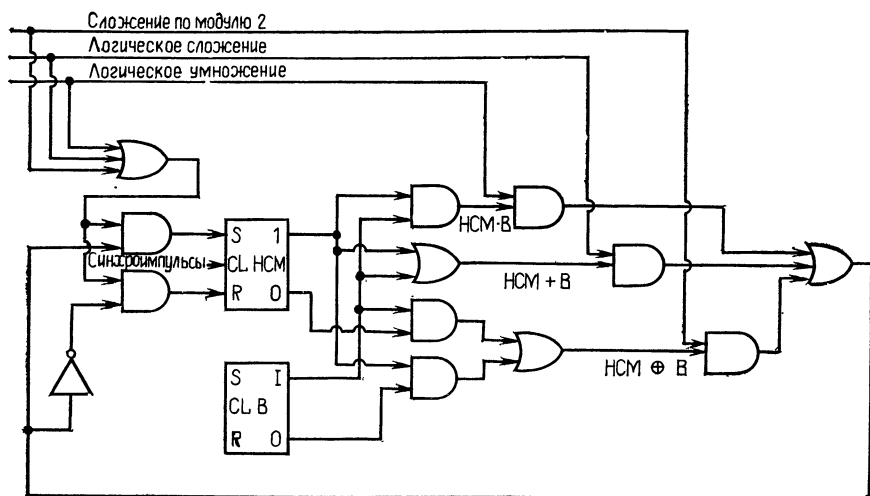


Рис. 6.20. Схема управления выполнением логических операций в триггере накапливающего сумматора.

(HCM · B) · ЛОГИЧЕСКОЕ УМНОЖЕНИЕ + (HCM + B) · ЛОГИЧЕСКОЕ СЛОЖЕНИЕ + (HCM ⊕ B) · СЛОЖЕНИЕ ПО МОДУЛЮ 2.

В большинстве ЭВМ также предусматриваются операция логического сложения и операция сложения по модулю 2. Ниже приведены правила выполнения этих операций:

**ЛОГИЧЕСКОЕ СЛОЖЕНИЕ СЛОЖЕНИЕ ПО МОДУЛЮ 2**

$0 + 0 = 0$	$0 \oplus 0 = 0$
$0 + 1 = 1$	$0 \oplus 1 = 1$
$1 + 0 = 1$	$1 \oplus 0 = 1$
$1 + 1 = 1$	$1 \oplus 1 = 0$

На рис. 6.20 показана схема, состоящая из одного триггера накапливающего сумматора, триггера регистра B и ряда вентилях, которые обеспечивают выполнение всех трех перечисленных логических операций. Такая схема может использоваться для каждого каскада регистра накапливающего сумматора.

Имеются три линии для управляющих сигналов: ЛОГИЧЕСКОЕ УМНОЖЕНИЕ, ЛОГИЧЕСКОЕ СЛОЖЕНИЕ и СЛОЖЕНИЕ ПО МОДУЛЮ 2. Если при поступлении синхронизирующего импульса подан сигнал на одну из управляющих линий или ее состояние равно 1, выполняется соответствующая операция и результат помещается в триггер накапливающего сумматора (HCM). Если ни один из управляющих сигналов не

равен 1, ни одна из указанных операций не выполняется и накапливающий сумматор сохраняет свое состояние.

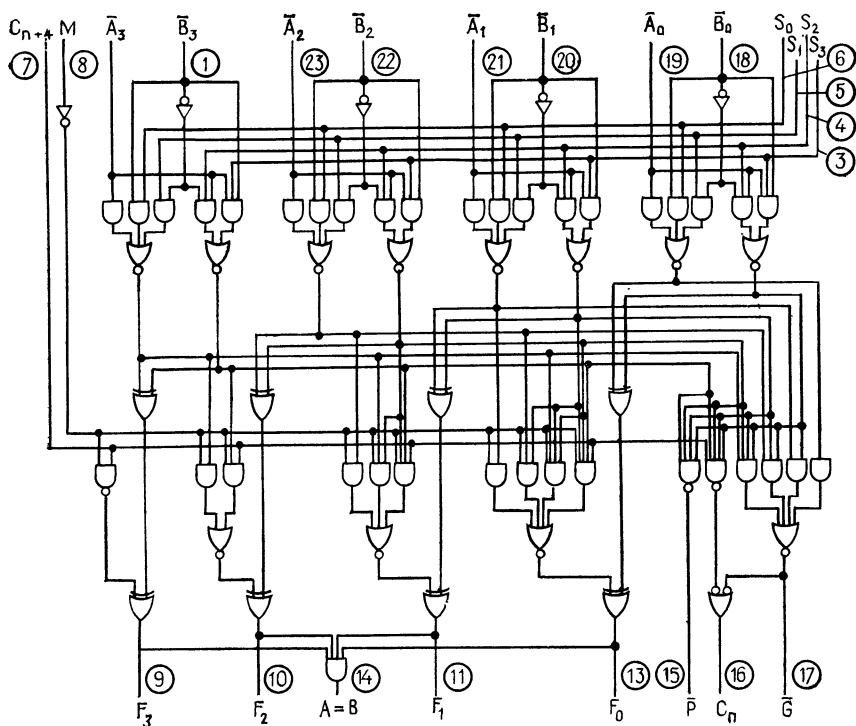
Требуемые значения результатов логических операций находятся с помощью трех групп вентилях: сначала одновременно формируются  $HCM \cdot B$ ,  $HCM + B$  и  $HCM \oplus B$ . Затем каждый из них поступает на входы соответствующих вентилях И вместе с управляющим сигналом операции. Наконец, три управляющих сигнала поступают на входы вентиля ИЛИ, выходной сигнал которого используется для пропускания соответствующего значения в триггер HCM, когда один из этих управляющих сигналов равен 1.

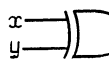
На рис. 6.20 показано, как именно, используя управляющие сигналы, можно осуществлять выбор значений нескольких различных функций для подачи их в один триггер. В число управляющих сигналов можно включить сигналы СЛОЖЕНИЕ, СДВИГ-ВПРАВО и СДВИГ-ВЛЕВО, просто добавив дополнительные вентиля.

На рис. 6.21 показан пример логической схемы, применяемой в современных ЭВМ для построения секций АЛУ. Все вентиля в этой блок-схеме расположены на одном кристалле интегральной схемы с 24 выводами. Эта ИС широко применяется (например, в сериях ЭВМ PDP-11 фирмы DEC и NOVA фирмы Data General). Для транзисторно-транзисторных логических схем (с диодами Шотки) максимальная задержка от момента поступления входного сигнала до выдачи выходного сигнала составляет 11 нс. (Для логических схем с эмиттерными связями максимальная задержка равна 7 нс.)

Такая ИС называется **четырёхразрядным арифметическим устройством** и может выполнять операции сложения, вычитания И, ИЛИ и т. д. над содержимым двух секций четырёхразрядных регистров. Два кристалла могут использоваться в логической схеме восьмиразрядного накапливающего сумматора, из четырех кристаллов можно построить 16-разрядный накапливающий сумматор.

Функция, выполняемая этой ИС, определяется значениями на входе режима  $M$  и четырех входах выбора функций  $S_0$ ,  $S_1$ ,  $S_2$  и  $S_3$ . Когда на входе режима низкий уровень (0), микросхема 74S181 выполняет такие арифметические операции, как СЛОЖЕНИЕ и ВЫЧИТАНИЕ. Когда на входе режима высокий уровень (1), АЛУ выполняет поразрядные логические операции над сигналами, поступающими на входы  $A$ ,  $B$ . (Отметим, что на рис. 6.21 при  $M=1$  вентиля, образующие переносы, заперты.) Например, если  $M=0$ ,  $S_1$  и  $S_2$  также равны 0, а  $S_0$  и  $S_3$  равны 1, то микросхема 74S181 выполняет арифметическое сложение. Если  $M=1$ ,  $S_0$  и  $S_3$  равны 1,  $S_1$  и  $S_2$  равны 0, то эта микросхема выполняет операцию



Входы выбора режима *				Активные входы и выходы на низком уровне		
$S_3$	$S_2$	$S_1$	$S_0$	Логика ( $M=N$ )	Арифметика ( $M=L$ ) ( $C_n=L$ )	
L	L	L	L	$\bar{A}$	$A - 1$	Сумматор по модулю 2 (ИСКЛЮЧАЮЩЕЕ ИЛИ) $z = x \oplus y$
L	L	L	H	$\bar{A} \bar{B}$	$\bar{A} \bar{B} - 1$	
L	L	H	L	$\bar{A} + \bar{B}$	$\bar{A} \bar{B} - 1$	
L	L	H	H	Логическая $\bar{A} + \bar{B}$	-1	
L	L	H	H	$\bar{B}$	$A \neq (A + \bar{B})$	≠ знак арифметического сложения
L	H	L	L	$\bar{B}$	$\bar{A} \bar{B} \neq (A + \bar{B})$	
L	H	L	H	$\bar{A} \oplus \bar{B}$	$A - B - 1$	
L	H	H	L	$\bar{A} + \bar{B}$	$A + \bar{B}$	
L	H	H	H	$\bar{A} \bar{B}$	$A \neq (A + B)$	
H	L	L	L	$A \oplus B$	$A \neq B$	
H	L	L	H	$A \oplus B$	$\bar{A} \bar{B} \neq (A + B)$	
H	L	H	L	$\bar{B}$	$A + \bar{B}$	
H	L	H	H	$A + \bar{B}$	$A + B$	
H	H	L	L	Логический 0	$A \neq A'$	
H	H	L	H	$\bar{A} \bar{B}$	$\bar{A} \bar{B} \neq A$	
H	H	H	L	$\bar{A} \bar{B}$	$\bar{A} \bar{B} \neq A$	
H	H	H	H	$A$	$A$	

\* L = 0; H = 1

Рис. 6.21. Четырехразрядное АЛУ.

ИСКЛЮЧАЮЩЕЕ ИЛИ (сложение по модулю 2) над сигналами, поступающими на входы  $A$  и  $B$  (она формирует  $A_0 \oplus B_0$ ,  $A_1 \oplus B_1$ ,  $A_2 \oplus B_2$  и  $A_3 \oplus B_3$ ).

В таблице, приведенной на рис. 6.21, дается более подробное описание функционирования этой ИС.

## 6.21. СИСТЕМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ТОЧКОЙ

В предыдущих разделах были описаны системы представления чисел, в которых положительные и отрицательные числа хранились в двоичных словах. В этих системах двоичная точка «фиксируется» в том смысле, что она размещается в конце каждого слова, и поэтому каждое представленное значение является целым числом. Когда на ЭВМ обрабатываются двоичные числа в этом формате, соответствующие операции называются **арифметическими операциями с фиксированной точкой**.

В научных расчетах часто приходится производить вычисления с очень большими или очень малыми числами. Поэтому ученые пользуются удобной системой представления числа с помощью **мантиссы** и **порядка**. Например, 4 900 000 можно записать как  $0.49 \times 10^7$ , где 0.49 — значение мантиссы, 7 — порядок. Число 0.00023 может быть записано в виде  $0.23 \times 10^{-3}$ . Это обозначение основано на соотношении  $y = a \times r^p$ , где  $y$  — представляемое число,  $a$  — мантисса,  $r$  — основание системы счисления ( $r = 10$  для десятичной системы и  $r = 2$  для двоичной системы),  $p$  — порядок (степень, в которую возводится основание).

Эта система представления чисел может быть использована и при вычислениях. Для умножения  $a \times 10^m$  на  $b \times 10^n$  нужно записать произведение в виде  $(a \times b) \times 10^{m+n}$ . Чтобы разделить  $a \times 10^m$  на  $b \times 10^n$ , частное надо записать как  $a/b \times 10^{m-n}$ . Для сложения чисел  $a \times 10^m$  и  $b \times 10^n$  необходимо сначала добиться равенства  $m$  и  $n$ . Если  $m = n$ , то  $a \times 10^m + b \times 10^n = (a + b) \times 10^m$ . Процесс, приводящий к равенству  $m$  и  $n$ , называется выравниванием порядков (чисел).

При выравнивании порядков чисел может потребоваться значительный объем вычислений и в случае, когда числа принимают значения в весьма широком диапазоне, могут возникнуть осложнения в отношении сохранения точности. При использовании ЭВМ эти трудности устраняются с помощью двух методов, благодаря которым ЭВМ (а не программист) прослеживает положение точки в позиционном представлении числа, автоматически осуществляя выравнивание порядков чисел. Первый метод состоит в использовании специальных подпрограмм для работы с плавающей точкой, в которых производится

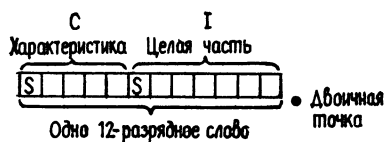


Рис. 6.22. 12-разрядное слово с плавающей точкой.

автоматическое выравнивание порядков чисел, обрабатываемых в процессе вычислений. При этом обеспечивается сохранение точности результатов и прослеживается изменение масштабных множителей. Эти подпрограммы применяются в малых ЭВМ, выполняющих только операции с фиксированной точкой. При использовании второго метода в оборудование ЭВМ встраивается аппаратура для реализации так называемых **операций с плавающей точкой**. Тогда для автоматического выравнивания и прослеживания изменений порядков в процессе вычислений используются логические схемы ЭВМ. При этом применяется система представления чисел, называемая **системой с плавающей точкой**.

Для числа с плавающей точкой в ЭВМ используется экспоненциальная система представления, и в процессе вычисления ЭВМ прослеживает изменения как порядков, так и мантисс. Машинное слово, описывающее число в системе с плавающей точкой, можно разбить на три части: 1) разряд знака, указывающий является ли число положительным или отрицательным, 2) порядок представляемого числа, 3) мантисса числа.

В качестве примера рассмотрим 12-разрядную ЭВМ, использующую слово с плавающей точкой (рис. 6.22). Обычно часть слова, соответствующую порядку, называют **характеристикой**, а часть слова, соответствующую мантиссе, — **целой частью**. Мы будем придерживаться этой терминологии.

Целая часть числа с плавающей точкой представляет значение в виде его величины со знаком (а не дополнения до двух, хотя и оно используется). Характеристика также записывается в виде величины со знаком. Значение рассматриваемого числа равно  $I \times 2^C$ , где  $I$  — значение целой части,  $C$  — значение характеристики.

На рис. 6.23 приведено несколько значений чисел с плавающей точкой как в двоичной записи, так и в десятичной. Поскольку характеристика имеет пять битов и задана в виде величины со знаком,  $C$  в выражении  $I \times 2^C$  может иметь значения в диапазоне от  $-15$  до  $+15$ . Значение  $I$  представлено в виде двоичной величины со знаком с семью битами, и поэтому  $I$  может принимать значения в пределах от  $-63$  до  $+63$ . Наибольшее число, представляемое в этой системе, долж-

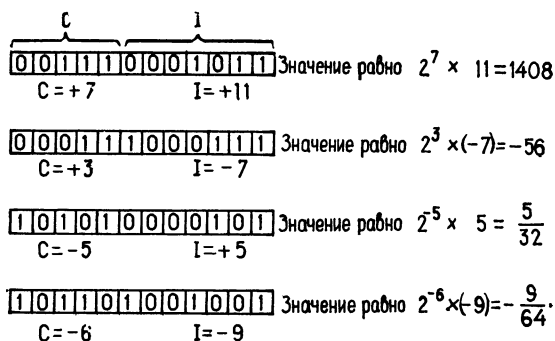


Рис. 6.23. Значения чисел с плавающей точкой в 12-разрядных словах.

но соответствовать максимальному  $I$  и быть равным  $63 \times 2^{15}$ . Наименьшим числом, представляемым в ней, будет  $-63 \times 2^{15}$ .

Этот пример иллюстрирует применение системы представления чисел с плавающей точкой для хранения в двоичном слове «действительных» чисел, принимающих значения из достаточно широкого диапазона.

Другой метод, получивший распространение на практике, состоит в представлении мантиисы слова в виде дробной величины, а не целой. Это согласуется с обычным использованием мантиис в научных расчетах, так как часто «нормальной» формой числа считают  $0.93 \times 10^4$  (а не  $93 \times 10^2$ ). В этом случае мантииса в десятичном представлении принимает значения в диапазоне от 0.1 до 0.999... Аналогично двоичная мантииса имеет значения в пределах от 0,5 (в десятичной форме) и почти до 1. В большинстве ЭВМ мантиисы поддерживаются в нормальной форме; при этом непрерывно корректируются слова, с тем чтобы значащий бит (1) всегда занимал левую крайнюю позицию в мантиисе (следующую после разряда знака).

Когда мантииса представлена в виде дробной величины, соответствующая часть слова называется **дробью**. При этом для 12-разрядных слов в рассмотренном примере числа с плавающей точкой можно записать просто, предполагая, что двоичная точка расположена левее величины (а не правее, как в случае представления целых чисел). В этой системе любое число представляется в виде  $F \times 2^C$ , где  $F$  — двоичная дробь,  $C$  — характеристика.

Для уже рассмотренного 12-разрядного слова дроби должны иметь значения в пределах от  $(1-2^{-6})$  до  $-(1-2^{-6})$ , т. е. от 0.111111 до 1.111111. Таким образом, можно представить

числа в диапазоне от  $(1-2^{-6}) \times 2^{15}$  до  $-(1-2^{-6}) \times 2^{15}$ , т. е. от 32 000 до -32 000. Наименьшим значением дробной части теперь будет 0.1000000, равное  $2^{-1}$ , а наименьшей характеристикой —  $2^{-15}$ , поэтому наименьшее представимое положительное число равно  $2^{-1} \times 2^{-15}$  или  $2^{-16}$ . Большинство ЭВМ используют такую систему, которая предусматривает представление мантиссы в виде дроби, хотя ЭВМ фирмы Bigrroughs и National Cash Register применяют вышеописанную целочисленную систему.

В ЭВМ UNIVAC 1108 числа с плавающей точкой представляются с одинарной точностью в следующем формате:



Для положительных чисел характеристика  $C$  рассматривается как двоичное целое число, разряд знака которого содержит 0, а дробная часть — как двоичная дробь со значениями из диапазона  $0.5 \leq F < 1$ . Значение представленного числа равно  $2^{C-128} \times F$ . Эта система называется **системой со смещением**, поскольку значение характеристики просто равно целочисленному значению, содержащемуся в рассматриваемой части слова, минус смещение, равное 128. Поэтому порядок может изменяться от -128 до +127, так как целая часть характеристики имеет длину в 8 бит.

Например, двоичное слово

0	1	0	0	0	0	0	1	1	1	0	0	...	0
характеристика								дробь					

имеет значение  $2^{129-128} \times \frac{3}{4} = 2 \times \frac{3}{4} = 1.5$ . Представление отрицательных чисел может быть получено путем формирования записи положительного числа с такой же абсолютной величиной и затем его дополнения до 1 (принимая все 36 бит за одно двоичное число).

В ЭВМ, использующих 16-разрядные слова (включая некоторые ЭВМ, выпускаемые фирмами DEC, Hewlett-Packard, Data General, IBM), слова с плавающей точкой представляются в виде двух смежных слов и поэтому каждое слово состоит из 32 бит. На рис. 6.24 показан формат слов с плавающей точкой для некоторых из этих ЭВМ. Дробная часть этих слов состоит из 24 бит, из которых 23 бит выделены для записи дроби, один бит — для знака. Порядок, или характеристика, состоит из 8 бит. (В ЭВМ фирмы Hewlett-Packard дробная и

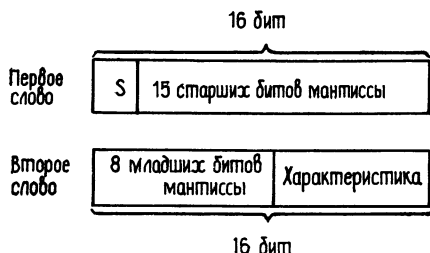


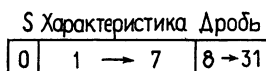
Рис. 6.24 Представление чисел с плавающей точкой с помощью двух слов.

характеристическая части представлены в форме дополнения до 2, как это предусмотрено для языка Фортран.) В каждой из этих ЭВМ могут быть представлены величины до  $2^{127}$  (или приблизительно  $10^{38}$ ) и дроби, наименьшая из которых равна  $2^{-138}$  (приблизительно  $10^{-38}$ ).

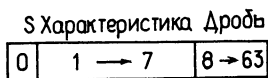
В качестве другого примера ЭВМ с внутренними схемами для выполнения операций с плавающей точкой и использования единого представления машинного слова для чисел с плавающей точкой можно привести серию ЭВМ IBM 360/370.

По терминологии фирмы IBM часть слова, представляющая порядок, называется **характеристикой**, а часть слова с мантиссой — **дробью**. Для ЭВМ серии IBM 360/370 слова, представляющие данные с плавающей точкой, могут иметь длину в 32 или 64 бит. Ниже приведены их основные форматы:

короткое число или число с плавающей точкой в одном слове



длинное число или число с плавающей точкой в двойном слове



В обоих случаях разряд знака  $S$  расположен в левой крайней позиции и определяет знак числа. Часть слова, представляющая характеристику, включает биты 1—7 и является двоичным целым числом, которое обозначим через  $C$ . Оно изменяется в диапазоне от 0 до 127. Значение масштабного множителя находится путем вычитания из  $C$  числа 64 и возведения 16 в эту степень. Таким образом, значение 64 в разрядах 1—7 определяет масштабный множитель  $16^{C-64} = 16^{64-64} = 16^0$ ; 93 (десятичное) в разрядах 1—7 дает масштабный множитель



$16^{C-64} = 16^{93-64}$ , который равен  $16^{29}$ ; число 24 в разрядах 1—7 соответствует дроби  $16^{-40}$ .

Величина числа, представленного в данном слове с плавающей точкой, равна произведению этого масштабного множителя на дробь, содержащуюся в разрядах 8—31 для короткого числа или в разрядах 8—63 для длинного числа. В каждом из этих случаев подразумевается, что разделительная точка расположена левее 8-го разряда. Таким образом, если разряды 8—31 содержат 10000...000, то десятичное значение дроби равно  $1/2$  или .1000...000 в двоичном представлении. Аналогично если разряды 8—31 содержат 11000...000, значение дробной части равно  $3/4$  (десятичное) или .11000...000 (двоичное).

Представленное число имеет величину, равную произведению значения дроби на значение, определенное характеристикой. Рассмотрим короткое число:

	Знак	Характеристика	Дробь
Число с плавающей точкой:	0	1 0 0 0 0 0 1	1 1 1 0 0 . . . 0
Позиция бита:	0	1 2 3 4 5 6 7	8 9 10 11 12 . . . 31

Поскольку в разряде знака стоит 0, это число положительное. Характеристика имеет двоичное значение 1000001, которое в десятичном представлении будет 65; таким образом, масштабный множитель равен  $16^1$ . Дробная часть имеет двоичное значение .111 или  $7/8$  в десятичной записи, поэтому представленное число равно  $7/8 \times 16$  или 14 в десятичной форме.

Рассмотрим следующее число:

	Знак	Характеристика	Дробь
Число с плавающей точкой:	1	1 0 0 0 0 0 1	1 1 1 0 0 . . . 0
Позиция бита:	0	1 2 3 4 5 6 7	8 9 10 11 12 . . . 31

Это число имеет значение  $-14$ , так как все его разряды, кроме разряда знака, такие же, как в предыдущем случае. (В рассматриваемой системе числа представляются в виде величин со знаком.)

Другими примерами могут служить:

Знак	Характеристика	Дробь	
0	1 0 0 0 0 1 1	1 1 0 . . . 0	$16^3 \times 3/4 = 3072$
0	0 1 1 1 1 1 1	1 1 0 . . . 0	$16^{-1} \times 3/4 = 3/64$

## 6.22. ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ НАД ЧИСЛАМИ С ПЛАВАЮЩЕЙ ТОЧКОЙ

Очевидно, что для автоматического выполнения операций над числами с плавающей точкой ЭВМ необходимы дополни-

тельные схемы. Некоторые ЭВМ выпускаются с устройствами для выполнения команд с фиксированной точкой. (Такие ЭВМ, как PDP-11/45 фирмы DEC, и другие можно дополнить схемами для выполнения операций над числами с плавающей точкой.)

Чтобы ЭВМ могла обрабатывать числа с плавающей точкой, она должна обладать способностью широкого использования операций сдвига и сравнения. Запишем правила умножения и деления чисел с плавающей точкой:

$$(a \times r^p) \times (b \times r^q) = ab \times r^{p+q}, \quad (a \times r^p) : (b \times r^q) = a/b \times r^{p-q}.$$

ЭВМ должна иметь возможность складывать и вычитать порядки чисел с плавающей точкой, а также выполнять операции умножения и деления над мантиссами чисел. Точность результатов обычно поддерживается посредством сдвига чисел до тех пор, пока старшие разряды не будут находиться в крайних левых позициях слова. После каждого сдвига порядок числа должен быть изменен. Если в ЭВМ производится сдвиг мантиссы влево, то при каждом сдвиге порядок должен быть уменьшен на 1. Например, рассмотрим машинное слово двоично-десятичной ЭВМ

0	10	0064
знак	порядок	мантисса

Для достижения необходимой точности в ЭВМ производится сдвиг мантиссы влево до тех пор, пока 6 не будет находиться в старшей позиции. Поскольку для этого необходимо два сдвига, порядок должен уменьшиться на 2; результирующее слово будет иметь вид 0.08 6400. Если все используемые числа масштабируются таким образом, то можно обеспечить максимальную точность в процессе вычислений.

При сложении и вычитании значения порядков должны быть согласованы. Например, чтобы сложить  $0.24 \times 10^5$  с  $0.25 \times 10^6$ , необходимо так масштабировать слагаемые, чтобы порядки были равны. Таким образом,

$$(0.024 \times 10^6) + (0.25 \times 10^6) = 0.274 \times 10^6.$$

ЭВМ также должна выполнять эту процедуру. Масштабирование чисел производится описанным способом, так что старший разряд той части каждого машинного слова, которая соответствует мантиссе, содержит старшую цифру хранимого числа. В этом случае выбирается наибольший из двух порядков, мантисса другого числа сдвигается и его порядок изменяется так, чтобы согласовать порядки обоих чисел. После этого числа складываются или вычитаются в соответствии с правилами

$$(a \times r^p) + (b \times r^p) = (a + b) \times r^p, \quad (a \times r^p) - (b \times r^p) = (a - b) \times r^p.$$

# 7

## ЭЛЕМЕНТ ПАМЯТИ

*Т. Барти*

### 7.1. ВВЕДЕНИЕ

Память вычислительной машины не представляет единого устройства, расположенного в одном месте. Используемая в ЭВМ память по своим характеристикам неоднородна и образует некоторую иерархию уровней запоминающих устройств. Например, устройствами памяти являются **операционные регистры**, представляющие собой триггерные регистры, которые используются в арифметических и управляющих блоках ЭВМ. Арифметические операции, включая сложение, умножение, сдвиги и т. д., выполняются в этих регистрах.

Следующим типом запоминающего устройства является **быстродействующая память, внутренняя память, или основная память**. Эта часть памяти ЭВМ состоит из набора запоминающих регистров, каждый из которых идентифицируется адресом, обеспечивающим возможность с помощью управляющего устройства записать информацию в данный регистр или считать из него.

Желательно, чтобы быстродействие этой части памяти ЭВМ было как можно выше, так как большая часть передач данных в устройство, обрабатывающее информацию, и из него происходит через основную память. Поэтому в качестве основной памяти обычно используют запоминающие устройства с очень малым временем доступа; к сожалению, имеющиеся в настоящее время устройства, достаточно быстродействующие для выполнения этой функции, не всегда обладают требуемым объемом памяти. Поэтому к большинству ЭВМ добавляется дополнительная память, которую называют **вспомогательной, или вторичной памятью**. Эта часть памяти ЭВМ характеризуется низкой стоимостью хранения единицы информации, но ее быстродействие значительно меньше, чем операционных регистров или основной памяти. Иногда эту память называют **внешней па-**

**мятью**, так как ее функция состоит в хранении данных, которые не помещаются во внутренней памяти.

Наконец, на низшем уровне иерархии запоминающих устройств находятся устройства, используемые для ввода информации в ЭВМ из «внешнего мира» и хранения результатов, передаваемых от ЭВМ к пользователю. Запоминающей средой в данном случае обычно являются такие входные носители, как перфоленты и перфокарты, а выходная информация — это в основном отпечатанные символы. И здесь стоимость хранения бита информации низкая, а быстродействие устройств ввода с перфолент и перфокарт, печатающих устройств и т. д. примерно в 1000 раз меньше, чем у операционных регистров. В данной главе рассматривается **внутренняя память** машины. Это такие запоминающие устройства, которые являются неотъемлемой частью ЭВМ и непосредственно управляются ею.

Каждая из перечисленных частей памяти имеет свои характерные особенности. Например, операционные регистры представляют сверхбыстродействующую память. Скорость выполнения операций в этих регистрах обычно в несколько раз превышает быстродействие основной памяти. Основная память должна обладать высоким быстродействием и иметь возможность хранить большие объемы данных ( $10^4$  —  $10^9$  бит). Быстродействие и стоимость являются противоречивыми характеристиками памяти, поэтому при ее выборе необходим определенный компромисс между стоимостью и быстродействием. Компромисс такого же рода часто требуется и в случае вспомогательной памяти. В больших машинах может оказаться необходимым хранение во вспомогательной памяти от  $10^8$  до  $10^{12}$  бит; в таких случаях слишком дорого использовать такие же устройства, как и в основной памяти.

Говоря о быстродействии, важно учитывать то обстоятельство, что, прежде чем прочитать слово, необходимо установить его местонахождение. Время, требуемое для установления местонахождения и считывания слова, называется **временем доступа**. По процедуре поиска информации запоминающие устройства можно разделить на два класса: с произвольным доступом и с последовательным доступом. Запоминающее устройство с **произвольным доступом** (ЗУПД) — это такое устройство, в котором любая ячейка памяти может быть выбрана произвольно, доступ к хранимой информации прямой и время доступа для всех ячеек памяти примерно одинаковое. Примерами запоминающих устройств с произвольным доступом являются триггерный регистр, а также память на магнитных сердечниках и память на интегральных схемах (ИС), которые будут рассмотрены ниже. Устройство с **последовательным доступом** — это такое устройство, в котором для получения доступа к определенной

ячейке памяти следует последовательно просмотреть все предшествующие ей ячейки памяти, так что время доступа зависит от расположения адресуемой ячейки памяти. Например, если необходимо прочитать слово, хранящееся на магнитной ленте, а участок ленты, на котором записано слово, находится в центре бобины ленты, то, прежде чем прочесть слово, придется последовательно просмотреть половину ленты.

Можно разделить запоминающие устройства и по другому принципу, а именно в зависимости от того, статические они или динамические. **Статическое** запоминающее устройство — это такое устройство, в котором информация не меняет своего положения в носителе информации; примерами статических запоминающих устройств являются триггерные регистры, регистры на магнитных сердечниках, а также перфокарты и перфоленты. **Динамические** запоминающие устройства — это такие устройства, в которых записанная информация непрерывно перемещается в запоминающей среде. Примером динамических запоминающих устройств могут служить регистры с циркулирующей в них информацией, которые используют линии задержки на приборах с зарядовой связью.

Основное внимание в этой главе будет сосредоточено на четырех устройствах, наиболее часто используемых для хранения информации внутренней памяти ЭВМ. К ним относятся: 1) запоминающие устройства на интегральных схемах, которые характеризуются высокой скоростью и небольшой стоимостью; 2) запоминающие устройства на магнитных сердечниках с произвольным доступом, которые используются преимущественно в основной памяти благодаря их высокому быстродействию и небольшой стоимости бита; 3) запоминающие устройства на магнитных барабанах и дисках с прямым доступом, которые используются во вспомогательной памяти, и 4) запоминающие устройства на магнитных лентах, применяемые только в качестве вспомогательной, или внешней, памяти; но способные хранить большие объемы информации при низкой стоимости одного бита. В разделах, посвященных барабанным, дисковым и магнитным ленточным устройствам, будут описаны способы записи цифровой информации на магнитной поверхности.

## **7.2. ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО С ПРОИЗВОЛЬНЫМ ДОСТУПОМ**

Основная память ЭВМ организована наиболее подходящим образом. На рис. 7.1 видно, что быстродействующая основная память ЭВМ организована в виде слов фиксированной длины. Как показано на рисунке, эта память делится на  $N$  слов, где  $N$  — обычно некоторая степень 2, а каждому слову присваи-

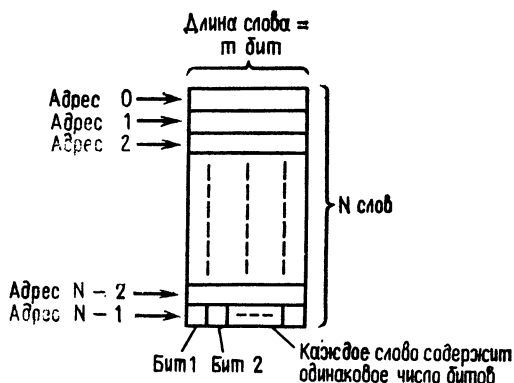


Рис. 7.1. Расположение слов в быстродействующей памяти.

вается **адрес** (или **ячейка**) в памяти. Каждое слово содержит одинаковое число битов, называемое **длиной слова**; считывая, например, слово из ячейки 72, получаем из памяти слово определенной длины.

Адреса, или адресные числа, в памяти — последовательно возрастающие числа, начиная от 0 и кончая наибольшим адресом. Так, по адресу 0 мы найдем одно слово, по адресу 1 — другое слово, по адресу 2 — третье и т. д. до самого последнего слова с наибольшим адресом. Вообще ЭВМ может считывать слово из любой ячейки памяти или записывать слово в любую ячейку памяти. Для памяти с 8-разрядным словом, если мы запишем слово 01001011 по адресу 17, а затем позже считаем из ячейки с этим адресом, то получим слово 01001011. Если же мы еще раз считаем слово по этому адресу позднее (но не записывая туда другого слова), то опять считаем слово 01001011. Говорят, что это память с **чтением без разрушения**, т. е. процесс чтения не разрушает и не меняет хранимое в памяти слово.

Важно понять разницу между **содержимым** памяти по какому-то адресу и самим адресом. Память похожа на большой шкаф, в котором столько ящиков, сколько адресов в памяти. В каждом «ящике» содержится слово, а адрес каждого слова написан на ящике. Когда мы записываем или запоминаем слово с адресом 17, то как бы помещаем его в ящик с меткой 17. Затем, считывая по адресу 17, мы как бы заглядываем в ящик, чтобы увидеть его содержимое. Мы не уничтожаем слово, когда читаем его, но только тогда меняем содержимое памяти по какому-либо адресу, когда запоминаем или записываем новое слово.

Быстродействующая основная память очень похожа на «черный ящик» с набором ячеек (или адресов), в которых можно

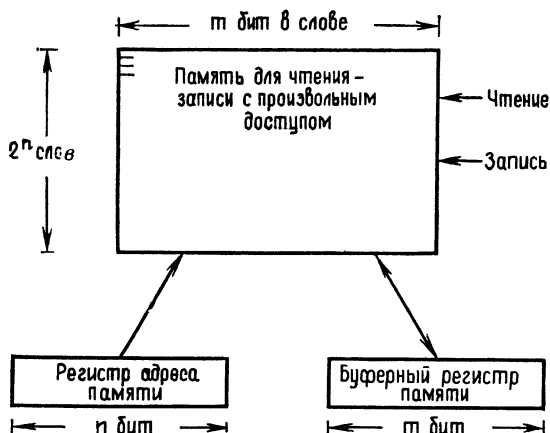


Рис. 7.2. Память для чтения-записи с произвольным доступом.

запомнить или из которых можно считать данные. Каждый адрес (или ячейка) содержит фиксированное число двоичных разрядов, которое называется **длиной слова** памяти. Память из 4096 ячеек с разными адресами, каждая из которых хранит 16 бит, называют **памятью емкостью 4096 16-разрядных слов**, или, как принято в вычислительной технике, **памятью в 4К слов по 16 бит**. (Число слов в памяти обычно равно  $2^n$ , где  $n$  — некоторое число, поэтому, если емкость памяти составляет, например,  $2^{14} = 16\,384$  слова, о такой памяти обычно говорят, как о памяти емкостью 16К, так как всегда ясно, что фактически в памяти содержится полностью  $2^n$  слов. Так память емкостью  $2^{15}$  16-разрядных слов называют **памятью емкостью 32К слов по 16 бит**.) Можно считывать из памяти (т. е. извлекать данные) или записывать в память (т. е. вводить данные в память). Память, которая допускает как считывание, так и запись, называется **памятью для чтения-записи**. В некоторых типах памяти программы или данные хранятся в неизменяемой форме; такая память называется **постоянной памятью**.

На рис. 7.2 изображена блок-схема памяти для чтения — записи; ЭВМ помещает адрес ячейки, в которую должны быть записаны данные, в **регистр адреса памяти**. Этот регистр состоит из  $n$  двоичных устройств (обычно триггеров), где  $2^n$  — количество слов, которое может храниться в памяти. Данные, подлежащие записи в память, помещаются в **буферный регистр памяти**, который содержит столько двоичных запоминающих устройств, сколько разрядов в каждом слове памяти. Для осуществления записи память получает команду записать в виде сигнала 1 на линию ЗАПИСЬ. Затем память запоминает содержи-

мое буферного регистра памяти в ячейке, адрес которой указан в регистре адреса памяти.

Чтобы считать слово, адрес ячейки помещают в регистр адреса памяти. Затем на линию ЧТЕНИЕ подают сигнал 1, и память передает содержимое этой ячейки в буферный регистр памяти.

Легко заметить, что ЭВМ взаимодействует с памятью с помощью регистра адреса памяти, буферного регистра памяти и входов ЧТЕНИЕ и ЗАПИСЬ. Память обычно конструктивно выполняется в виде отдельных модулей или блоков. У многих изготовителей можно приобрести модуль памяти необходимого размера; например, можно приобрести уже готовый модуль памяти емкостью 8К слов по 16 бит. Купив ЭВМ с определенной емкостью основной памяти, можно наращивать память, вставляя дополнительные модули.

Память называют **памятью с произвольным доступом**, если она позволяет «немедленно» считывать из ячейки или записывать в нее, т. е. если нет задержки в доступе к одной из ячеек по сравнению с другой. В ЭВМ почти всегда в качестве быстродействующей основной памяти используется память для чтения-записи с произвольным доступом, а для хранения вспомогательных данных применяется внешняя медленнодействующая память.

### 7.3. ОРГАНИЗАЦИЯ ПАМЯТИ С ЛИНЕЙНОЙ ВЫБОРКОЙ

Чаще всего используемыми видами памяти с произвольным доступом являются память на магнитных сердечниках и память на интегральных схемах. Как будет показано в дальнейшем, запоминающих устройств, выпускаемых промышленностью.

Для пояснения основных принципов будет рассмотрена идеализированная модель памяти на ИС и некоторые особенности запоминающих устройств, выпускаемых промышленностью.

В любой памяти обязательно должна быть базовая ячейка памяти. На рис. 7.3 показана базовая ячейка памяти, состоящая из RS-триггера и связанной с ним схемы управления. Однако, чтобы использовать эту ячейку в памяти, должны быть выбраны определенный метод выборки таких ячеек, адрес которых находится в регистре адреса памяти, а также метод управления записью и считыванием.

На рис. 7.4 показана организация базовой памяти для памяти на ИС с **линейной выборкой**. Это четырехадресная память с трехразрядным словом. Регистр адреса памяти (РАП) определяет ячейки памяти (триггеры), из которых нужно считать или в которые требуется записать, с помощью **дешифратора**,



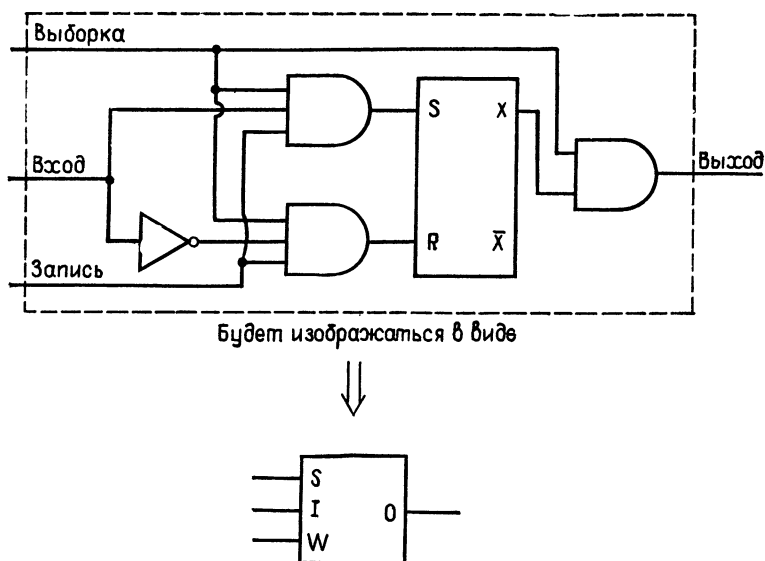


Рис. 7.3. Базовая ячейка памяти.

выбирающего три триггера для каждого адреса, находящегося в регистре адреса памяти.

На рис. 7.5, а изображена структурная схема дешифратора. Сигналы на входы этой схемы поступают с триггеров, выходы которых должны быть декодированы. Как показано на рис. 7.5, а, при двух входных битах на выходе дешифратора необходимо иметь четыре выходные линии по одной для каждого состояния (значения) которое может принимать входной регистр. Так, например, если РАП содержит 0 в обоих триггерах, то 1 появится на верхней линии дешифратора, а 0 — на трех остальных линиях. Аналогично если обе ячейки памяти содержат 1, то 1 окажется на нижней выходной линии дешифратора, а 0 — на остальных выходных линиях. Подобные рассуждения покажут, что для каждого возможного входного состояния 1 будет только на одной выходной линии, а на остальных линиях будет 0.

На рис. 7.5, б показан дешифратор на три входа. Дешифратор имеет 8 выходных линий. В общем случае для  $n$  входных битов дешифратор будет иметь  $2^n$  выходных линий.

Принцип действия дешифратора на рис. 7.5, б такой же, как и дешифратора, изображенного на рис. 7.5, а. Для каждого входного состояния дешифратор выберет одну выходную линию, на которой будет 1; на остальных линиях будет 0.

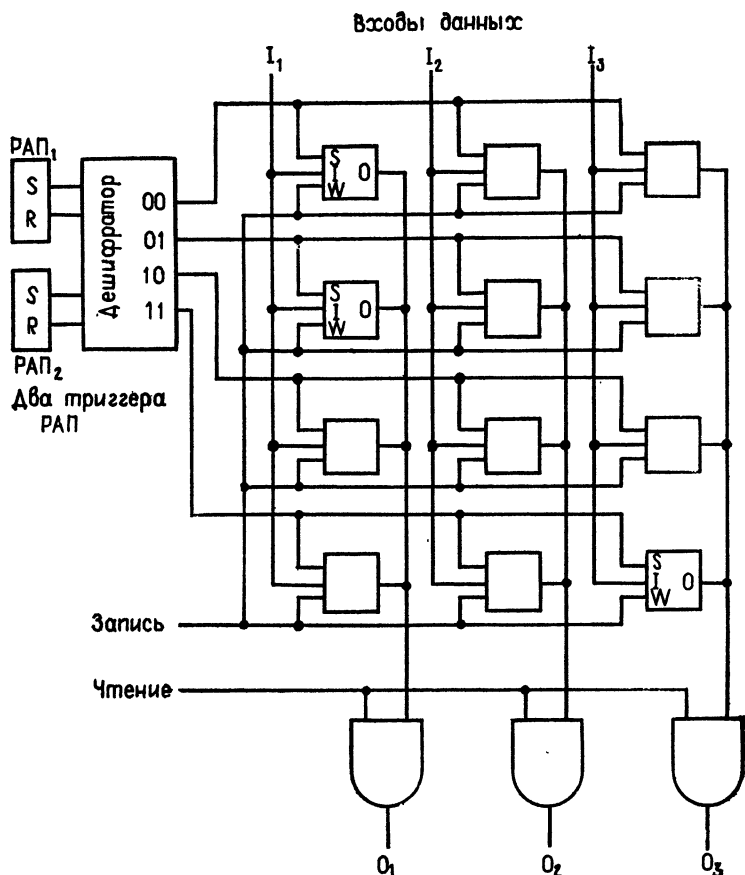


Рис. 7.4. Память на ИС с линейной выборкой.

Возвращаясь к рис. 7.4, мы теперь видим, что для каждого значения, которое может поступить в РАП, будет выбрана выходная линия дешифратора со значением 1. На остальных выходных линиях дешифратора будет 0, и не будут выбираться вентили И, находящиеся на входах и выходах триггеров в этих горизонтальных рядах (строках). (См. также рис. 7.3.)

Память на рис. 7.4 организована следующим образом: она содержит четыре слова, каждая строка из трех ячеек памяти составляет слово. В любой заданный момент времени РАП выбирает ячейку в памяти. Если на линию ЧТЕНИЕ подается 1, то содержимое трех ячеек выбранного слова считывается на линии  $O_1$ ,  $O_2$  и  $O_3$ . Если на линию ЗАПИСЬ подается 1, то в память будут считаны значения с линий  $I_1$ ,  $I_2$  и  $I_3$ .

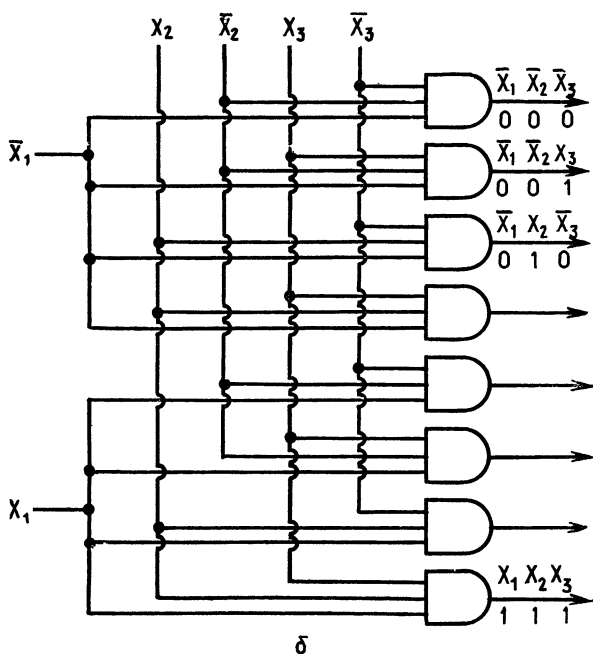
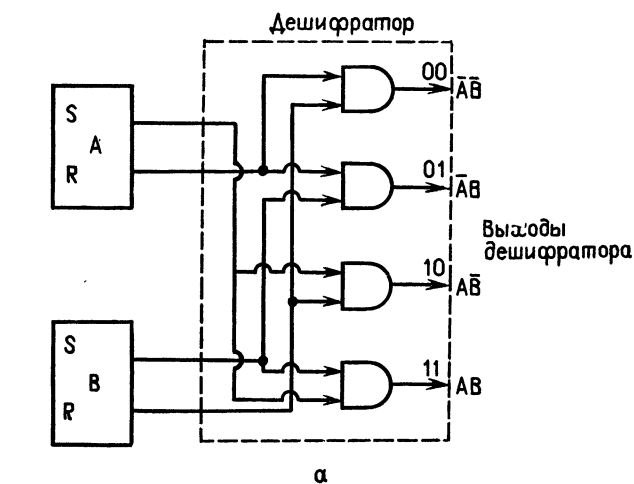


Рис. 7.5.  $\alpha$  — дешифратор с четырьмя выходами;  $\delta$  — дешифратор на три входа.

Вентили И, подключенные к линиям ВЫХОД ячеек памяти, изображенных на рис. 7.3, должны обладать способностью сохранить на выходе высокий уровень, когда несколько выходных линий вентилях И соединены вместе. (Если хотя бы на одном выходе будет 1, на линии останется 1, в противном случае — 0.) Такое соединение называют **монтажным ИЛИ**. На рис. 7.4 все четыре ячейки памяти в первом столбце объединены монтажным ИЛИ, так что если хотя бы на одной из выходных линий ячеек будет 1, то и на общей линии будет 1. (Ячейки памяти на ИС построены таким же образом.)

Теперь если на линию ЧТЕНИЕ (рис. 7.4) подана 1, то все выходные значения триггеров в выбранном горизонтальном ряду будут пропущены на выходные линии соответствующих битов в памяти. Например, если второй ряд в памяти содержит 110 в своих трех ячейках и РАП содержит 01, то на второй выходной линии дешифратора (обозначенной как 01) будет 1, и будут выбраны все входные и выходные вентили, относящиеся к этим трем ячейкам памяти. Если на линии ЧТЕНИЕ будет 1, то выходные значения трех ячеек памяти второго ряда поступят на входы вентилях И в нижней части рисунка, которые выдадут 110 в качестве выходного значения из памяти.

Если на линию ЗАПИСЬ подана 1, а РАП опять содержит 01, то будут выбраны входы второго горизонтального ряда триггеров. Затем входные значения с  $I_1$ ,  $I_2$  и  $I_3$  будут считаны в триггеры во втором ряду.

Очевидно, что это уже память, полностью способная осуществлять чтение и запись. Такая память будет хранить данные в течение любого периода времени и может выполнять операции со скоростью, какую допускают вентили и триггеры. Единственная проблема заключается в сложности памяти. Базовая ячейка памяти (триггер со связанной с ним схемой) сложна, и при большой емкости памяти требуется большой дешифратор.

Для того чтобы продолжить рассмотрение организации памяти, следует сначала подробнее исследовать структуру дешифратора, схемы селекции, которые обычно используются, и, наконец, несколько образцов памяти на ИС.

## 7.4. ДЕШИФРАТОРЫ

Важной частью системы, которая осуществляет выбор ячеек при чтении или записи, является дешифратор. Эта особая схема, которую называют **дешифратором типа «1 из  $2^n$ », декодирующей матрицей** или просто **дешифратором**, обладает тем отличительным свойством, что для каждой из  $2^n$  возможных комбинаций входных двоичных значений, которые могут принять  $n$

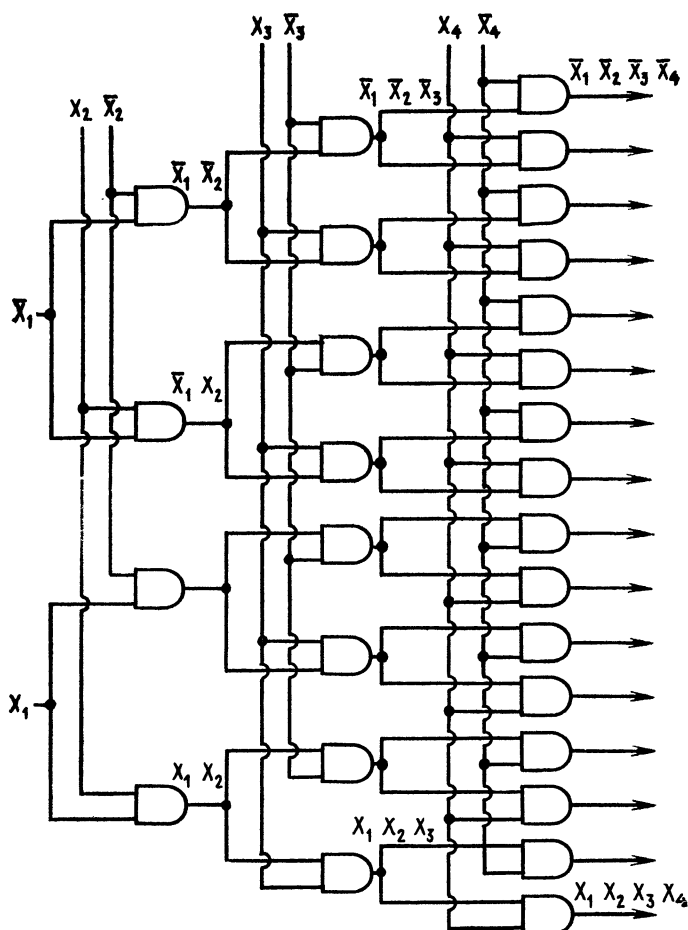


Рис. 7.6. Пирамидальный дешифратор.

входных ячеек, матрица выберет одну из своих  $2^n$  выходных линий.

На рис. 7.5,б изображен одноступенчатый (параллельный) дешифратор, предназначенный для декодирования трех триггеров. Следовательно, есть  $2^3 = 8$  выходных линий, и для каждого из 8 состояний, в которых могут находиться три входа (триггера), будет выбрана одна выходная линия. Дешифраторы такого типа часто строятся с использованием вентиля И на диодах (или транзисторах). При этом выполняется следующее правило: число диодов (или транзисторов) в каждом вентиле И равно числу входов в каждом вентиле И. На рис. 7.5,б это чис-

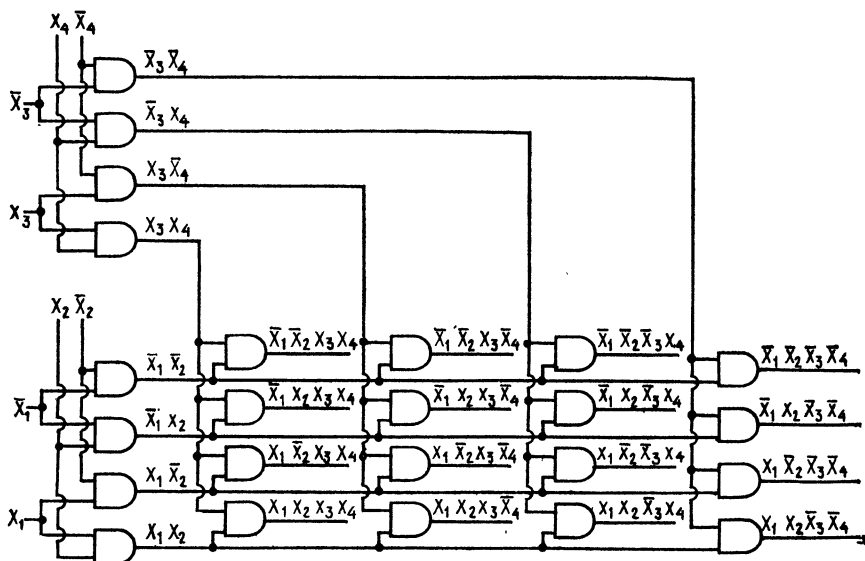


Рис. 7.7. Многоступенчатый дешифратор.

ло равно количеству входных линий (декодируемых триггеров). Далее, число вентилях И равно количеству выходных линий,  $2^n$  ( $n$  — число входных декодируемых триггеров). Общее число диодов, таким образом, равно  $n \times 2^n$ ; чтобы построить схему двоичной декодирующей матрицы, изображенной на рис. 7.5, б, потребуется 24 диода. Можно заметить, что число требуемых диодов резко увеличивается с ростом числа входов в схему. Например, для декодирования восьмитриггерного регистра потребовалось бы  $8 \times 2^8 = 2048$  диодов, если бы дешифратор был построен таким образом.

Вследствие этого при построении схем дешифраторов часто используют некоторые другие типы структур. Одна из таких структур, называемая **пирамидальным дешифратором**, изображена на рис. 7.6. Эта древовидная сеть декодирует четыре триггера и поэтому имеет  $2^4 = 16$  выходных линий, одна из которых выбирается для каждого состояния триггеров. Расчеты показывают, что для построения такой сети потребуется 56 диодов, тогда как для одноступенчатого дешифратора, изображенного на рис. 7.5, необходимо  $2^4 \times 4 = 64$  диода.

На рис. 7.7 показана схема дешифратора еще одного типа. Она называется **многоступенчатым дешифратором**. Заметим, что для такой схемы необходимо всего 48 диодов. Можно показать, что для полной схемы дешифратора потребуется мини-

мальное число диодов, если использовать схему такого типа, как схема, изображенная на рис. 7.7. Разница между количеством диодов, или декодирующих элементов, требуемых для построения схемы, такой, как на рис. 7.7, и количеством диодов, требуемых для схем на рис. 7.5 и 7.6, становится еще более значительной с ростом числа декодируемых триггеров. Однако схема, изображенная на рис. 7.5, имеет преимущество в том смысле, что она наиболее быстродействующая и обладает самой регулярной структурой из всех трех типов схем.

Изучив три типа декодирующих матриц, которые используются в цифровых машинах, мы будем в дальнейшем представлять схемы дешифраторов в виде блока с  $n$  входами и  $2^n$  выходами, имея в виду, что в этом блоке реализован один из трех типов схем, показанных на рис. 7.5—7.7. Часто к дешифраторам подсоединяют только прямые (неинвертированные) входы, а инверторы включают в блок дешифратора. Тогда у трехвходового (трехтриггерного) дешифратора будет только три входные линии и восемь выходов.

## 7.5. СИСТЕМЫ АДРЕСАЦИИ ПАМЯТИ

Организация памяти на рис. 7.4 имеет базовую систему выборки с одномерной (линейной) адресацией. Это простейшая организация. Однако дешифратор при такой системе выборки с ростом размера памяти становится довольно большим.

В качестве примера рассмотрим одноступенчатый дешифратор, изображенный на рис. 7.5, б. Такие дешифраторы широко применяются в ИС благодаря их быстродействию и регулярной (симметричной) структуре.

Рассмотрим теперь дешифратор для памяти емкостью 4096 слов (обычная емкость для ИС). Каждый вентиль И будет иметь 12 входов, а всего потребуется 4096 вентилях И. Если для каждого входа вентиля И необходим один диод (или транзистор), то всего потребуется  $12 \times 4096 = 49\,152$  диода (или транзистора). Большое количество транзисторов — основной недостаток такой организации памяти.

Для реализации **системы двумерной адресации** нужно добавить к базовой ячейке памяти еще один вход ВЫБОРКА. Это показано на рис. 7.8. Теперь для выбора триггера на оба входа ВЫБОРКА 1 и ВЫБОРКА 2 должны быть поданы единицы.

На рис. 7.9 показана память с системой двумерной адресации, использующая такую ячейку. Для этой памяти емкостью 16 слов длиной каждого всего 1 бит требуются два дешифратора. Регистр адреса памяти имеет 4 разряда и, следовательно, 16 состояний. Два сигнала из РАП поступают в один дешифратор и два — в другой.

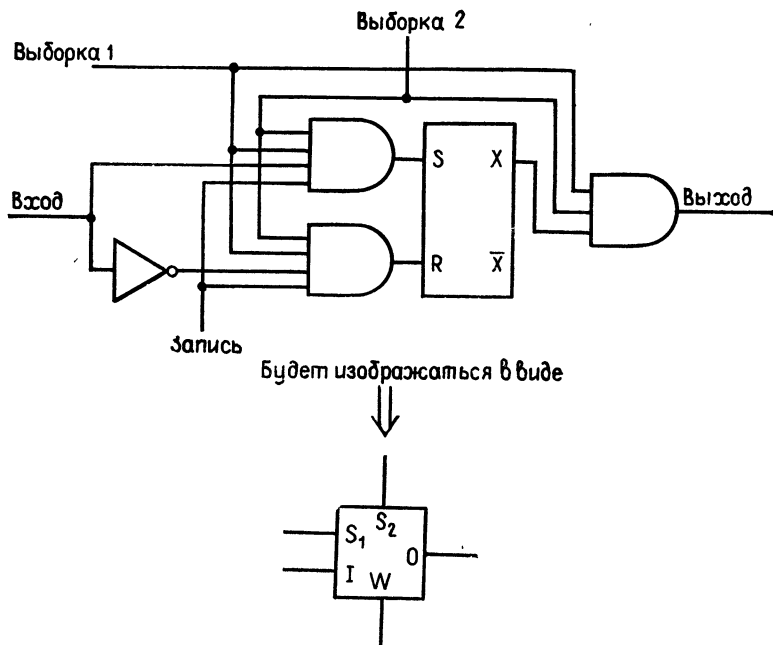


Рис. 7.8. Ячейка памяти с двумерной адресацией.

Проиллюстрируем работу памяти для случая, когда РАП содержит 0111, значение 01 подается на левый дешифратор, а 11 — на верхний дешифратор. Это приведет к выбору второй строки (линии) в левом дешифраторе и крайнего правого столбца в верхнем дешифраторе. В результате только у ячейки (триггера) на пересечении второй строки и крайнего правого столбца будут активизированы обе ее линии ВЫБОРКА (и, следовательно, ее вентили И). В результате только эта ячейка будет выбрана и только в этот триггер можно произвести запись или осуществить считывание из него.

Если РАП содержит 1001, то 1 будет на линии третьей строки левого дешифратора, а также на линии второго столбца. Выбрана будет только ячейка памяти на пересечении этой строки и этого столбца. Если на линию ЧТЕНИЕ подана 1, то будет выполняться считывание из выбранной ячейки; если на линию ЗАПИСЬ подана 1, то будет выполняться запись в нее.

Теперь определим количество необходимых элементов. Если бы при построении памяти емкостью 16 одноразрядных слов применялась система линейной адресации, то потребовался бы дешифратор с  $16 \times 4$  входами и, следовательно, 64 диодами (или транзисторами).



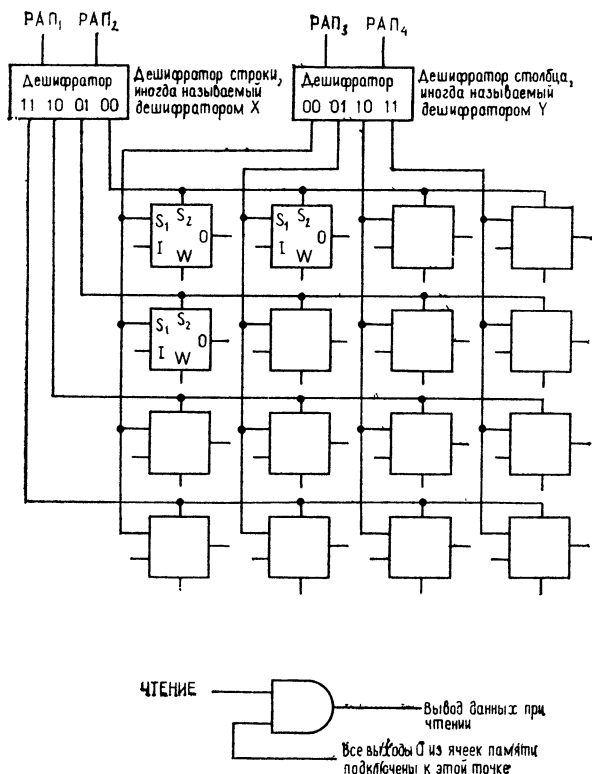


Рис. 7.9. Структура памяти на ИС с двумерной адресацией.

Все входы  $W$  ячеек подсоединены к шине ЗАПИСЬ. Все входы  $I$  ячеек подсоединены к шине ВХОД.

В случае системы двумерной адресации необходимо два дешифратора с 2 входами и 4 выходами и 8 диодов (транзисторов) для каждого дешифратора; таким образом, потребуется 16 диодов для обоих дешифраторов.

Для памяти емкостью 4096 слов по 1 разряду на слово эти цифры более внушительные. Для памяти в 4096 слов при системе линейной адресации требуется 12-разрядный РАП. Следовательно, для дешифратора необходимо  $4096 \times 12 = 49\,152$  диода или транзистора. Для системы двумерной адресации нужны два дешифратора, каждый с шестью входами. Таким образом, для каждого дешифратора понадобится  $2^6 \times 6 = 384$  диода или транзистора, т. е. всего 768 диодов или транзисторов для обоих дешифраторов. Это существенный выигрыш, и он будет еще значительнее для больших объемов памяти.

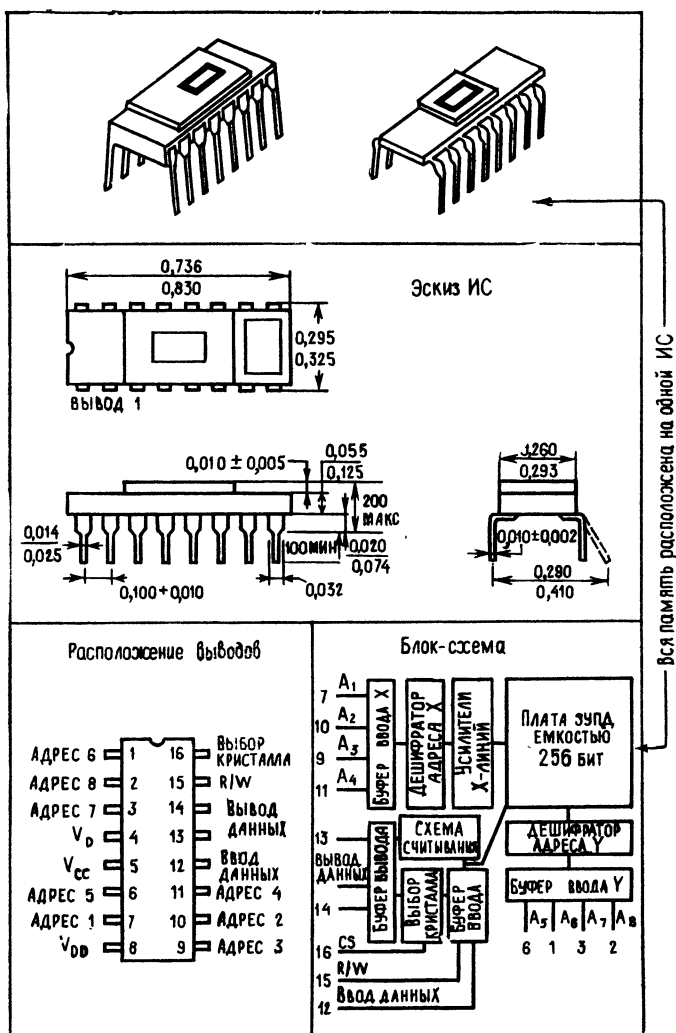


Рис. 7.10. Память на ИС емкостью 256 бит. (С разрешения фирмы Intel.)

Для того чтобы построить память с большим числом разрядов на слово, можно просто для каждого разряда в слове построить такую память, какая показана на рис. 7.9 (за исключением того, что потребуется только один РАП и два исходных дешифратора).

В вышеописанной памяти используется классическая система двумерной адресации. Такая структура применяется в

большинстве ЗУ на магнитных сердечниках и в некоторых ЗУ на интегральных схемах. На рис. 7.10 показана память на ИС с 256 бит на одном кристалле. Видно, что это память с системой двумерной адресации.

Однако в памяти с системой двумерной адресации за упрощение дешифратора приходится расплачиваться усложнением ячейки памяти. В некоторых случаях это усложнение не является дорогостоящим, но часто оно представляет собой проблему и тогда приходится видоизменять схему.

На рис. 7.11 показан другой вариант базовой системы двумерной адресации. Как и в предыдущей схеме, в памяти используются два дешифратора, однако ячейки памяти — такие же базовые ячейки, какие показаны на рис. 7.3.

В схеме выборки используются вентили на входах ЧТЕНИЕ и ЗАПИСЬ для получения требуемой двумерности.

Рассмотрим операцию ЗАПИСЬ. Предположим, что в РАП содержится 0010. Тогда на выходе 00 из верхнего дешифратора будет 1, и будет выбрана верхняя строка ячеек. На выходе 10 нижнего дешифратора будет 1 и она поступит на вентиль И в нижней части диаграммы, а выход этого вентиля будет подан на входы  $W$  в третьем столбце. В результате на входы  $S$  и  $W$  ячеек памяти, находящейся в верхней строке и третьем столбце, будут поданы 1. Ни одна другая ячейка памяти не будет иметь оба входа  $S$  и  $W$ , равными 1, и, следовательно, ни в одной другой ячейке в триггер  $RS$  не может быть установлено входное значение. (Заметим, что все входы  $I$  подключены к входному значению  $D_I$ .)

Рассмотрение других состояний РАП покажет, что для каждого значения существует единственная ячейка памяти, которая будет выбрана для операции записи. Таким образом, для каждого состояния РАП будет производиться запись только в одну ячейку памяти.

Операция чтения происходит аналогично. Если РАП содержит 0111, то на линии 01 верхнего дешифратора будет 1, которая выберет входы  $S$  во втором горизонтальном ряду ячеек памяти. В результате только эти четыре ячейки во всем массиве будут способны записать 1 в выходные линии. (Опять выходы памяти образуют монтажное ИЛИ, так как они соединены вместе в группы по четыре.)

На входе нижнего дешифратора будет 11, и, таким образом, на его нижней выходной линии будет 1. Это значение поступает на вход крайнего правого вентиля И в самой нижней строке, который выберет выход из крайнего справа столбца ячеек памяти. Однако включенный выход есть только у второй ячейки сверху, и поэтому значение этой ячейки в качестве выходного окажется на выходе самого правого вентиля И. Затем это

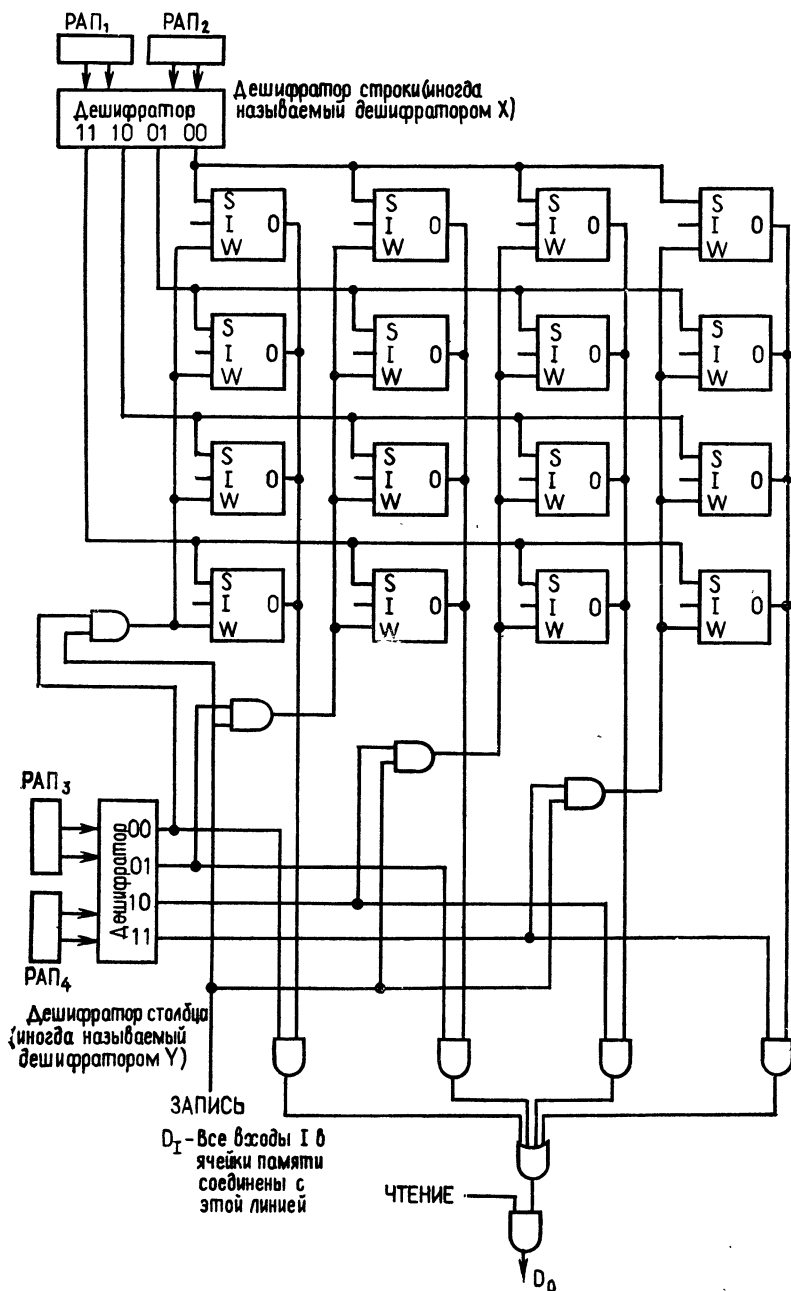


Рис. 7.11. Схема памяти на ИС.

значение проходит через вентиль ИЛИ и вентиль И, открытый сигналом ЧТЕНИЕ, в нижней части диаграммы.

Проверка покажет, что каждое входное значение из РАП выберет единственную ячейку для чтения, и эта ячейка будет та же, какая была бы выбрана для записи, если бы выполнялась операция записи.

Именно такая организация памяти используется в настоящее время в большинстве ЗУ на ИС. Кристаллы содержат до 64Кбит. Число строк и число столбцов массива определяются разработчиками, которые рассчитывают эти числа так, чтобы сократить общее число компонент.

На одном кристалле размещаются все необходимые для памяти схемы, кроме триггеров РАП, которые очень часто не помещаются на кристалле, а входы кристалла соединены непосредственно с дешифратором. Это станет яснее после того, как рассмотрим интерфейс (сопряжение) с шиной.

## 7.6. СВЯЗЬ КРИСТАЛЛОВ ПАМЯТИ С ШИНОЙ ЭВМ

В настоящее время при подключении памяти ЭВМ имеется тенденция соединять центральный процессор (ЦП), который выполняет арифметические операции, осуществляет управление и т. д., с памятью посредством **шины**. Шина — это набор проводников, совместно используемых всеми элементами памяти.

В микропроцессорах и мини-ЭВМ почти всегда для сопряжения памяти применяется шина; и в этом случае элементами памяти являются кристаллы ИС, помещенные в корпус ИС, такие, как изображенные на рис. 7.10.

Шина, используемая для подключения памяти, обычно состоит из 1) группы **адресных линий** для передачи адреса слова в памяти (это фактически выход из РАП на кристалл микропроцессора); 2) группы **проводов данных** для ввода данных из памяти и вывода данных в память и 3) группы **управляющих проводов** для управления операциями чтения и записи.

На рис. 7.12 показана шина микроЭВМ. В целях упрощения мы используем шину памяти только с тремя адресными линиями, тремя линиями выходных данных, двумя линиями управляющих сигналов и тремя линиями входных данных. Следовательно, память имеет емкость 8 трехразрядных слов.

Два управляющих сигнала используются следующим образом. Когда на линию R/W поступает 1, нужно произвести чтение из памяти; когда на линию R/W поступает 0, нужно осуществить запись в память. Сигнал РАЗРЕШЕНИЕ (ME) равен 1, когда должно быть произведено либо считывание, либо запись в память; в любом другом случае он равен 0.

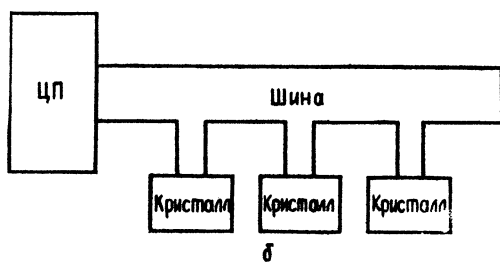
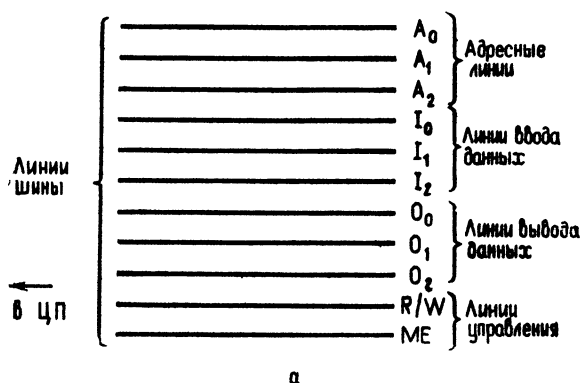


Рис. 7.12. Шина ЭВМ.

а — линии шины; б — шина (ЦП) структуры памяти.

На рис. 7.13 изображен модуль ИС-памяти. Модуль ИС имеет три адресных входа  $A_0$ ,  $A_1$  и  $A_2$ , вход  $R/W$ , выход  $D_0$ , вход  $D_1$  и вход ВЫБОР КРИСТАЛЛА  $\overline{CS}$ . В модуле содержится память в 8 одноразрядных слов.

Кристалл ИС-памяти работает следующим образом. В адресные линии  $A_0$ ,  $A_1$  и  $A_2$  должно быть установлено значение адреса для чтения или записи (рис. 7.13). Если выполняется операция ЧТЕНИЕ, то линия  $R/W$  устанавливается в 1, а линия  $\overline{CS}$  переключается в состояние 0 (нормальным состоянием

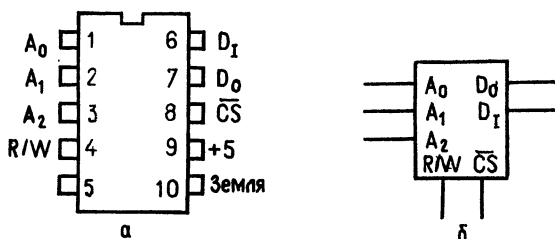


Рис. 7.13. Модуль ИС и условное обозначение ЗУПВД.

а — расположение выводов; б — символическое обозначение кристалла.

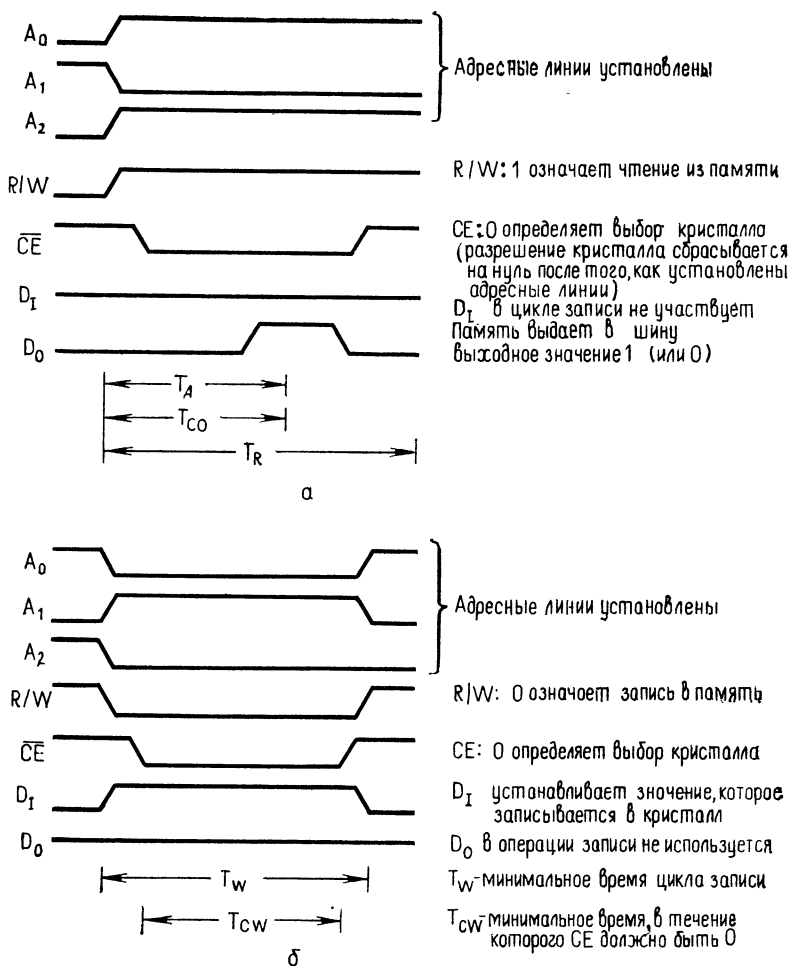


Рис. 7.14. Временные характеристики шины памяти на ИС.  
а — цикл ЧТЕНИЕ; б — цикл ЗАПИСЬ.

линии  $\overline{CS}$  является 1). Затем бит данных может быть считан на линию  $D_0$ . Однако должны быть удовлетворены определенные временные ограничения; они устанавливаются изготовителем ИС. На рис. 7.14 показаны некоторые из них.  $T_R$  — это минимальное время цикла, требуемое для операции чтения. В течение этого периода времени состояние адресных линий не должно изменяться. Значение  $T_A$  — это время доступа, которое представляет собой максимальный интервал времени между моментом, когда адресные линии установлены в стабильное со-

стояние, и моментом, когда данные могут быть считаны из памяти. Значение  $T_{C0}$  — это максимальный интервал времени между моментом, когда линия  $\overline{CS}$  приведена в состояние 0, и моментом, когда данные могут быть считаны.

Временные соотношения шины должны соответствовать этим интервалам времени. Важно, чтобы действия шины были согласованы с кристаллом и шина находилась в состоянии ожидания по меньшей мере в течение времени  $T_A$  после установления адресных линий перед чтением и по крайней мере в течение времени  $T_{C0}$  после того, как на линии  $\overline{CS}$  появился 0 перед чтением. Кроме того, адресные линии должны поддерживаться стабильными в течение по меньшей мере периода  $T_R$ .

Для операции ЗАПИСЬ на адресных линиях устанавливается адрес для записи в память, линия R/W приводится в 0,  $\overline{CS}$  сбрасывается в 0, и записываемые данные подаются на линию  $D_I$ .

Интервал времени  $T_W$  — минимальная длительность цикла ЗАПИСЬ;  $T_H$  — время, в течение которого данные, записываемые в кристалл, должны поддерживаться неизменными. Различные типы памяти имеют разные временные ограничения, которым должна удовлетворять шина. Будем считать, что наша шина соответствует этим ограничениям.

Для формирования из этих модулей (кристаллов) ИС памяти емкостью 8 трехразрядных слов применяется схема соединений, изображенная на рис. 7.15. Здесь адресная линия каждого кристалла соединена с соответствующим адресным выходом на шине микроЭВМ. Вход ВЫБОР КРИСТАЛЛА (линия  $\overline{CS}$  каждого кристалла) соединен через инвертор с выходом МЕ (линия РАЗРЕШЕНИЕ ПАМЯТИ) из микропроцессора, а линия шины R/W соединена с входом R/W каждого кристалла.

Для того чтобы произвести считывание из памяти, ЦП микропроцессора подает адрес чтения на адресные линии, устанавливает 1 на линию R/W, а затем переключает линию МЕ в состояние 1. После этого в каждом кристалле производится считывание выбранного бита в его выходную линию, и ЦП может считать эти значения на свои линии  $I_1$ ,  $I_2$  и  $I_3$ . (Заметим, что выход кристалла является входом шины.)

Аналогично, для того чтобы записать слово в память, ЦП помещает адрес для записи на адресные линии, записываемые биты — на линии  $O_1$ ,  $O_2$ ,  $O_3$ , сбрасывает в 0 значение на R/W и затем переключает МЕ в состояние 1.

В настоящее время слова памяти в микропроцессорах обычно содержат по 8 бит (по 16 бит в некоторых новых больших микропроцессорах). Как правило, у них 16 адресных линий, и, таким образом, в памяти может быть использовано  $2^{16}$  слов.



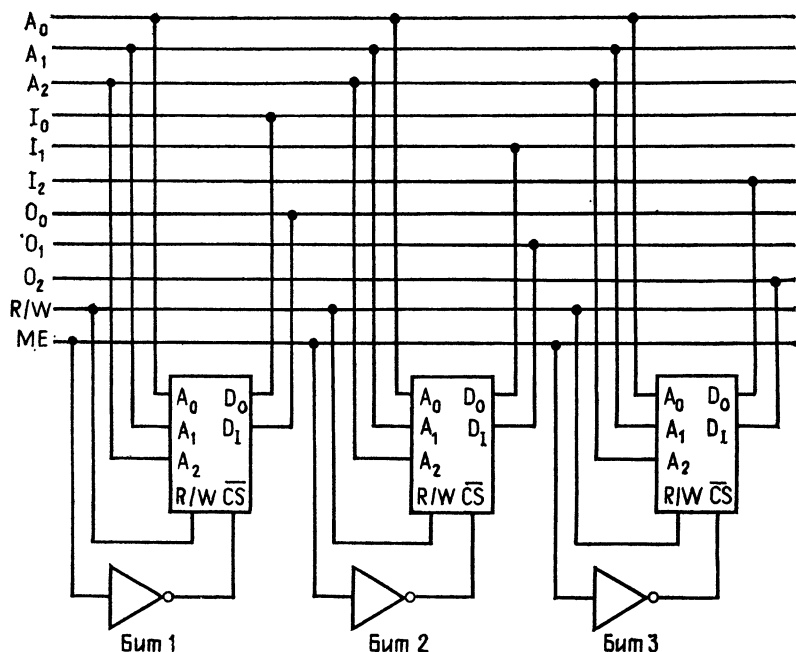


Рис. 7.15. Соединение кристаллов с шиной.

С другой стороны, наблюдается тенденция иметь в кристаллах памяти в основном от 8 до 14 (максимально) адресных линий памяти. Существует простой способ расширения памяти, который иллюстрируется на рис. 7.16.

В этом примере кристаллы опять имеют три адресные линии, а шина микропроцессора включает пять линий. Чтобы сделать возможным соединение, к двум самым старшим битам адресной секции шины подключается дешифратор с двумя входами, в то время как три младших бита соединены с адресными шинами кристалла, как и прежде.

Теперь каждый из выходов дешифратора связан с управляющим сигналом ME с помощью вентиля И-НЕ, так что, когда ME принимает значение 1, в 0 сбрасывается линия ВЫБОР КРИСТАЛЛА (в нормальном состоянии выходы вентиля И-НЕ находятся в состоянии 1). Таким образом, дешифратор выбирает необходимый кристалл, а адресные линии в этом кристалле выбирают ячейку памяти для чтения или записи. Затем путем декодирования в кристалле выбирается определенная ячейка памяти для чтения из нее или записи в нее.

Принцип, проиллюстрированный на рис. 7.16, широко используется в ЭВМ. Кристаллы памяти почти всегда имеют

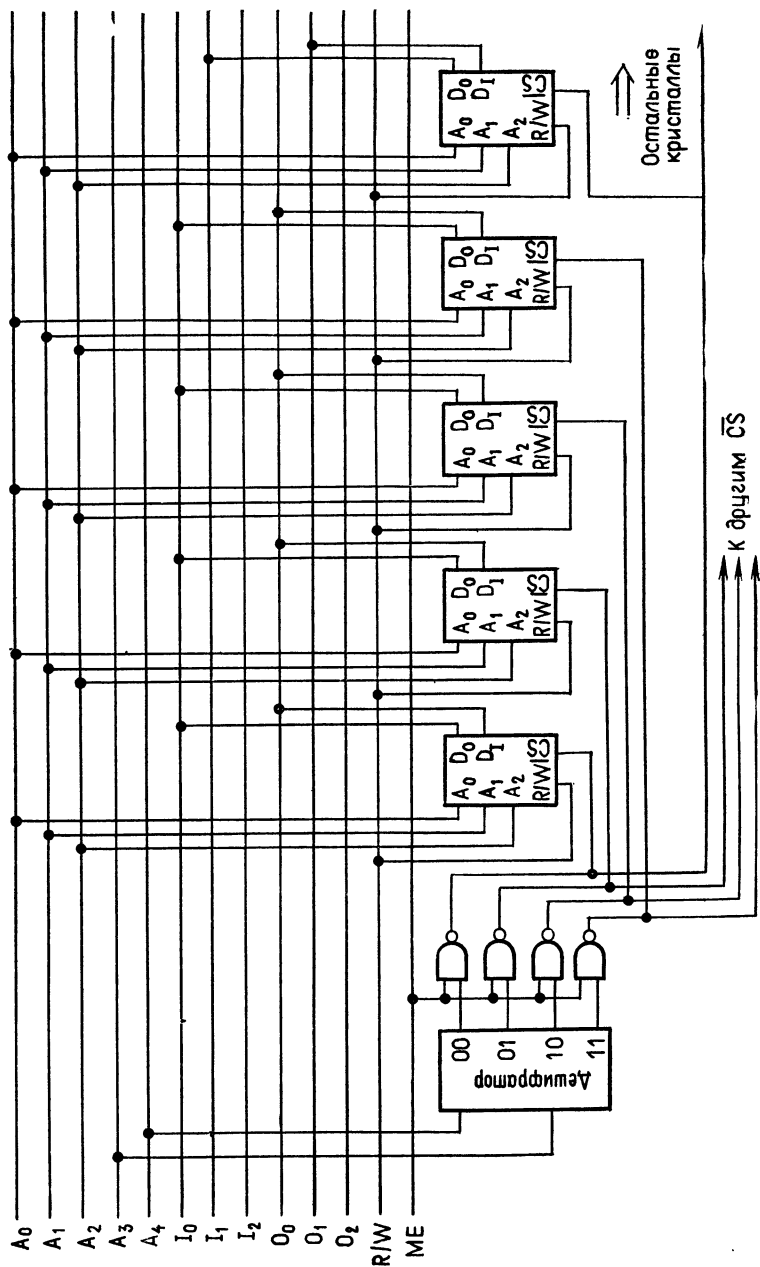


Рис. 7.16. Схема присоединения памяти к шине.

меньше адресных входов, чем шины, и поэтому для использования памяти необходим такой способ расширения. Заметим, что на рис. 7.16 полностью изображен только один бит слова. (Показан также один кристалл из второго бита.) Всего для памяти из 32 трехразрядных слов понадобится 12 таких кристаллов.

Отсюда видно, что, используя микро- или мини-ЭВМ с минимальной памятью, при желании можно расширять память, добавляя кристаллы в соответствии с числом линий адресных шин.

## **7.7. ПОЛУПРОВОДНИКОВЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА С ПРОИЗВОЛЬНЫМ ДОСТУПОМ**

Возможность довольно дешевого изготовления из электронных компонент больших матриц, требующих простых управляющих устройств, в небольших корпусах сделала в настоящее время наиболее популярной память на полупроводниках.

Несмотря на то что имеется много разных схем и устройств, основными сейчас являются шесть видов памяти на ИС.

**1. Биполярные запоминающие устройства.** Это быстродействующие, но довольно дорогие ЗУ с триггерами, изготовленными с использованием стандартных транзисторов с  $p-n$ -переходами.

**2. Статические запоминающие устройства на МОП-транзисторах.** В триггерных схемах этих устройств используются полевые МОП-транзисторы. Они имеют меньшее быстродействие, чем биполярные, но дешевле, расходуют меньше энергии и обладают высокой плотностью упаковки.

**3. Динамические запоминающие устройства на МОП-транзисторах.** При их изготовлении применяются МОП-транзисторы, но в качестве базовой ячейки памяти вместо триггера используется конденсатор (или конденсаторы), в котором происходит накопление заряда; состояние ячейки определяется наличием или отсутствием этого заряда. Для того чтобы воспринимать или накапливать заряд на конденсаторе, используют МОП-транзисторы. Так как со временем будет медленно происходить утечка заряда, необходимо его периодически обновлять; поэтому такие ЗУ называют **динамическими**. (МОП- или биполярные триггерные запоминающие устройства называются **статическими ЗУ**.) Такие ЗУ обладают меньшим быстродействием по сравнению с ЗУ других типов, но они дешевле, расходуют меньше энергии и имеют высокую плотность упаковки.

**4. Запоминающие устройства на комплементарных МОП-транзисторах (КМОП).** В КМОП-транзисторах используются как  $p$ -канальные, так и  $n$ -канальные транзисторы на одной и той же подложке. Благодаря этому технология изготовления услож-

няется. Быстродействие КМОП-транзисторов выше, чем у  $p$ - и  $n$ -канальных МОП-транзисторов, но и стоимость больше.

**5. Запоминающие устройства на сапфи́ро-силиконовых элементах.** Эти ЗУ подобны КМОП. Устройства выполнены на изолирующей подложке из сапфира. Это уменьшает емкость устройств и увеличивает быстродействие. Однако такая память самая дорогая.

**6. Запоминающие устройства на интегральной инжекционной логике (И<sup>2</sup>Л).** В схемах И<sup>2</sup>Л отсутствуют резисторы нагрузки и источники тока, имеющиеся в схемах ТТЛ-типа. Это

Таблица 7.1. Характеристики полупроводниковой памяти

Характеристика	n-МОП-устройства		
	быстродействующие	высокой плотности	биполярные
Число битов на кристалл	4096	65 536	16 364
Время доступа, нс	10	100	40
Рассеиваемая мощность, мВт/бит	0,05	0,01	0,1
Стоимость при средней емкости, цент/бит	0,1	0,05	0,1

сокращает расходы энергии по сравнению с биполярной памятью и обеспечивает большую плотность упаковки, чем в биполярных ЗУ. В результате в ЗУ на И<sup>2</sup>Л сочетается быстродействие биполярных ЗУ с плотностью упаковки ЗУ на МОП-транзисторах. Это ЗУ средней стоимости.

Способы выборки и считывания, а также другие основные принципы, используемые в ЗУ на магнитных сердечниках, применимы также и к полупроводниковым ЗУ. Так как запоминающие устройства выполняют на кристалле, то на том же кристалле можно поместить усилители считывания, дешифраторы и т. д.; довольно часто отдельный кристалл содержит по существу полный небольшой элемент памяти. Следующие разделы охватывают некоторые важные черты и детали полупроводниковых ЗУ. В табл. 7.1 перечислен ряд основных характеристик ЗУ.

## 7.8. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА НА БИПОЛЯРНЫХ ИС

Биполярные ЗУ выполняются с использованием модификаций биполярного транзисторного триггера с высокой плотностью упаковки. Одна из проблем увеличения плотности заключается в изоляции отдельных транзисторов друг от друга; в промышленности разработаны способы уменьшения площади.

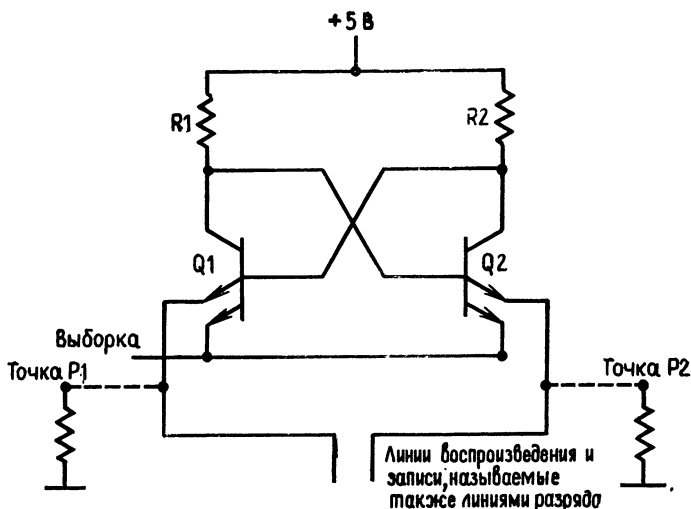


Рис. 7.17. Базовая биполярная ячейка памяти.

необходимой для изоляции транзисторов. Преимуществами биполярных ЗУ являются простота схемы связи, необходимой для совместимости с МОП-элементами, а также быстродействие.

На рис. 7.17 показана базовая биполярная ячейка памяти. Схема состоит из двух перекрестно-соединенных транзисторов, образующих обычный триггер. Каждый транзистор имеет два эмиттера, один из которых используется для выбора ячейки в матрице, а другой — для считывания состояния триггера и записи в триггер.

Линии, присоединенные к верхним эмиттерам в транзисторах Q1 и Q2, образуют так называемую **линию разряда** (данных). Она используется как для считывания состояния триггера, так и для записи в него.

Схема работает следующим образом. Если на линию ВЫБОРКА подано низкое напряжение (около 0 вольт), триггер будет находиться в устойчивом состоянии и ток открытого транзистора будет замыкаться на землю через эмиттер, подключенный к линии ВЫБОРКА. Второй транзистор будет заперт. Пока линии разряда будут иметь нулевое или положительное напряжение, как это имеет место в нормальном состоянии, они не изменят состояния триггера.

Если напряжение на линии ВЫБОРКА повысить до положительного значения (например, до +5 В), то эмиттеры, подключенные к линиям разряда, будут управлять током транзисторов. На рис. 7.17 пунктирными линиями указаны связи линий разрядов через резисторы с землей.

Теперь предположим, что на линию ВЫБОРКА подано высокое напряжение и необходимо считать состояние триггера. Если транзистор  $Q1$  открыт, то ток пойдет через левый резистор, соединенный с землей. В это время транзистор  $Q2$  будет заперт и, таким образом, через крайний правый резистор ток совсем не будет идти. Это означает, что точка  $P1$  будет иметь положительное напряжение относительно точки  $P2$ . (Количественно это положительное напряжение определяется значением резистора в данной части схемы.)

Если транзистор  $Q2$  открыт, а  $Q1$  заперт, ток открытого транзистора  $Q2$  пойдет через правый резистор на землю, а через левый резистор, соединенный с запертым транзистором  $Q1$ , ток идти не будет и  $P2$  будет иметь положительное напряжение относительно  $P1$ .

Из вышеизложенного следует, что при положительном напряжении на линии ВЫБОРКА можно определить состояние триггера, измеряя напряжение в точках  $P1$  и  $P2$ . Это обычно выполняется схемой усилителя, называемого **усилителем считывания**, который измеряет эту разность напряжений и выдает 1 или 0 в зависимости от того, где больше напряжение, в  $P1$  или  $P2$ .

Используя линии разряда, можно также записывать информацию в ячейку. Предположим, что на линии ВЫБОРКА высокое напряжение, и установим напряжение в  $P1$  выше, чем в  $P2$ ; тогда транзистор  $Q2$  откроется и запрет транзистор  $Q1$ . Аналогично повышение напряжения в  $P2$  относительно  $P1$  откроет транзистор  $Q1$  и запрет транзистор  $Q2$ .

Как можно заметить, одни и те же две линии разряда используются и для считывания состояния триггера и для установки заданного значения в выбранную ячейку.

Изображенная на рис. 7.18 схема с двумя линиями ВЫБОРКА и двумя ячейками памяти показывает, как эти ячейки могут объединяться в памяти. Как отмечено на рисунке, могут быть добавлены ячейки и в вертикальном, и в горизонтальном направлениях.

Допустим, что на верхней линии ВЫБОРКА установлено высокое напряжение, а на нижней — низкое. При такой комбинации выбирается верхняя ячейка. Теперь, если необходимо проинициализировать считывание из памяти, усилитель считывания выдаст состояние именно выбранного триггера, так как у другой ячейки (или ячеек) в этом столбце линии ВЫБОРКА будут иметь низкое напряжение, замыкая ток открытых транзисторов на землю, и только в выбранной ячейке ток открытого транзистора будет замкнут на землю через усилитель считывания. Это означает, что в каждом столбце может быть выбрана и считана путем подключения дешифратора к входам линии

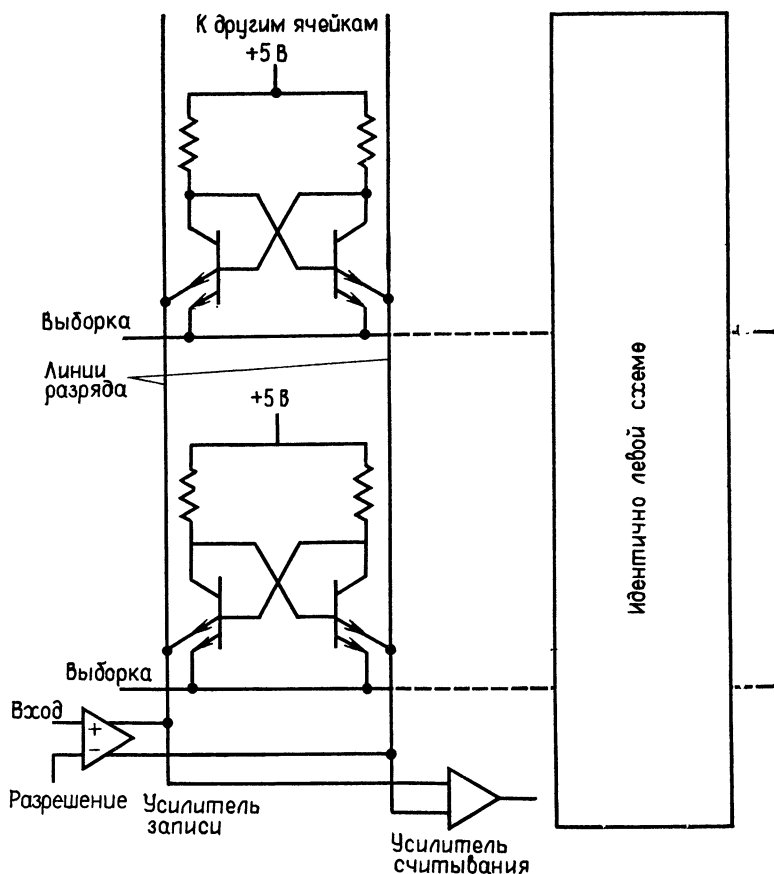


Рис. 7.18. Биполярные ячейки памяти в матрице.

**ВЫБОРКА** только одна ячейка. Аналогично можно записать 1 или 0 в выбранную ячейку в столбце, используя усилитель записи, который установит на одной из линий разряда положительное напряжение относительно другой.

Память в этом случае организована так же, как на рис. 7.11, за исключением того, что обе линии разряда используются и для считывания, и для записи в ячейку памяти. Усилитель считывания используется для считывания выходных значений в столбце, а усилитель записи столбца (который может быть выключен) — для записи в ячейку.

На рис. 7.19 показана биполярная ячейка памяти для системы с двумерной адресацией. Обе линии **ВЫБОРКА** должны иметь высокое напряжение для того, чтобы была выбрана

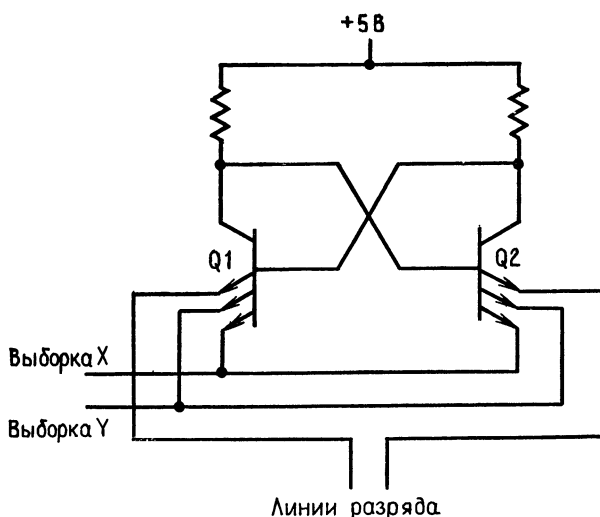


Рис. 7.19. Биполярная ячейка с двумерной выборкой.

ячейка. В противном случае схема действует подобно схемам, показанным на рис. 7.17 и 7.18, используя линии разряда как

*Таблица 7.2. Биполярные запоминающие устройства на ИС*

Структура	Тип микросхемы	Максимальное время доступа, нс	Максимальный требуемый ток, мА
1024×4	6250/6251-1	50	175
1024×8	6282/6283-2	35	170
	6280/6281-1		
4096×1	93L 471	45	165
4096×1	93F 471	30	195

для считывания состояния ячейки, так и для записи в выбранную ячейку.

Ячейку памяти, показанную на рис. 7.19, можно использовать для памяти, имеющей такую же организацию, какая показана на рис. 7.9 и 7.10.

В табл. 7.2 приведены некоторые характеристики биполярных ЗУ и ИС.

## 7.9. СТАТИЧЕСКИЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА НА МОП-ТРАНЗИСТОРАХ

Как только появилась полупроводниковая память, стало ясно, что МОП-устройства отличаются простотой изготовления и экономичностью компоновки. (При производстве МОП-памяти



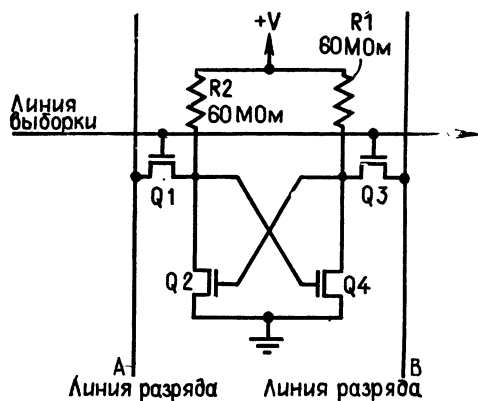


Рис. 7.20. Ячейка статической памяти на МОП-транзисторах. (С разрешения фирмы GTE Microcircuits Division, GTE Products.)

необходим один процесс диффузии, а количество литографий составляет примерно две трети от их количества при производстве биполярных устройств.) МОП-ячейки занимают от половины до четверти площади, занимаемой биполярными ячейками, и, следовательно, обладают значительным преимуществом в стоимости. На рис. 7.20 изображен МОП-триггер, аналогичный биполярному устройству на рис. 7.17.

Выбор ячейки осуществляется просто повышением напряжения на линии ВЫБОРКА. Q1 и Q3 служат в качестве вентилях, подключенных к линиям разряда. Q2 и Q4 образуют обычный триггер. Состояние ячейки памяти можно считать, подавая высокое напряжение на линии ВЫБОРКА, при этом будет открываться либо Q1, либо Q3 в зависимости от состояния триггера. Операция записи выполняется путем подачи высокого напряжения на линии ВЫБОРКА и установки требуемого значения подачей высокого и низкого напряжения на соответствующие линии разряда (рис. 7.17).

В самых ранних ЗУ преобладающим типом МОП-транзисторов было *p*-канальное устройство в режиме усиления (*p*-МОП) с дырками в качестве носителей. Несложная в изготовлении, недорогая и надежная *p*-МОП-схема, однако, является менее действующим устройством с ограниченной плотностью упаковки в БИС. В настоящее время наиболее распространены *n*-канальные транзисторы, имеющие важные преимущества по сравнению с *p*-канальными, а именно низкое рабочее напряжение и более высокую скорость: Так как подвижность электронов выше, чем у дырок, то *n*-канальные транзисторы работают в два-три раза быстрее, чем *p*-канальные того же размера.

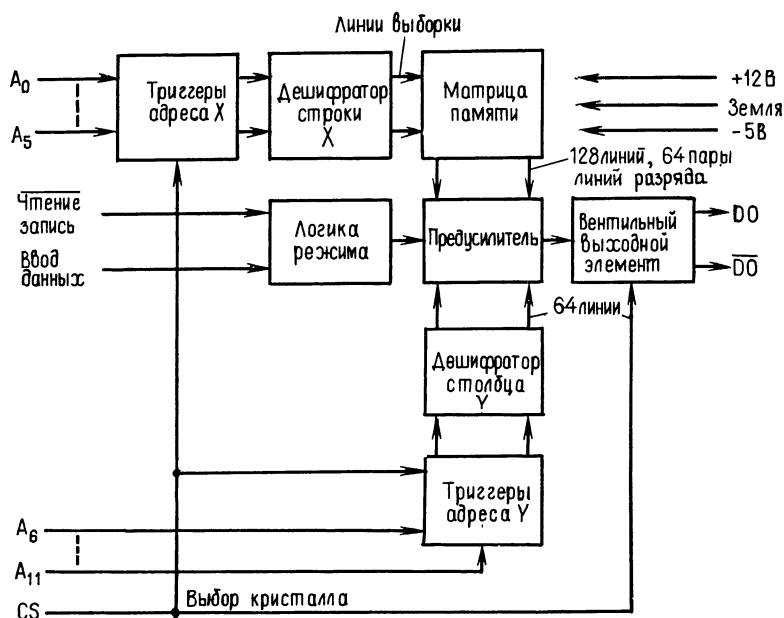


Рис. 7.21. Блок-схема статической памяти на МОП-транзисторах. (С разрешения фирмы GTE Microcircuits Division, GTE Products.)

При равных скоростях  $n$ -канальные транзисторы имеют меньший размер и допускают большую плотность упаковки; используя более высокие уровни легирования подложки, можно увеличить плотность еще больше. Кроме того, имеется много вариантов  $n$ -МОП-технологии: V-МОП, Н-МОП и т. д.

Если сравнить технологии биполярной и МОП-памяти, то можно увидеть, что биполярная память дает выигрыш в скорости, хотя до сих пор ограничения, вызванные необходимостью изоляции между транзисторами, ограничивали плотность упаковки и, следовательно, емкость кристалла памяти. Биполярные элементы могут обеспечить время доступа около 10 нс, тогда как  $p$ -МОП дают 300 нс, а  $n$ -МОП — 20 нс. У МОП-устройств относительно высокие внутренняя емкость и внутренний импеданс, что приводит к большим значениям постоянных времени и времени доступа. На приводимых рисунках не указан тип канала униполярных полевых транзисторов, как это принято в ЗУ. Предполагается, что все устройства являются  $n$ -МОП-устройствами.

Принцип действия 4096-разрядной статической  $n$ -МОП-памяти детально проиллюстрирован на рис. 7.21–7.23. 4096 бит памяти организованы в матрицу из 64 столбцов и 64 строк.

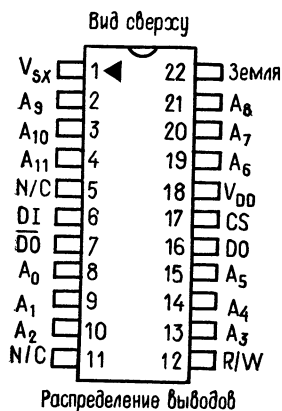


Рис. 7.22. Распределение выводов в корпусе статической памяти на МОП-транзисторах. (С разрешения фирмы GTE Microcircuits Division, GTE Products.)  
 NC — не используется, DI — ввод данных, DO — вывод данных, CS — кристаллы, R/W — чтение/запись.

Доступ к битам памяти производится одновременным декодированием  $X$ -адреса  $A_0—A_5$  для строк и  $Y$ -адреса  $A_6—A_{11}$  для столбцов. [Каждый столбец содержит предусилитель (Presence Amplifier); выходы этих усилителей, объединенные вентилем ИЛИ, подсоединяются к выходному каскаду.] Каждому биту (ячейке памяти) соответствует показанный на рис. 7.20 стандартный триггер, включающий  $R1$ ,  $R2$  (являющиеся в действительности МОП-транзисторами),  $Q2$ ,  $Q4$ ,  $Q1$  и  $Q3$ .  $Q1$  и  $Q3$  используются для соединения ячейки с линиями разряда всякий раз, когда на линию ВЫБОРКА подано высокое напряжение. Для считывания из этой ячейки необходимо подать высокое напряжение на линию ВЫБОРКА; после этого напряжение на одной из линий разряда ячейки станет ниже по сравнению с нормальным высоким значением. Схема считывания, выбранная дешифратором столбца  $Y$ , обнаружит перепад напряжения на выбранной линии

разряда и усилит его. Для записи в ячейку выбирается  $X$ -линия, напряжение на ней с помощью специальной схемы понижается, а выбранная ячейка принимает состояние, заданное ей выбранной линией разряда.

Вход ВЫБОР КРИСТАЛЛА ( $CS$ ) управляет работой памяти. Когда напряжение на  $CS$  низкое, буферные регистры входных адресов, дешифраторы, цепи воспроизведения и выходные каскады поддерживаются в выключенном состоянии и на элементы памяти подается только питание. Когда напряжение на  $CS$  становится высоким, память приводится в работоспособное разрешенное состояние. Тактовые импульсы  $CS$  синхронизируют адресные регистры, выполненные с использованием ТТЛ-схем, сигналы ЧТЕНИЕ — ЗАПИСЬ, входные данные в  $D$ -триггер и открывают выходной каскад. При чтении из ячейки один из двух выходных сигналов будет равен 1. ( $DO$ , если в ячейке записана 1,  $\overline{DO}$ , если в ячейке записан 0.)

Как показано на рис. 7.22, этот кристалл памяти представляет собой модуль с 22 двухрядными выводами. Из большого количества таких кристаллов можно построить большую память с умеренным быстродействием. Время доступа у такого кристалла порядка 50 нс. На рис. 7.23 показаны временные диаграммы, соответствующие одному циклу работы такой памяти.

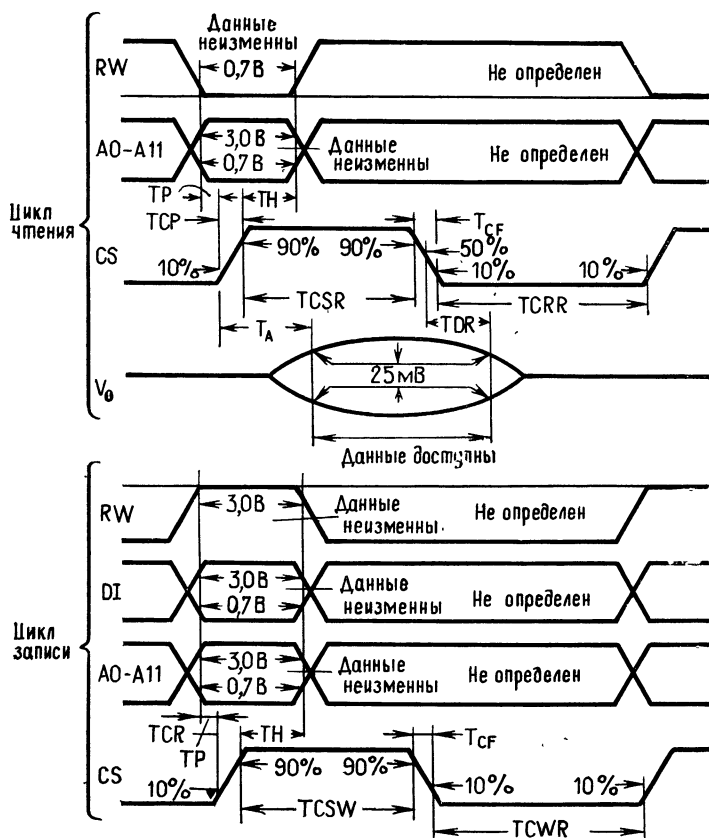


Рис. 7.23. Временные диаграммы. (С разрешения фирмы GTE Microcircuits Division, GTE Products.)

Заметим, что базовая ячейка в такой памяти — это такое устройство, какое показано на рис. 7.3. Двумерную адресацию получают, разбивая память на строки и столбцы и используя много (64) усилителей считывания и усилителей-формирователей (драйверов) ЗАПРЕТА, или разрядных усилителей записи (разрядных драйверов) в этих столбцах. Когда выполняется операция чтения или записи,  $Y$ -дешифратор выбирает соответствующий разрядный усилитель записи или предусилитель считывания. Таким образом, линия ВЫБОРКА СТРОКИ приводит в активное состояние строку ячеек, но на самом деле будет использоваться только одна ячейка на пересечении разрешенной строки с выбранным усилителем считывания или усилителем записи линии разряда (рис. 7.11). Для этой схемы требуется

большее количество разрядных усилителей записи и усилителей считывания, но упрощается конструкция отдельных триггерных ячеек памяти. (Потребуется еще два МОП-транзистора на одну ячейку для того, чтобы осуществить обычную двумерную адресацию, такую, как была ранее описана для биполярной ячейки.) Вообще для двумерной адресации полупроводниковых ячеек требуется большее количество схем (обычно два транзистора на ячейку), но зато используются более простые дешифраторы. Использование только одномерной адресации приводит к

Таблица 7.3. Статические запоминающие устройства на  $n$ -МОП-транзисторах

Тип микросхемы, изготовитель, структура	Рассеиваемая мощность, мВт	Максимальное время доступа, нс	Максимальное время цикла, нс	Число выводов модуля
91L30/AMD (1K×4)	350	500	840	22
9135/AMD (4K×1)	675	80	130	18
SEMI 4402 (4K×1)	500	100	100	22
5256/National (1K×4)	400	250	400	22
5147H/Intel (4K×1)	500	45	60	18
TMS4045/TI (1K×4)	400	150	150	18

увеличению дешифратора, но упрощает отдельные ячейки памяти. В результате часто принимают компромиссные решения.

В табл. 7.3 приведены характеристики нескольких  $n$ -МОП-ЗУ,

## 7.10. ДИНАМИЧЕСКИЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

МОП-ячейки обычно используют в качестве основы для динамической системы памяти. В выбранной ячейке памяти при подаче заряда на конденсатор записывается 1, а при отсутствии заряда или разряде конденсатора запоминается 0. Чтение осуществляется выявлением наличия или отсутствия заряда на конденсаторе. Однако, поскольку всегда имеется некоторая утечка заряда конденсатора, необходимо периодическое его восстановление (регенерация).

На рис. 7.24 показана простейшая, чаще всего используемая, запоминающая ячейка с одним переключающим МОП-транзистором и конденсатором, выполняющим функцию запоминающего элемента (тоже МОП-прибором). Хотя для однотранзисторной ячейки памяти требуются более сложные схемы

считывания и записи, из-за малого размера она удобна для запоминающих устройств с высокой плотностью, такой, как в памяти с 64 000 ячейками на одном кристалле ИС.

На рис. 7.24 линия ВЫБОРКА СТРОКИ выполняет те же функции, как и в других ЗУ: выбор ячейки производится подачей на эту линию высокого напряжения. Линия ДАННЫХ используется для считывания из ячейки.

Ячейки компонуются в двумерный массив с усилителями считывания, подключенными к каждой линии ДАННЫХ и в свою очередь ко всем ячейкам в этом столбце. Когда линия ВЫБОРКА приводит строку в разрешенное состояние, все транзисторы в этой строке переходят в проводящее состояние (транзисторы в других строках заперты). Эти открытые транзисторы переносят любой заряд с конденсатора на линию ДАННЫХ, производя считывание с разрушением. Каждый столбец в массиве имеет собственный усилитель считывания, который распознает заряд, усиливая зафиксированный уровень до логической 1 (или выдавая 0, если нет заряда). (Эти усилители считывания должны быть разработаны очень тщательно, так как заряд ослабляется общей емкостью, образованной линией ДАННЫХ всего столбца.)

Для записи в ячейку подается высокое или низкое напряжение (для 1 или 0) на линию ДАННЫХ, а затем повышается напряжение на линии ВЫБОРКА СТРОКИ (при этом конденсатор заряжается или не заряжается).

На рис. 7.25 показаны временные соотношения при адресации и схема размещения выводов для динамической памяти с произвольным доступом емкостью 16К бит. Так как в кристаллах отсутствует необходимое количество адресных линий, осуществляется мультиплексорная передача адреса с **временным разделением** линий (т. е. двумя частями, одна сразу за другой). Сначала (первая половина) адрес строки помещают на адресные линии  $A_0 - A_6$  и понижают напряжение на  $\overline{RAS}$ , затем на адресные линии  $A_0 - A_6$  помещают адрес столбца и понижают напряжение на  $\overline{CAS}$ . Чтобы использовать память с произвольным доступом такого типа, необходима дополнительная схема для мультиплексирования адресных линий и для генерации сигнала РЕГЕНЕРАЦИЯ. Но, несмотря на это, высокая плотность упаковки и низкая стоимость компенсируют усложнение, связанное с этими специальными схемами; такие ЗУ широко используются.

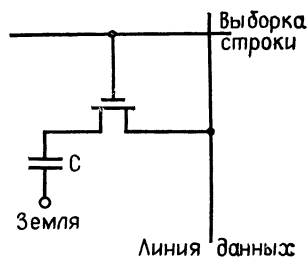


Рис. 7.24. Ячейка памяти с одним транзистором.

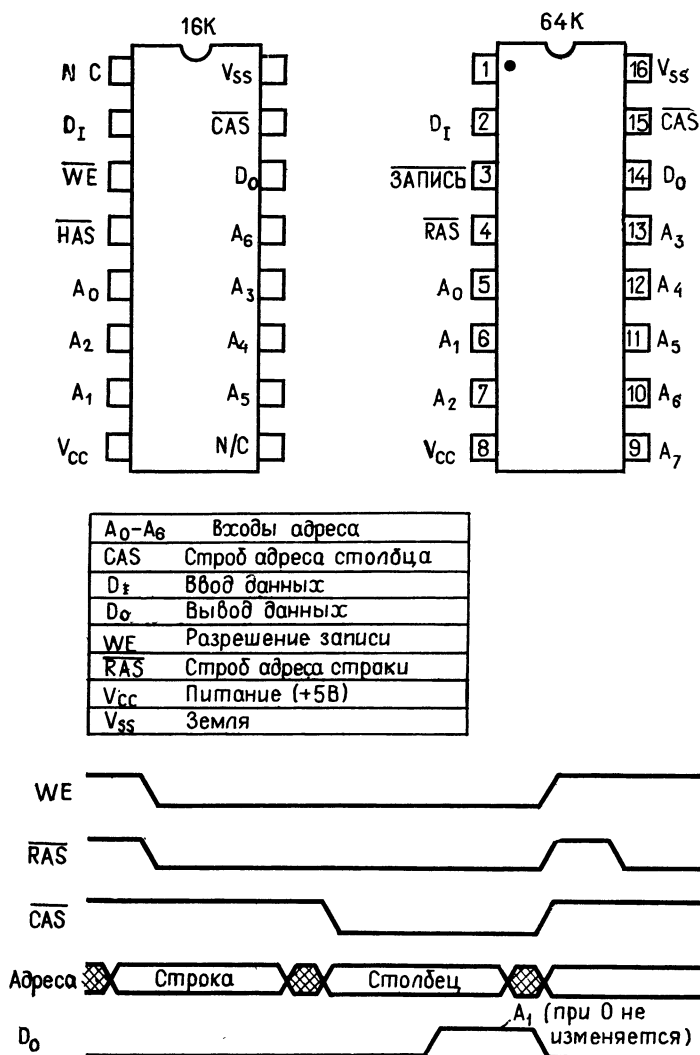


Рис. 7.25. Схема выводов и временные соотношения при адресации для динамической памяти в 16К и 64К.

Главным преимуществом динамических запоминающих устройств на МОП-транзисторах является то, что отдельные ячейки довольно простые. Еще одно преимущество заключается в том, что при отсутствии обращения (чтение или запись) динамическая ячейка памяти не потребляет ток. Это способствует

увеличению плотности упаковки на кристалле. Очевидным недостатком является необходимость регенерации заряда в этих ячейках через каждые несколько миллисекунд, так как происходит постоянная утечка заряда конденсаторов (обычно меньше, чем каждые 2 мс). Для управления регенерацией заряда обычно требуется внешняя схема, но иногда в ЗУ включают специальные схемы, которые выполняют регенерацию по команде. Следовательно, необходимы дополнительные циклы памяти для регенерации заряда, однако на них затрачивается лишь небольшой процент всего операционного времени памяти. В табл. 7.4 перечислены некоторые характеристики динамических ЗУ.

**Таблица 7.4.** Выпускаемые промышленностью динамические *n*-МОП-запоминающие устройства емкостью 16К

Изготовитель	Тип микро-схемы	Время доступа, нс	Время цикла чтения, нс	Время цикла записи, нс	Рабочая мощность, мВт	Число выводов	Выходы
Intel	2116	150	375	375	720	16	Фиксируемые, с тремя состояниями, совместимые с ТТЛ
TI	TMS4070	150	550	550	550	16	Нефиксируемые, с тремя состояниями, совместимые с ТТЛ
Mostek	МК4116	120	375	375	600	16	Нефиксируемые, с тремя состояниями, совместимые с ТТЛ
Motorola	MCM6616	250	375	375	500	16	Фиксируемые, с тремя состояниями, совместимые с ТТЛ

Системы адресации и операции считывания и записи для динамических и статических ЗУ аналогичны, за исключением того, что требуется более сложная временная диаграмма работы. Быстродействие динамических ЗУ обычно меньше статических, но и стоимость их в расчете на 1 бит тоже меньше.

### 7.11. ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Широко применяемым типом запоминающих устройств являются **постоянные запоминающие устройства (ПЗУ)**. Особенностью ПЗУ является то, что из них можно считывать данные,



а записывать нельзя. Таким образом, информация, хранящаяся в этих ЗУ, вводится в память таким способом, что она становится постоянной или полупостоянной. Иногда информация, записываемая в ПЗУ, заносится в память во время ее создания, а иногда используют устройства, в которых можно изменять информацию. В этом разделе будет рассмотрено несколько типов ПЗУ. Они являются характерными для этого класса устройств; в большинстве ПЗУ варьируются принципы, которые будут здесь представлены.

ПЗУ — это устройство с несколькими входными и выходными линиями, причем для каждого входного значения существует

Таблица 7.5. Значения двоичных кодов и соответствующих им кодов Грея

Вход				Выход			
$X_1$	$X_2$	$X_3$	$X_4$	$Z_1$	$Z_2$	$Z_3$	$Z_4$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

единственное выходное значение. Таким образом, ПЗУ физически реализует таблицу истинности, или таблицу комбинаций. Здесь приведена такая типичная (небольшая) таблица (табл. 7.5).

Эта таблица входных и выходных значений фактически является таблицей соответствия двоичного кода коду Грея. Важно отметить, что эту таблицу можно рассматривать, во-первых, как таблицу для вентильной схемы с четырьмя входами и четырьмя выходами и, во-вторых, как список адресов от 0 до 15, задаваемых значениями  $X$ , где содержимое каждого адреса определяет значение  $Z$ .

Таким образом, можно построить вентильную схему, такую, как показана на рис. 7.26, которая будет выдавать пра-

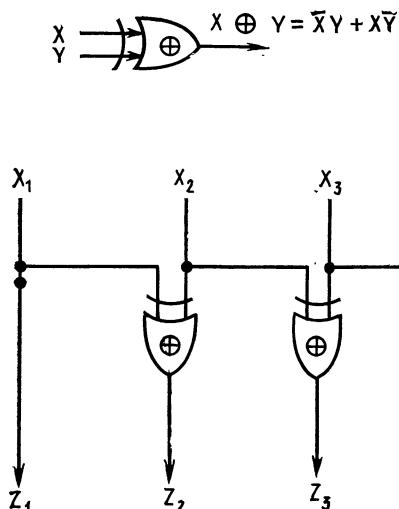


Рис. 7.26. Комбинационная схема для преобразования двоичного кода в код Грея.

вильный выход  $Z$  для каждого входа  $X$ . (Элементы со знаком  $\oplus$  — это сумматоры по модулю 2.)

Табл. 7.5 можно реализовать с помощью памяти на магнитных сердечниках емкостью 16 четырехразрядных слов, в которую записаны 0000 по адресу 0; 0001 — по адресу 1; 0011 — по следующему адресу и т. д. до 1000 по последнему адресу. Если в последующем запретить запись в эту память, ее можно рассматривать как постоянную память, и она будет выполнять те же функции, что и вентильная схема на рис. 7.26.

На рис. 7.27, а показана схема, реализующая табл. 7.5 с использованием схемы дешифратора с четырьмя входами  $X_1$ ,  $X_2$ ,  $X_3$  и  $X_4$  и некоторым количеством диодов. Для заданной комбинации входов (или заданного адреса) высокий уровень напряжения будет только на одной выходной линии дешифратора. Предположим, что входным значением является  $X_1 = 0$ ,  $X_2 = 1$ ,  $X_3 = 1$ ,  $X_4 = 1$ , что соответствует линии 0111 на выходе дешифратора на рис. 7.27, а. Выходные линии дешифратора при помощи диодов соединены с выходными линиями схемы со значениями 1; подобные соединения отсутствуют там, где должны появиться нули. Для входного значения 0111 к выходной линии  $Z_2$  подключен только один диод, так как на выходе должно быть 0100. Аналогично для входного значения 0110 диоды подключены к  $Z_2$  и  $Z_4$ , так как на выходе должно быть 0101. Схема работает следующим образом. В любой момент времени только на одной выходной линии дешифратора может быть вы-

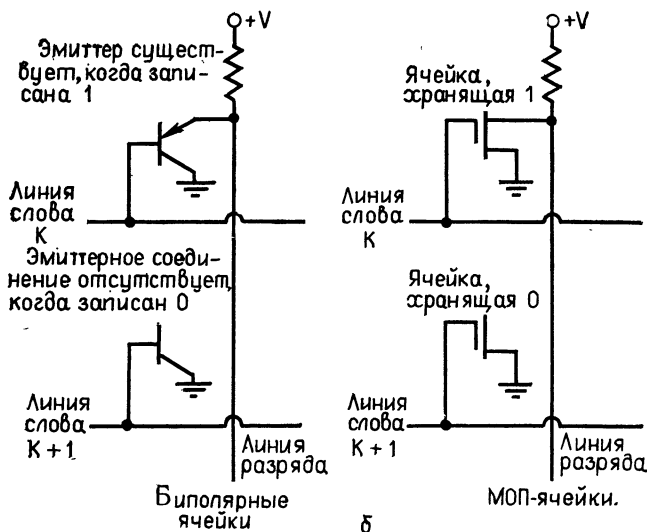
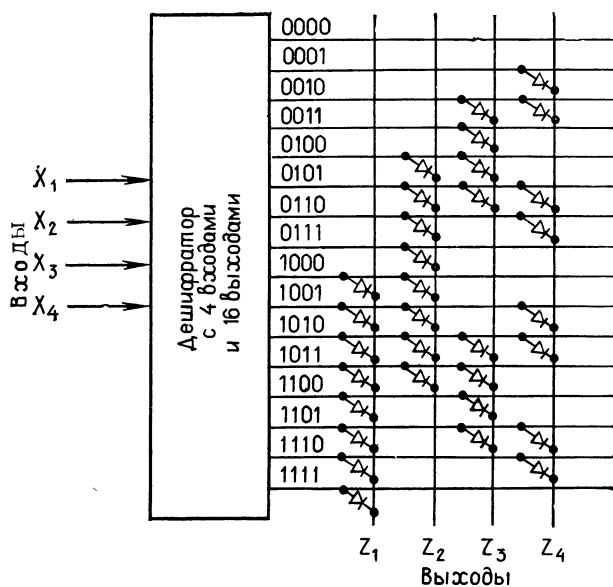


Рис. 7.27. а — ПЗУ на диодах; б — ячейки ПЗУ на МОП- и биполярных транзисторах.

сокий уровень напряжения. Ток с этой линии течет только на те выходные линии, с которыми эта линия соединена диодом. Так, для входного значения 0101 ток пойдет на линии  $Z_2$ ,  $Z_3$  и  $Z_4$ , но не  $Z_1$ , поэтому на  $Z_2$ ,  $Z_3$ ,  $Z_4$  будут единицы, а на  $Z_1$  — нуль.

Вся вышеописанная схема реализует ПЗУ с электронными вентилями (диоды образуют вентили ИЛИ). Используя технологию БИС, матрицы такого вида можно изготавливать без больших затрат в небольших корпусах. Память емкостью 512 восьмиразрядных слов — это средний размер для таких ПЗУ; всего такая память хранит 4096 бит.

В качестве диодов, показанных на рис. 7.27, а, чаще всего используются транзисторы. На рис. 7.27, б показаны типичные ячейки памяти полупроводниковых ПЗУ, использующих биполярные и МОП-транзисторы. Эти ячейки применяются при системе одномерной адресации; обратите внимание на линию выборки слова и линию разряда. В ПЗУ используются обе системы адресации: как одномерная, так и двумерная. ПЗУ на МОП-транзисторах обычно имеют время доступа от 40 до 200 нс; биполярные ЗУ — от 10 до 100 нс.

Если ПЗУ изготовлено таким образом, что пользователь может электрическим (или каким-либо другим способом) записывать информацию в память, то такое ПЗУ называется **программируемым ПЗУ**, или ППЗУ. Часто используют такую схему, где в каждой позиции кристалла памяти предварительно установлены единицы, а нули можно вводить в заданные позиции, помещая на входные линии адрес, а затем повышая напряжение на каждой выходной линии, где должен появиться 0, до определенного значения, при котором разрушается (происходит переключение) перемычка в выбранной ячейке. Иногда память в начальном состоянии во всех позициях содержит нули, а единицы вводятся пользователем. (Производятся также такие устройства, которые программируют ППЗУ, считывая содержимое перфолент, перфокарт, магнитных лент и т. д. и занося его в ППЗУ.)

Выполняя ПЗУ по заказу, изготовители предоставляют пользователю формуляры, которые он заполняет значениями 1 или 0, а изготовитель затем в соответствии с заполненным формуляром изготавливает заказанную маску и производит кристаллы БИС, в которых реализовано содержимое памяти, заданное пользователем. Память на одном таком кристалле может стоить дорого, но при больших партиях изделий обычно стоимость в расчете на один кристалл уменьшается. Такие устройства выпускают емкостью до 64К в одном корпусе ИС.

На рис. 7.28 изображена блок-схема памяти на МОП-транзисторах емкостью 64Кбит, скомпонованной в виде 4096 восьмиразрядных слов. Пользователь помещает адрес на 13 вход-

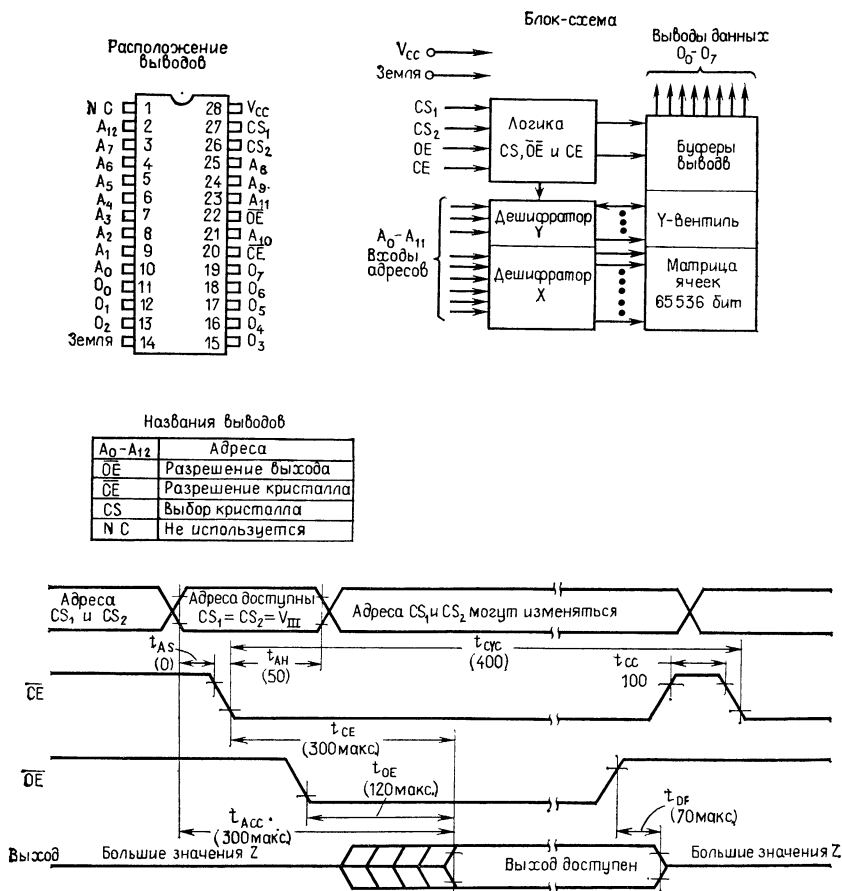


Рис. 7.28. ПЗУ емкостью 64К. (С разрешения фирмы Intel.)

*Примечания*

1. Все времена, указанные в скобках, — это минимальные значения времени в наносекундах, если не указаны другие единицы времени

2.  $t_{DF}$  указывается с момента появления  $\overline{OE}$  и  $\overline{CE}$  в зависимости от того, какой из них появится раньше.

3.  $t_{ACC}$  может быть задержан на 180 нс по отношению к отрицательному фронту  $\overline{CE}$  без влияния на  $t_{ACC}$ .

ных линиях  $A_0 - A_{12}$ , а затем подает высокое напряжение на  $CS_1$  и  $CS_2$  (на  $\overline{CE}$  напряжение должно быть низким). При этом требуемое слово появляется на линиях  $O_0 - O_7$ . Это заказная память, содержимое которой определяется пользователем. Изготовитель затем подготавливает маску, создающую необходимые битовые комбинации на кристалле ИС, и изготавливает ПЗУ с этими комбинациями. Время задержки этой памяти

порядка 75 нс. Если ПЗУ изготовлено так, что содержимое памяти может быть установлено по желанию пользователя, а позднее это содержимое можно стереть и записать новые значения, то такое ПЗУ называют **стираемым программируемым постоянным запоминающим устройством (СППЗУ)**.

Таблица 7.6. Характеристики ПЗУ

Биполярные ПЗУ						
Структура	Тип микросхемы		Время доступа, нс		Потребляемый ток, А	
512×8	6240/6241-1		90		170	
1014×4	6252/6253-1		60		175	
1024×8	6282/6283-1		55		170	
1024×10	6255/6256-1		100		165/175	
2048×8	6275/6276-1		110		190	
МОП ЭППЗУ						
Номер модели	Размер	Структура	Максимальное время доступа, нс	Питание, В	Максимальный активный ток, мА	Ток в режиме хранения, мА
TI 2716	16K	2048×8	450	12,±5	45	45
Intel 2732	32K	4096×8	300	5	40	15
Ti 2532	32K	4096×8	450	5	168	10
МОП ЭИПЗУ						
Номер модели		Размер		Время доступа, мс		
Nitron NC7050		256×4		2—5		
Nitron NC7051		1024×1		2—5		
GI ER3402		1024×4		0,95		
GI ER3800		2048×4		2,6		

Например, некоторые кристаллы изготавливают с прозрачными окнами в крышке корпуса. Облучение кристалла (через эти окна) ультрафиолетовыми лучами разрушает наборы, записанные в кристалле, и новые наборы можно будет записать электрическим способом. Такая перезапись может повторяться многократно.

В табл. 7.6 приведены характеристики некоторых биполярных ПЗУ и ряда электрически программируемых ПЗУ

(ЭППЗУ), содержимое которых можно стирать, подвергая их ультрафиолетовому облучению, а затем программировать (записывать) заново, подавая на входы необходимые значения напряжений. В табл. 7.6 приведены также некоторые характеристики электрически изменяемых ПЗУ (ЭИПЗУ), содержимое которых можно перезаписать посредством подачи в необходимые места в схеме надлежащего входного напряжения.

Ряд фирм производит устройства для программирования ППЗУ. Некоторые из этих устройств управляются с клавиатуры, другие — с ленты, а ряд устройств — с внешних входов, например от микропроцессоров.

Когда необходимо запрограммировать кристалл ППЗУ, следует повысить напряжение, разрешающее запись, что приведет выходные линии в состояния, способные к приему данных. На адресных линиях затем устанавливают адрес ячейки, в которую следует произвести запись. После этого для некоторых кристаллов напряжение на выходных линиях, которые должны иметь единицы, повышают до высокого значения (или же на них подают серию импульсов большой амплитуды) или для других кристаллов на выходные линии подаются сигналы, которые соответствуют нормальным логическим уровням, а на специальный программный вход подается серия высоковольтных импульсов (25 В). В любом случае запись в ячейку памяти можно выполнить путем установления адресных линий, а затем записи желаемого содержимого на выходные линии. Применение стирания (облучая ИС ультрафиолетовым светом в течение некоторого определенного времени) обычно разрушает все содержимое памяти.

## 7.12. ПАМЯТЬ НА МАГНИТНЫХ СЕРДЕЧНИКАХ

В течение примерно двадцати лет в качестве основной памяти ЭВМ главным образом использовали ЗУ на магнитных сердечниках. Однако в последние несколько лет ЗУ на ИС применяется чаще, чем ЗУ на магнитных сердечниках. Тем не менее общий объем памяти на магнитных сердечниках возрастает каждый год, и многие ЭВМ все еще используют их.

Базовое запоминающее устройство, используемое в памяти на магнитных сердечниках, состоит из небольшого тороидального (в форме кольца) тела из магнитного материала, которое называется **магнитным сердечником**. Магнитный сердечник обычно выполнен из ферромагнитного керамического материала.

На рис. 7.29 изображен магнитный сердечник, увеличенный во много раз. Показано, что через сердечник проходит входная обмотка (проводник, шина). Если через эту обмотку пропустить

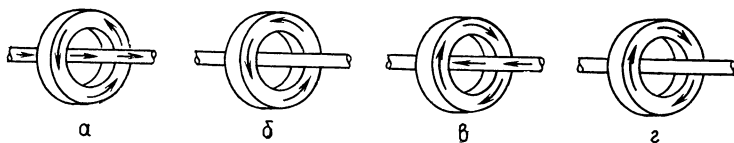


Рис. 7.29. Магнитный сердечник.

*a* — ток подан; *б* — сердечник намагничен, магнитный поток в сердечнике направлен против часовой стрелки; *в* — направление тока изменено на противоположное, состояние сердечника изменилось на противоположное; *г* — ток снят, сердечник остается намагниченным, магнитный поток направлен по часовой стрелке.

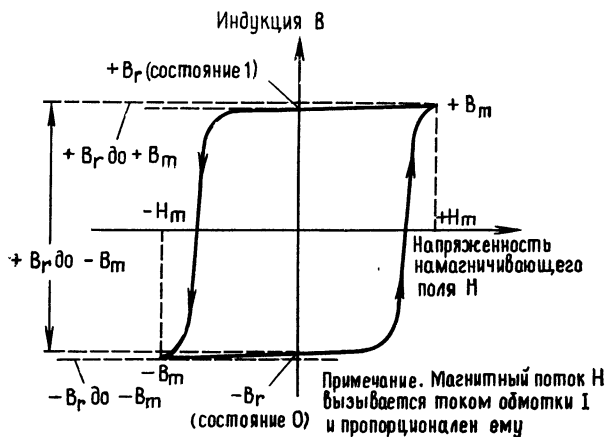


Рис. 7.30. Петля гистерезиса магнитного материала.

ток, то будет создан магнитный поток, направление которого зависит от направления тока через обмотку. Сердечник имеет форму кольца и выполняется из материала с высокой магнитной проницаемостью с тем, чтобы он представлял собой контур с малым магнитным сопротивлением для магнитного потока. В зависимости от направления тока, протекающего через входную обмотку, сердечник намагничивается в направлении по часовой стрелке или против часовой стрелки. Материал, используемый для сердечников, обладает способностью сохранять остаточную намагниченность в такой степени, что, когда намагничивающее поле снимается, сердечник остается намагниченным, сохраняя большую часть своего потока.

Характеристики магнитного сердечника заданного типа обычно изучаются с помощью графика, на котором по абсциссе откладывают напряженность  $H$  намагничивающего поля, создаваемого током обмотки, а по ординате — результирующую плотность потока (магнитную индукцию) через сердечник  $B$  (рис. 7.30). Если во входную обмотку на рис. 7.29 подать периодический ток с чередующейся положительной и отрицатель-



ной полярностью, то каждое значение входного тока будет определять значение напряженности намагничивающего поля  $H$ , приложенного к сердечнику. Если построить график зависимости плотности потока  $B$  в сердечнике от напряженности намагничивающего поля, то получится кривая, называемая **петлей гистерезиса**. На рис. 7.30 достаточная для насыщения сердечника напряженность намагничивающего поля  $H$  достигается при обоих экстремальных, положительном и отрицательном, значениях входного тока. Максимальная плотность потока через сердечник, находящийся в состоянии насыщения при положительном значении  $H$ , обозначена через  $+B_m$ , а при отрицательном значении  $H$  — через  $-B_m$ .

Заметим, что для каждого значения напряженности намагничивающего поля существуют два значения плотности потока, соответствующих возрастанию и убыванию напряженности поля. При изменении напряженности намагничивающего поля от  $-H_m$  до  $+H_m$  значение плотности потока будет перемещаться вдоль нижней части кривой, а при изменении напряженности поля от  $+H_m$  до  $-H_m$  — вдоль верхней части кривой в направлении стрелок до точки  $-B_m$ .

Если на входную шину подать ток, достаточный для насыщения сердечника, а затем снять этот ток, то плотность потока через сердечник вернется в точку  $+B_r$  или  $-B_r$  в зависимости от полярности тока. Эти два состояния называются **состояниями остаточной намагниченности**; когда поток через сердечник принимает одно из этих значений, сердечник будет подобен небольшому постоянному магниту. Если сердечник находится в состоянии  $+B_r$  и подается ток с амплитудой, достаточной для того, чтобы создать намагничивающее поле с напряженностью  $-H_m$ , то полученный поток будет противоположен остаточному потоку в сердечнике  $+B_r$  и поток через сердечник изменит знак на противоположный, а текущее состояние сместится в  $-B_m$ . После того как ток будет снят, большая часть потока останется и плотность потока в сердечнике будет равна значению в точке  $-B_r$  на кривой. Тот факт, что сердечник может находиться в любом из двух особых состояний намагниченности, дает возможность принять одно состояние за 0 ( $-B_r$  на рисунке), а другое — за 1 ( $+B_r$  на рисунке).

Вторая обмотка, называемая **обмоткой считывания** (рис. 7.31, б), используется для того, чтобы определить, содержит сердечник 0 или 1. Чтобы считать состояние сердечника, во входную шину подают ток, достаточный для создания напряженности намагничивающего поля  $-H_m$ . Если сердечник вначале находится в состоянии  $+B_r$ , то рабочая точка сместится по направлению стрелок на рис. 7.30 в точку  $-B_m$ . Если исходное состояние сердечника соответствует  $-B_r$ , то рабочая точка

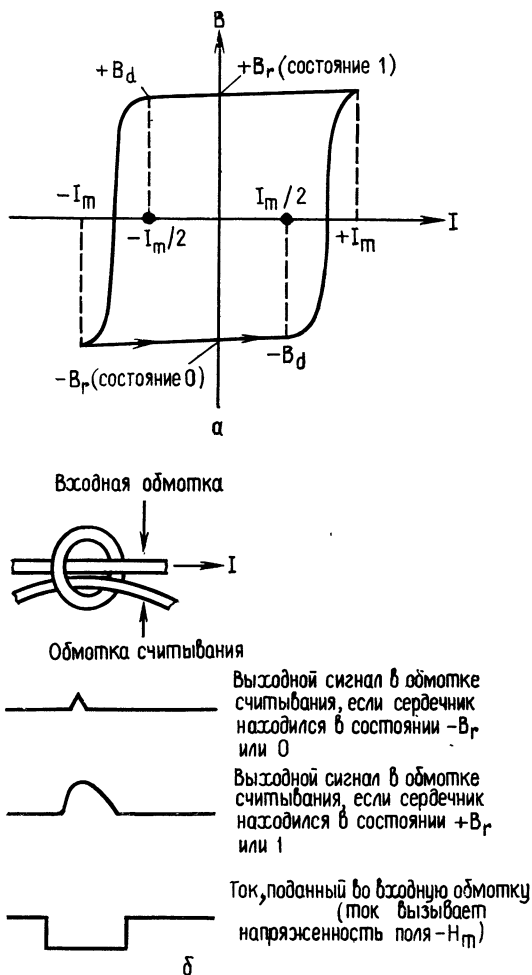


Рис. 7.31. *а* — петля гистерезиса магнитного сердечника, представляющая зависимость плотности потока от тока; *б* — обмотка считывания.

переместится по горизонтали в  $-B_m$ . Величина изменения потока в сердечнике будет совершенно различной для этих двух случаев. Если сердечник вначале находится в состоянии 1 (точка  $+B_r$  на кривой гистерезиса), то направление намагниченности сердечника изменится и поток через сердечник изменится с  $+B_r$  на  $-B_m$ . Величина этого изменения намагниченности сердечника от  $+B_r$  до  $-B_m$  указана на рисунке. Если же сердечник вначале находится в состоянии 0 (точка  $-B_r$

на кривой), то изменение намагниченности будет небольшим только от  $-B_r$  до  $+B_m$ . Изменение плотности потока через сердечник индуцирует в обмотке считывания напряжение, пропорциональное скорости изменения потока. Если сердечник вначале находится в состоянии 0, то величина изменения и скорость изменения потока будут малы и поэтому напряжение, индуцированное в обмотке, будет небольшим. Если же в исходном состоянии сердечник соответствует 1, то величина изменения и скорость изменения потока будут большими и, следовательно, напряжение, индуцированное в обмотке считывания, будет велико. Таким образом, малое выходное значение на обмотке считывания будет означать, что сердечник находится в состоянии 0, а большое будет указывать, что сердечник находится в состоянии 1.

Трудность в использовании этого метода для считывания состояния сердечника заключается в том, что сердечник всегда возвращается в состояние 0. Такой метод считывания, следовательно, является «разрушающим» в том смысле, что сердечник после считывания не содержит информации, которая ранее в нем хранилась.

### 7.13. ЗАПОМИНАНИЕ ИНФОРМАЦИИ В ДВУМЕРНОЙ МАТРИЦЕ МАГНИТНЫХ СЕРДЕЧНИКОВ

Выше было показано, как можно запомнить и считать один бит информации из одного сердечника. Однако в цифровых вычислительных системах необходимо хранить большое количество битов информации; в некоторых больших ЭВМ память содержит более 10 миллионов сердечников.

На рис. 7.31, а показана зависимость плотности потока для магнитного сердечника от тока  $I$  входной обмотки сердечника. Если ток достигнет значения  $+I_m$ , достаточного для насыщения сердечника, а потом подача тока прекратится, то сердечник перейдет в состояние  $+B_r$ , или 1. Если же рабочая точка на кривой находится в  $+B_r$ , причем входной ток в обмотке отсутствует, и подается отрицательный импульс тока с амплитудой  $-I_m$ , то сердечник переключится в состояние 0. Важно отметить, что когда сердечник находится в состоянии 1 и подается импульс тока  $-I_m/2$ , то состояние сердечника не изменится. Вместо этого под воздействием такого тока рабочая точка сместится из  $+B_r$  в  $+B_d$ , а затем при прекращении его действия вернется назад приблизительно в точку  $+B_r$ . Аналогичная картина наблюдается, когда сердечник находится в состоянии 0 (точка  $-B_r$ ). Ток  $+I_m$  переключит сердечник в состояние 1, но ток  $+I_m/2$  только переместит рабочую точку в  $-B_d$ ; когда ток снимется, сердечник возвратится в точку  $-B_r$ .

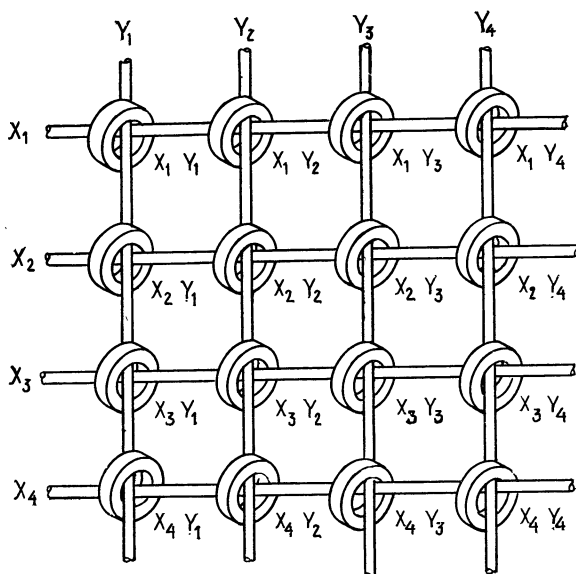
На рис. 7.32, а показаны 16 сердечников, образующих квадратную матрицу. Каждый сердечник имеет две входные обмотки, одна — от группы  $X$  входных линий, другая — от группы  $Y$  входных линий. Предположим, что все сердечники находятся в состоянии 0, и в сердечник  $X_2Y_2$  нужно записать 1. Если на линию  $X_2$  подать ток с амплитудой  $+I_m$ , то сердечник  $X_2Y_2$  переключится в состояние 1. Однако в состояние 1 перейдут также все сердечники, подключенные к входной линии  $X_2$ . Такая же картина наблюдается, когда ток  $+I_m$  подают на входную линию  $Y_2$ ; в этом случае все сердечники, подключенные к этой линии  $Y$ , перейдут в состояние 1.

Если же, однако, токи  $+I_m/2$  одновременно подать на обе линии  $X_2$  и  $Y_2$ , то только сердечник  $X_2Y_2$  получит полный ток  $+I_m$  (рис. 7.32, б). Любой другой сердечник вдоль входной линии  $Y_2$ , а также каждый из остальных сердечников вдоль входной линии  $X_2$  получит ток  $+I_m/2$ . Поскольку эти сердечники начинают перемагничиваться из точки  $-B_r$  петли гистерезиса и получают ток  $+I_m/2$ , из рабочая точка переместится в точку  $-B_d$  петли гистерезиса. Однако рабочая точка не достигнет крутого подъема, ведущего к участку петли гистерезиса со значением  $+B_r$ . После прекращения действия тока эти сердечники, за исключением  $X_2Y_2$ , вернуться в точку  $-B_r$  и окажутся в состоянии 0. Следовательно, только сердечник  $X_2Y_2$  получит полный ток  $+I_m$  и переключится в состояние 1.

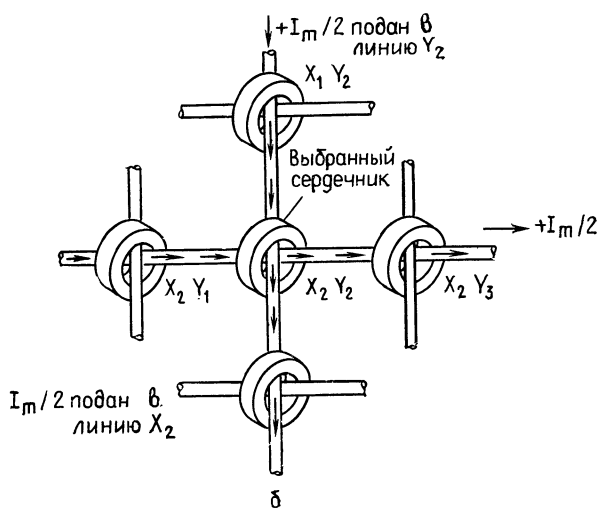
Таким способом можно выбрать из матрицы любой заданный сердечник. Например, сердечник  $X_1Y_3$  можно выбрать, подавая одновременно токи  $+I_m/2$  во входные линии  $X_1$  и  $Y_3$ . Этот способ известен как выборка совпадением токов или полуточков. Сердечник, который получает полный ток  $+I_m$ , называется **полностью выбранным сердечником**, а все сердечники, получающие ток  $+I_m/2$ , называются **полувыбранными сердечниками**. Например, если во входные линии  $X_1$  и  $Y_3$  подан ток  $+I_m/2$ , то  $X_1Y_3$  будет полностью выбранным сердечником, а сердечники  $X_1Y_1$ ,  $X_1Y_2$ ,  $X_2Y_3$  и  $X_3Y_3$  — полувыбранными.

Заметим, что независимо от предыдущего состояния каждого сердечника в матрице токи  $+I_m/2$ , поданные в одну входную линию  $X$  и одну линию  $Y$ , дадут в результате ток, достаточный для переключения только одного сердечника в матрице. Если сердечники, находящиеся в состоянии 0, являются полувыбранными, то они останутся в состоянии 0; если же сердечники, находящиеся в состоянии 1, являются полувыбранными, то они останутся в состоянии 1.

Пользуясь тем же способом, можно считать состояние любого заданного сердечника. Если во входные линии  $X_2$  и  $Y_2$  поданы токи  $-I_m/2$ , то через сердечник  $X_2Y_2$  будет протекать полный ток  $-I_m$ , который вызовет изменение его состояния, если



а



б

Рис. 7.32. а — двумерная плата сердечников; б — прошивка сердечников, выборка  $X_2, Y_2$ .

он содержал 1, и оставит в том же состоянии, если он содержал 0. Полностью выбранный сердечник будет единственным, который может изменять свое состояние, и поэтому единственным способным вызвать заметный сигнал на обмотке считывания. Обмотка считывания прошивается через все сердечники; при выборе определенного сердечника выходной сигнал на обмотке считывания будет представлять состояние только этого сердечника.

Видно, что здесь выполняются две операции: 1) запись информации в сердечник и 2) чтение информации, хранящейся в сердечнике. Подавая ток  $-I_m/2$  или  $+I_m/2$  в соответствующую пару входных линий, в любой сердечник можно записать 0 или 1, а подавая ток  $-I_m/2$  в подходящую пару входных линий и определяя величину выходного напряжения на обмотке считывания, можно считать состояние любого сердечника в матрице.

#### **7.14. БЛОК ПЛАТ СЕРДЕЧНИКОВ В ПАМЯТИ НА СЕРДЕЧНИКАХ**

На рис. 7.33 изображена небольшая **плата памяти на сердечниках**. В этой плате можно выбрать любой сердечник и либо записать 0 или 1 в этот сердечник, либо считать состояние любого сердечника, подавая подходящие токи одновременно в одну из входных  $X$ -линий и одну из входных  $Y$ -линий. Полная память на сердечниках состоит из набора таких плат, составленных в прямоугольную матрицу. Обмотки  $X$  каждой платы соединены последовательно, так что ток, поданный в обмотку  $X_1$  первой платы, должен пройти через обмотку  $X_1$  второй платы и т. д., пока не пройдет через обмотку последней платы в матрице. Если такие платы, как изображенная на рис. 7.33, соединяются в матрице, то обмотка  $X_1$ , верхняя слева, должна соединяться с одним концом обмотки  $X_1$  предыдущей платы сердечников, а обмотка  $X_1$ , нижняя слева, должна соединяться с обмоткой  $X_1$  следующей платы. Если таким образом собрать 10 плат сердечников размера, показанного на рис. 7.33, то импульс входной линии  $X_1$  будет проходить через 80 сердечников. Шины  $Y$  соединены таким же образом. Однако каждая плата имеет свою собственную обмотку считывания, и обмотки считывания не соединены друг с другом; вместо этого к выходу обмотки считывания из каждой платы подключен **усилитель считывания**.

Назовем импульс тока  $+I_m/2$  импульсом **ЗАПИСЬ 1**, так как такой импульс, поданный на обмотку  $X$  или  $Y$ , создает возможность записи 1 в сердечники, через которые проходит эта обмотка. Одновременная подача импульса **ЗАПИСЬ 1** в

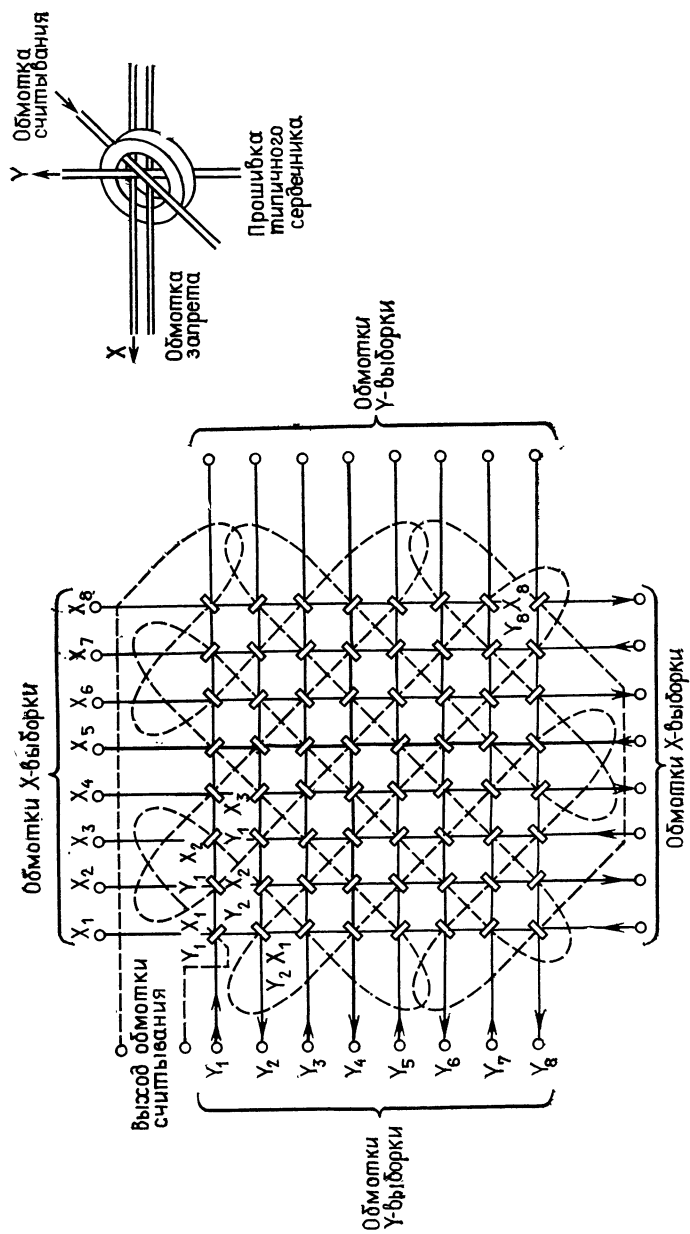


Рис. 7.33. Плата памяти на сердечниках.

выбранную линию  $Y$  и выбранную линию  $X$  вызовет запись 1 в единственный сердечник, находящийся в одном и том же относительном положении на каждой плате. Так, например, если импульс  $+I_m/2$  подается во входные линии  $X_3$  и  $Y_4$ , то сердечник  $X_3Y_4$  на каждой плате получит ток  $+I_m$ . Если же в те же самые  $X$ - и  $Y$ -линии выборки подать импульс  $-I_m/2$ , то на обмотке считывания каждой платы будет воспринят выходной сигнал.

Вообще в матрице столько плат, сколько битов в слове памяти, в которой используется эта матрица. Если в слове 15 бит, то в матрице будет 15 плат; если длина слова 35 бит, то будет 35 плат. Для каждой платы должна быть обмотка считывания, которая позволяет воспринять выходной сигнал от каждого выбранного сердечника.

Но одна проблема все еще остается. Импульс ЗАПИСЬ 1 ( $+I_m/2$ ) на паре входных линий  $X$  и  $Y$  вызовет запись 1 в сердечник каждой платы, находящейся в одном и том же относительном положении, и, следовательно, в каждый разряд слова, но может оказаться необходимым записать 0 в одни сердечники, и 1 — в другие. Поэтому в сердечник добавляется четвертая обмотка. Эта обмотка называется **обмоткой запрета** и используется для того, чтобы запретить запись 1 в заданный выбранный сердечник в плате. Одна обмотка запрета прошивается через каждый сердечник в плате в таком направлении, чтобы ток  $-I_m/2$  на шине был противоположен току ЗАПИСЬ 1. Следовательно, обмоток запрета столько, сколько плат сердечников. Каждой обмотке запрета соответствует свой драйвер (формирователь тока запрета), который при записи 0 в выбранный сердечник платы возбуждается, посылая в обмотку запрета ток с амплитудой  $-I_m/2$ , равной амплитуде тока в линиях выборки  $X$  и  $Y$ . При записи 1 драйвер находится в запертом состоянии. Обмотка запрета прошивается через сердечники таким образом, чтобы напряженность намагничивающего поля тока, поданного в обмотку запрета, была всегда противоположна напряженности намагничивающего поля тока в импульсах ЗАПИСЬ 1.

Так как амплитуда импульса ЗАПИСЬ 1 от драйверов (формирователей адресного тока)  $X$  и  $Y$  равна  $+I_m/2$ , а амплитуда импульса от драйвера ЗАПРЕТА равна  $-I_m/2$ , то если одновременно подаются импульс ЗАПИСЬ 1 от заданной пары драйверов  $X$  и  $Y$  и импульс ЗАПРЕТА, то полный ток через выбранный сердечник будет  $(+I_m/2) + (+I_m/2) + (-I_m/2) = +I_m/2$ . Этот ток недостаточен для переключения сердечника в состояние 1. Следовательно, для того чтобы записать 0 или 1 в выбранный сердечник заданной платы, следует сначала переключить все сердечники в состояние 0, пропуская импульс



— $I_m/2$  по соответствующей паре линий выборки, а затем включить драйвер запрета для каждой платы, в которой нужен 0, и выключить — для каждой платы, в которой требуется 1, подав в линии ВЫБОРКА  $X$  и ВЫБОРКА  $Y$  импульс ЗАПИСЬ 1.

### 7.15. ВРЕМЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ

Одна и та же временная последовательность применяется и для записи информации в память на сердечниках, и для считывания информации из памяти. Общее время, которое занимает вся временная последовательность, называется **циклом памяти**; оно является одним из главных факторов, определяющих скорость ЗУ на сердечниках. На рис. 7.34 показан полный цикл памяти. Рассмотрим сначала операцию записи, а затем операцию считывания. Предположим, что в полной матрице имеются пять плат сердечников, а в каждой плате содержатся 64 сердечника, как на рис. 7.33.

**1. Запись в память на сердечниках.** Пусть требуется записать двоичное число 10100 в сердечник с адресом в памяти  $X_3Y_4$ . Отсюда следует, что нули должны быть записаны в выбранные сердечники плат 2, 4 и 5.

Сначала выбираются необходимые драйверы. Так как нужно выполнить запись в сердечник  $X_3Y_4$  каждой платы, то возбуждаются драйверы, подсоединенные к линии  $X_3$ , а также драйверы, подсоединенные к линии матрицы  $Y_4$ . При подаче импульсов ВРЕМЯ ЧТЕНИЯ в сердечник  $X_3Y_4$  каждой из пяти плат записываются нули. Через 0,4 мкс каждый из выбранных сердечников будет содержать 0. Через 0,5 мкс после начала временной последовательности чтение-запись включаются драйверы ЗАПРЕТА, соединенные с платами с 2, 4, 5; кроме того, начинается время записи. На платах 1 и 3 в выбранный сердечник будет записана 1, а на платах 2, 4, и 5 в результате вычитания тока запрета из тока совпадения в обмотках выбранных сердечников эти сердечники останутся в состоянии 0. После этой последовательности импульсов выбранные сердечники плат 1 и 3 будут содержать 1, в сердечниках плат 2, 4 и 5 будет 0, и, следовательно, машинное слово 10100 будет записано в ячейке памяти  $X_3Y_4$ .

**2. Считывание из памяти на сердечниках.** Предположим, что позднее потребовалось считать данные из ячейки  $X_3Y_4$ , в которой до этого была выполнена вышеописанная операция записи. При этом сохраняется временная последовательность, представленная на рис. 7.34, несмотря на то что есть некоторые различия в ее выполнении. Сначала в выбранные сердечники подаются токи чтения (в интервале от 0 до 0,5 мкс на рис. 7.34). Если в этом периоде времени на входе усилителя считывания,

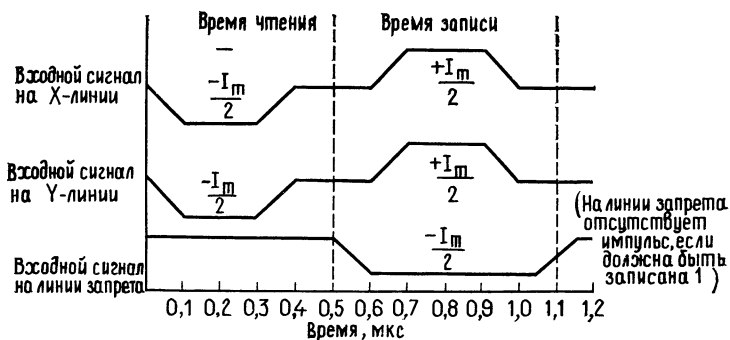


Рис. 7.34. Временные диаграммы памяти на сердечниках.

соединенном с заданной платой, получен большой сигнал, т.е., следовательно, выбранный сердечник в этой плате содержал 1; если получен малый сигнал, то, значит, сердечник содержал 0. Следовательно, обмотки считывания, подключенные к платам 1 и 3, выдадут сигналы, указывающие на наличие 1, а обмотки считывания, подключенные к платам 2, 4 и 5, выдадут малые сигналы, которые указывают на наличие 0. Отсюда следует, что в памяти находилось слово 10100. Теперь нужно снова записать это слово в матрицу памяти. Выходной сигнал каждой обмотки считывания усиливается и используется для установки ячейки памяти (сердечника) в состояние 0 или 1 в течение времени чтения, а содержимое каждой из этих ячеек памяти затем используется для управления драйверами ЗАПРЕТА в течение времени записи. Следовательно, только драйверы запрета, соединенные с платами 2, 4 и 5, будут включаться сигналами с соответствующих ячеек памяти и проводить ток в течение времени записи. Выбранные сердечники этих плат при этом останутся в состоянии 0. По истечении времени записи выбранные сердечники будут опять содержать 10100, как и до операции чтения.

### 7.16. УПРАВЛЕНИЕ АДРЕСНЫМИ ЛИНИЯМИ X И Y<sup>1)</sup>

Рассмотрим путь следования тока через память на сердечниках с помощью простой идеализированной модели. В качестве запоминающих устройств как для регистра адреса памяти, так и для регистра буфера памяти обычно используются триггеры. Блок-схема, приведенная на рис. 7.35, иллюстрирует действие системы выборки для памяти на сердечниках, в которой каждая плата такого же размера, как плата на рис. 7.33. Так

<sup>1)</sup> Эти линии называются также координатными линиями. — Прим. ред.

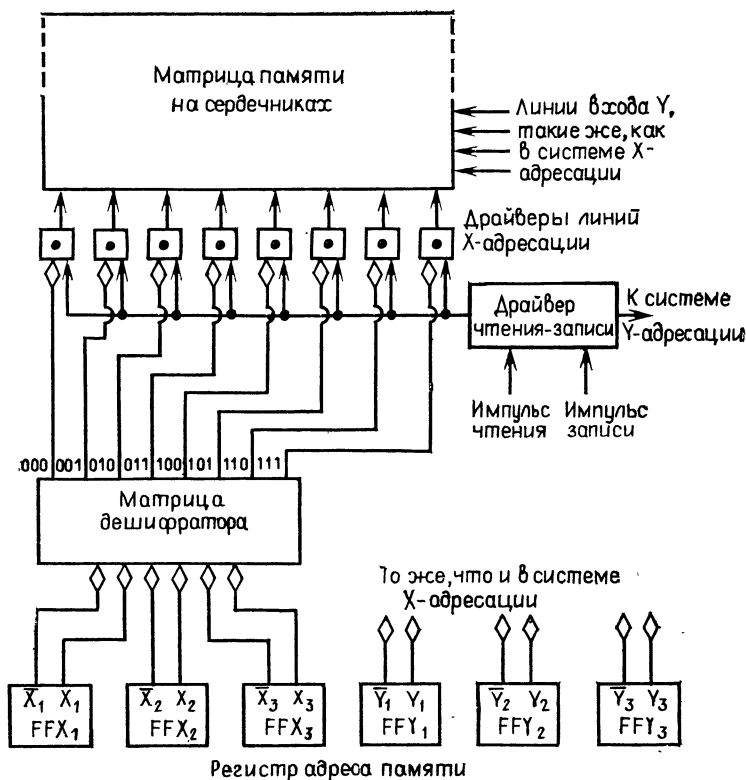


Рис. 7.35. Система X- и Y-адресации.

как имеются восемь адресных линий  $X$  и восемь адресных линий  $Y$ , то для выбора  $X$ -линий потребуются три триггера, и для выбора  $Y$ -линий — еще три. Таким образом, в регистре адреса памяти имеются шесть триггеров, так что можно выбрать любую из 64 ( $2^6$ ) ячеек в каждой плате. Триггеры в изображенном регистре обозначены через  $X_1, X_2, X_3$  и  $Y_1, Y_2, Y_3$ , чтобы показать, что первые три используются для выбора обмотки  $X$ , а вторые три — для выбора обмотки  $Y$ .

**1. Драйверы адресных линий  $X$ .** На рис. 7.35 показаны восемь драйверов адресных линий  $X$ , каждый из которых имеет два входа: один — от дешифратора, другой — от драйвера ЧТЕНИЯ-ЗАПИСИ. Выходные токи из драйвера ЧТЕНИЯ-ЗАПИСИ могут быть либо положительными, либо отрицательными; следовательно, драйверы адресных линий  $X$  способны пропускать ток в любом направлении. Драйверы адресных линий  $X$  выполняют также и функции вентиля И, так как пропускают

ток только тогда, когда выходной сигнал из матрицы дешифратора представляет собой 1. Только один из драйверов адресной линии  $X$  будет включен дешифратором в любой заданный момент времени. Например, если триггеры  $X$  в регистре адреса памяти содержат 000, включенным будет только крайний левый драйвер на рисунке. Все драйверы адресных линий  $X$  получают от драйвера ЧТЕНИЯ-ЗАПИСИ вначале импульс тока ЧТЕНИЕ, а затем импульс ЗАПИСЬ (противоположной полярности). Так как только один драйвер адресной линии  $X$  открывается сигналом 1 дешифратора, то только одна из обмоток  $X$  в матрице получит управляющие токи. Такая же картина наблюдается для системы адресных линий  $Y$ , идентичной системе  $X$ . Вследствие того что во время каждого цикла памяти импульс тока появляется только в одной линии  $X$  и одной линии  $Y$ , лишь один сердечник в каждой плате получит импульс полной выборки ЧТЕНИЕ-ЗАПИСЬ.

**2. Драйвер ЧТЕНИЯ-ЗАПИСИ.** Драйвер ЧТЕНИЯ-ЗАПИСИ получает два тактовых импульса за каждый цикл памяти. Первый — импульс ЧТЕНИЕ — вызывает передачу положительного импульса тока ПОЛУВЫБОРКА от драйвера ЧТЕНИЯ-ЗАПИСИ к обоим драйверам адресных линий  $X$  и  $Y$ . Второй входной импульс к драйверу ЧТЕНИЯ-ЗАПИСИ — это импульс ЗАПИСИ, который вызывает подачу отрицательного импульса тока на драйверы адресных линий  $X$  и  $Y$ . Ток от драйверов ЧТЕНИЯ-ЗАПИСИ обычно порядка 0,1—1 А.

В результате работы регистра адреса памяти и связанной с ним схемы один сердечник в каждой плате получит импульс тока ПОЛНОЕ ЧТЕНИЕ, а затем импульс ПОЛНАЯ ЗАПИСЬ. Регистр адреса памяти обычно загружается непосредственно из команды, которая интерпретируется машиной; это значит, что адресная часть слова считывается непосредственно в регистр адреса памяти.

## **7.17. БУФЕРНЫЙ РЕГИСТР ПАМЯТИ И СВЯЗАННЫЕ С НИМ СХЕМЫ**

Буферный регистр памяти состоит из триггерного регистра, содержащего один триггер для каждой платы в матрице. Число плат в матрице, а следовательно, и длина буферного регистра памяти равны числу битов в базовом машинном слове. АЛУ и устройство управления ЭВМ обычно взаимодействуют непосредственно с буфером памяти, либо загружая в него слово, записываемое в память, либо считывая из него.

На рис. 7.36 изображены два триггера буферного регистра памяти вместе со связанной с ними схемой. Для каждого бита в буферном регистре памяти имеется идентичная схема. Могут

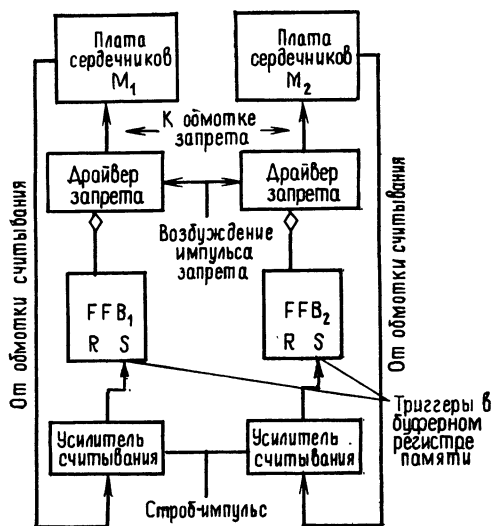


Рис. 7.36. Действие буферного регистра памяти.

иметь место две различные последовательности действий, одна из которых приводит к считыванию слова из памяти, а другая — к записи слова в память.

**1. Чтение из памяти.** Адрес слова, считываемого из памяти, сначала загружается в регистр адреса памяти (рис. 7.35), а затем буферный регистр памяти очищается. За время чтения (рис. 7.34) выбранный сердечник в каждой плате получает импульс полного тока и устанавливается в состояние 0. Если выбранный сердечник в заданной плате содержит 1, то появляется сигнал на обмотке считывания. Его усиливают с помощью усилителя считывания, а затем используют для установки в 1 триггера буфера памяти, который управляет драйвером ЗАПРЕТА для платы сердечников. Если выбранный сердечник в плате содержит 0, сигнал, полученный усилителем считывания, будет небольшим и на выходе усилителя считывания не появится импульс, а триггер останется в состоянии 0.

Усилитель считывания «стробится» узким импульсом (рис. 7.36), так как, несмотря на то что сердечник, содержащий 0, создает при считывании небольшой сигнал, многие сердечники в каждой плате получают импульсы ПОЛУВЫБОРКИ, и каждый из них вызовет небольшие помехи в обмотке считывания. Если это аддитивные помехи, то они могут приблизиться к уровню сигнала от сердечника с противоположной полярностью. Обнаружено, что помехи, генерируемые полувывбранными сердечниками, затухают вскоре после начала импульса ПОЛУВЫБОРКИ. Поэтому усилитель считывания стробируется в

конце времени чтения, когда ток считывания в сердечнике в двух его противоположных состояниях будет иметь наибольшее значение по отношению к помехам из полувыбранных сердечников. Во время строб-импульса в линию УСТАНОВКИ триггера пропускается только выходной сигнал.

Во время записи (рис. 7.34) каждый выбранный сердечник получает от системы адресации  $X$  и  $Y$  ток полной выборки. В результате операции чтения все эти сердечники предварительно устанавливаются в состояние 0. Однако теперь триггеры буфера памяти содержат 1 для каждой платы, в полностью выбранном сердечнике которой была записана 1. Выходы этих триггеров используются для включения драйверов ЗАПРЕТА каждой платы, в которой должен остаться 0. Все остальные полностью выбранные сердечники будут установлены в состояние 1. Следовательно, выход 0 каждого триггера используется для включения драйвера ЗАПРЕТА для платы, связанной с триггером. По истечении времени записи выбранный сердечник в каждой плате будет находиться в своем исходном состоянии.

**2. Запись в память на сердечниках.** При записи слова в память система адресации  $X$  и  $Y$  проходит через ту же временную последовательность, что и при чтении слова из памяти. Однако действие буферного регистра памяти отличается. Слово, записываемое в выбранную ячейку, загружается в буферный регистр памяти, а адрес памяти для записи — в регистр адреса памяти. Во время чтения усилитель считывания не стробируется; следовательно, состояние буферного регистра памяти не меняется.

По истечении времени чтения все выбранные сердечники будут установлены в 0. Во время цикла записи все эти сердечники получают импульсы полного тока ЗАПИСЬ от систем адресации  $X$  и  $Y$ . Однако у всех плат, в которых должны сохраниться 0, драйверы ЗАПРЕТА возбуждаются триггерами буфера памяти (рис. 7.35 и 7.36), а импульсы полутoka от этих драйверов гасят ток адресации, оставляя полностью выбранные сердечники в состоянии 0. Все остальные полностью выбранные сердечники устанавливаются в состояние 1.

По истечении времени записи слово, находящееся в буферном регистре памяти, будет записано в сердечники.

## **7.18. ОРГАНИЗАЦИЯ ПАМЯТИ НА СЕРДЕЧНИКАХ И ПРОШИВКА СЕРДЕЧНИКОВ**

Описанная выше организация памяти на сердечниках представляет собой **классическую память произвольного доступа с совпадением токов** на сердечниках с четырьмя обмотками. В этом разделе будут в общих чертах рассмотрены некоторые варианты памяти на базе этой схемы.

Прежде всего отметим, что обмотка СЧИТЫВАНИЯ и обмотка ЗАПРЕТА могут быть объединены в одну обмотку. Усилитель считывания используется в течение только части временного цикла памяти, связанного с чтением, а драйвер ЗАПРЕТА включается только в течение отрезка временного цикла, связанного с записью. Просто объединив обмотку запрета с усилителем считывания и убрав обмотку считывания, можно построить матрицу всего с тремя обмотками на один сердечник. Чтобы представить это, предположим, что на рис. 7.33 удалена обмотка считывания. Тогда усилитель считывания должен быть подключен к обмотке запрета; такая обмотка называется **обмоткой разряда** или **обмоткой бита**. Конечно, усилитель считывания должен быть способен выдержать большой входной сигнал от драйвера запрета. Память с совпадением токов с тремя обмотками такого типа емкостью 16 четырехразрядных слов изображена на рис. 7.37.

Наряду с упрощением построения памяти сокращение числа обмоток позволяет уменьшить размеры сердечников. Сердечники меньших размеров могут переключаться быстрее больших сердечников, так что чем меньше обмоток, тем быстрее память. Но с сокращением числа обмоток усложняются схемы.

На рис. 7.38 показана память с линейной выборкой в 16 четырехразрядных слов.

Память 2,5D представляет собой еще один тип организации памяти на сердечниках. Один из вариантов памяти 2,5D применяется для больших недорогих ЗУ, а другой — для небольших быстродействующих ЗУ. Мы не будем детально рассматривать этот тип памяти, только заметим, что его основной принцип состоит в построении разрядных секций плат, разряды которых выбираются методом совпадения токов, но обмотки  $X$  проходят через все платы. Каждая плата имеет отдельную обмотку  $Y$ . Во время цикла ЧТЕНИЯ происходит совпадение тока выбранной обмотки  $X$ , который поступает на все платы, с током всех обмоток  $Y$ ; таким образом, возбуждается каждый выбранный драйвер  $Y$ . Во время цикла ЗАПИСИ включаются только драйверы  $Y$  тех плат, в которые должны быть записаны 1; драйверы  $X$  используются, как обычно. Фирма CDC применяет память с 12 млн. сердечников, имеющую цикл памяти 275 нс и организацию 2,5D. Организация в основном такая, как на рис. 7.11.

Память 2,5D может быть с тремя обмотками, причем с отдельной обмоткой считывания, либо она может состоять из нескольких ЗУ с двумя обмотками в сердечнике, в которых выход из выбранного сердечника считывается на линии драйвера  $Y$  с помощью специальной схемы. (Так устроена очень большая память IBM.) Память 2,5D организована таким образом, что

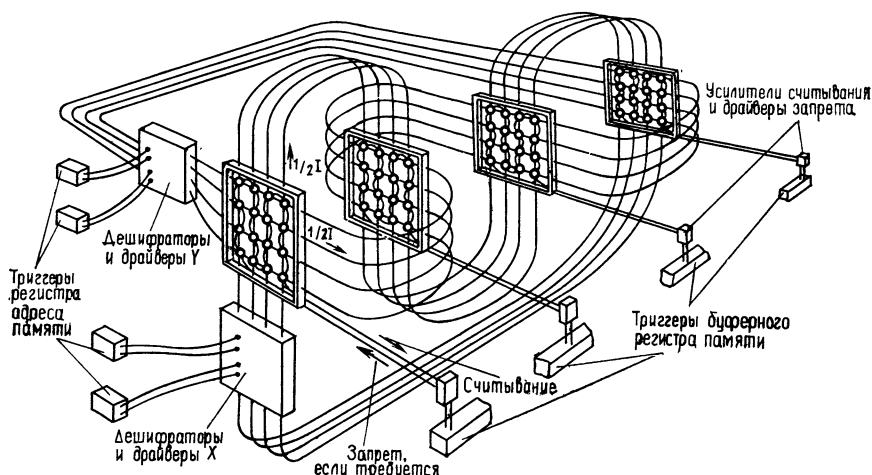


Рис. 7.37. Память с совпадением токов с тремя обмотками.

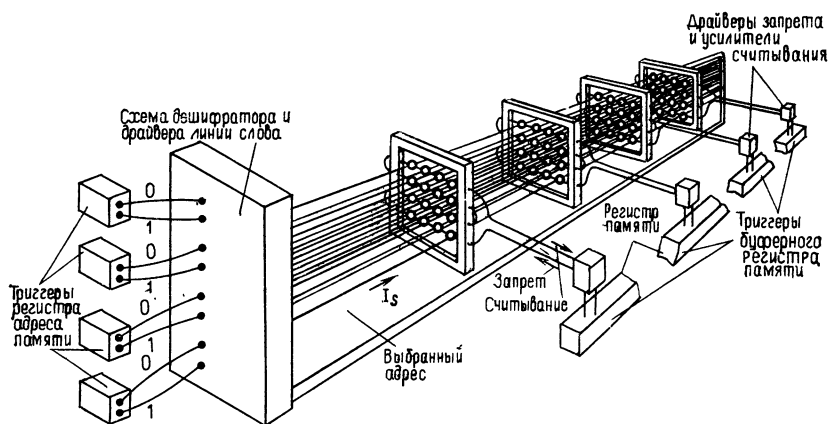


Рис. 7.38. Память с линейной выборкой.

для считывания используется метод совпадения полутоков, а для упрощения записи применяются отдельные обмотки  $Y$ , что усложняет пропускание сигналов этими обмотками  $Y$  по сравнению с обмотками запрета.

## 7.19. ПАМЯТЬ НА МАГНИТНОМ БАРАБАНЕ

В описанных ЗУ используется принцип установки устройства с двумя устойчивыми состояниями в одно из двух его состояний. Доступ к устройствам, по существу, произвольный — при



подаче адреса сердечник сразу становится доступным, а триггер, или динамическая ячейка, постоянно осуществляет индикацию своего состояния. Ограничения в использовании памяти такого типа связаны со сложностью, а следовательно, со стоимостью и надежностью хранения большого количества битов. В то время как ЗУ на ИС могут хранить десятки миллионов битов, в некоторых больших машинах требуется хранить до  $10^{13}$  бит. А это потребует 1 млн. ЗУ на ИС. За быстродействие памяти на ИС, следовательно, приходится платить усложнением памяти и повышением ее стоимости.

Магнитные барабаны были одними из первых недорогих средств хранения информации со сравнительно небольшим временем доступа. Магнитный барабан состоит в основном из вращающегося цилиндра, покрытого тонким слоем ферромагнитного материала с петлей гистерезиса, такой же, как у материала, используемого в магнитных сердечниках (рис. 7.30). Вдоль поверхности барабана устанавливается ряд головок записи. Они используются для записи информации на поверхность барабана и чтения с нее путем намагничивания небольших участков или считывания намагниченности участков, на которых была записана информация. На рис. 7.39, *а* показано только несколько головок; некоторые магнитные барабаны имеют несколько сотен головок, распределенных по их поверхности.

При вращении барабана небольшой участок его поверхности непрерывно проходит под каждой из головок. Этот участок называют **дорожкой** (рис. 7.39, *а*). Каждая дорожка делится на ячейки, а каждая ячейка может запомнить один двоичный бит.

Обычно одна из дорожек используется для синхронизации барабана. На окружности, образующей дорожку синхронизации, записывают непрерывную последовательность синхронизирующих сигналов, и каждый сигнал определяет **единицу времени** для системы с магнитным барабаном. Синхронизирующая дорожка затем используется для определения местоположения любой группы запоминающих ячеек на окружности выбранной дорожки. Например, если длина синхронизирующей дорожки 60 дюймов, а синхронизирующие импульсы записываются с плотностью 100 на дюйм, то на каждой из дорожек будет 6000 положений для битов (ячеек). Если на барабане 30 дорожек и одна синхронизирующая дорожка, то емкость памяти барабана составит 180 000 бит.

Для записи информации на барабан подается ток в обмотку головки записи. Это приводит к созданию магнитного потока в сердечнике головки. На рис. 7.39, *б* изображена головка для записи информации на барабан или считывания с него. В некоторых барабанных системах используются отдельные головки для записи и считывания, а в других — комбинированные головки

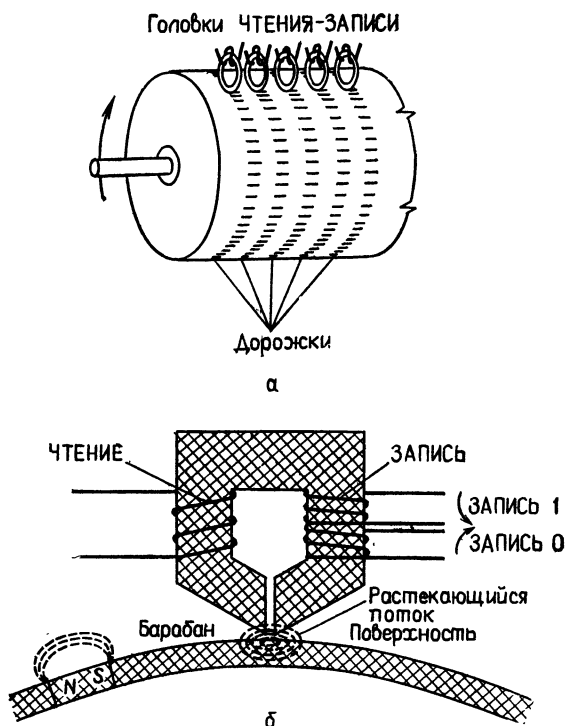


Рис. 7.39. Память на магнитном барабане.

а — дорожки магнитного барабана; б — головка чтения-записи.

считывания-записи. Головка на рис. 7.39, б состоит из обмотанного проводом кольца, изготовленного из материала с высокой магнитной проницаемостью. Когда нужно записать информацию на поверхности барабана, в обмотку записи 0 или 1 подаются импульсы тока. Направление потока через головку и в свою очередь полярность магнитного поля, записываемого на поверхности барабана, зависят от того, произведена запись 1 или 0.

Зазор в сердечнике представляет собой довольно высокое магнитное сопротивление для потока, генерируемого током обмотки головки. Поскольку магнитный материал поверхности барабана движется относительно головки вблизи зазора, большая часть потока проходит через этот магнитный материал, что вызывает намагничивание небольшого участка поверхности барабана; так как материал покрытия имеет большую остаточную намагниченность, то магнитное поле сохраняется и после того, как этот участок выйдет из-под головки, или подача тока в обмотку головки фактически прекратится. Следует заметить, для того чтобы предотвратить износ, головки располагают очень

Таблица 7.7. Характеристики систем памяти с магнитными барабанами

Характеристики	FN 432 UNIVAC 1108	2301 IBM 360	2303 IBM 360	1964 ICL Серия 1900	FASTRAND II UNIVAC 1108
Число дорожек на барабанах	144	220	800	512	6 144
Число символов на дорожке	12 288	20 483	4 892	4 048	10 752
Число символов в ус- тройстве	1 572 864	4 096 600	3 910 000	2 072 576	132 120 756
Скорость передачи, Кбит	1 440	1 200	312	100	153
Тип головки	Фиксированная	Фиксированная	Фиксированная	Фиксированная	Подвижная
Число головок в барабанах	128	200	800	512	32×192 позиций
Диаметр барабана, дюйм	10,5	10,7		18,5	32,8
Скорость барабана, об/мин	7 100	3 490	3 428	1 500	870
Продольная плотность записи, бит/дюйм	627	1 250	1 105	1 000	1 000
Среднее время доступа, мс	4,25	8,6	8,75	20,5	93
Примерная стоимость за- поминания 6-разряд- ного символа, цент	8	5	4	3	0,2

близко к поверхности барабана, но так, чтобы они не касались ее. Барабан, следовательно, должен иметь очень точный постоянный диаметр, чтобы расстояние головки от поверхности барабана было постоянным. На практике головки могут слегка перемещаться по отношению к поверхности барабана, а пружинный механизм подталкивает их к поверхности. Воздушная подушка между головкой и вращающейся поверхностью поддерживает требуемое относительно постоянное расстояние от магнитной поверхности.

Сигналы, записываемые на поверхности барабана, считываются аналогичным образом. Когда намагниченные участки проходят под головкой, часть магнитного потока взаимодействует с головкой и изменения этого потока индуцируют сигналы в обмотке. Затем эти сигналы усиливаются и интерпретируются. Описание техники записи приведено в последнем разделе этой главы.

Размеры и емкость памяти магнитных барабанов весьма разнообразны. Есть небольшие барабаны емкостью меньше 200 000 бит. Барабаны такого размера обычно имеют 15—25 дорожек и 15—50 головок. Чтобы уменьшить время доступа, головки иногда размещают группами по окружностям барабана, так что у барабана с 15 дорожками может быть 30 головок, разделенных на две группы по 15 головок, причем группы отстоят друг от друга на  $180^\circ$ . Для достижения очень малого времени доступа количество групп головок может быть увеличено.

Очень большие барабаны могут хранить до  $10^9$  бит при наличии от 500 до 1000 дорожек. Большие барабаны обычно вращаются намного медленнее малых; скорости изменяются от 120 до 75 000 об/мин. Очевидно, что с увеличением скорости барабана время доступа уменьшается. Однако еще одним важным фактором является плотность упаковки вдоль дорожки — продольная плотность. Большинство современных барабанов имеют продольную плотность от 600 до 2000 бит/дюйм. (Удерживая головки очень близко к поверхности барабана и медленно вращая барабан, можно достичь значений продольной плотности свыше 1000 бит/дюйм.) Запоминающее устройство на магнитном барабане IBM 2303 имеет 800 дорожек и 4892 байта на каждой дорожке (и, таким образом, емкость около 3,9 млн. байтов), а барабан совершает полный оборот за 17,5 мс. В табл. 7.7 приведены некоторые характеристики современных систем памяти с магнитными барабанами.

## **7.20. МАГНИТНЫЕ БАРАБАНЫ ПАРАЛЛЕЛЬНОГО И ПОСЛЕДОВАТЕЛЬНОГО ДЕЙСТВИЯ**

Барабан может работать либо в параллельном, либо в последовательном режиме. При параллельной работе все биты

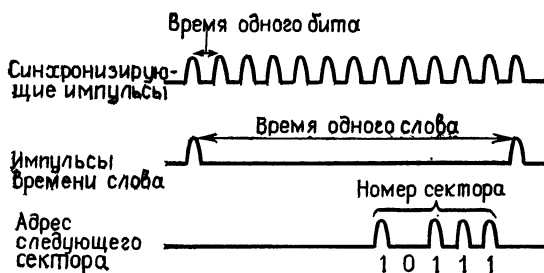


Рис. 7.40. Синхронизирующие сигналы магнитного барабана.

слова могут быть записаны и считаны одновременно. Если машинное слово содержит 40 бит, с барабана можно одновременно считывать 41 дорожку (одна дорожка синхронизирующая); таким образом, все машинное слово будет считываться за время, требуемое для 1 бита. Когда считывание с барабана или запись в него выполняется в параллельном режиме, требуется отдельный усилитель чтения и записи для каждой используемой дорожки, так что для считывания 40-разрядного слова за время чтения 1 бит потребуется 40 усилителей считывания. Затем выбирается подходящий набор головок, и барабанная система определяет местонахождение выбранного набора ячеек.

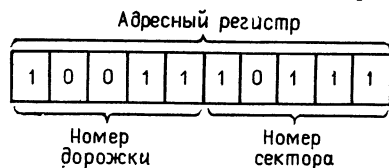
Заметим, что положение слов в параллельной системе можно установить, используя синхронизирующую дорожку. Если каждая дорожка содержит 8192 бит, 13-разрядный счетчик после полного оборота барабана может устанавливаться в нуль в одной и той же позиции и продвигаться на единицу при появлении очередного синхронизирующего импульса. Таким образом, ячейка с адресом 1096 будет 1096-м участком на окружности дорожки, считая от 0-й ячейки. Если адрес считываемого слова загрузить в регистр, сигналы с барабана могут быть переданы в ЭВМ при совпадении содержимого счетчика с содержимым регистра. Таким же образом могут быть помещены слова на поверхность барабана при записи.

Магнитный барабан может также работать в последовательном режиме. В этом случае в любое заданное время можно считывать или записывать только на одной дорожке. Так как у каждого барабана много дорожек, то должна быть выбрана необходимая дорожка, а также определено местоположение требуемых битов на окружности этой дорожки.

Поэтому каждой дорожке присваивается номер. Кроме того, каждая дорожка разделяется на секторы, а каждый сектор содержит одно или более полных машинных слов.

Чтобы указать адрес слова на последовательно работающем магнитном барабане, нужно прежде всего задать номер до-

рожки и номер сектора. Если в секторе всего одно слово, этого будет достаточно; если в каждом секторе больше одного слова, следует указать еще и адрес этого слова. Рассмотрим барабан с 32 дорожками плюс синхронизирующая дорожка и 32 словами (секторами) на каждой дорожке. Адрес слова на барабане в двоичной машине будет состоять из 10 разрядов: 5 разрядов — для спецификации дорожки и 5 — для сектора.



Пять триггеров, содержащих номер дорожки, можно подключить к дешифратору, который затем выберет необходимую головку чтения-записи.

Местонахождение выбранного сектора можно определить несколькими способами с помощью синхронизирующих дорожек. Один из способов основан на использовании нескольких синхронизирующих дорожек вместо одной. На рис. 7.40 иллюстрируется метод с применением трех синхронизирующих дорожек. На одной из дорожек записана последовательность сигналов, указывающих местоположение каждого бита на окружности дорожек. Вторая дорожка содержит группу импульсов, причем каждый импульс расположен в начале времени слова <sup>1)</sup>. (Показанные временные сигналы слова отстоят друг от друга на 12 разрядов, так что базовое слово должно иметь длину 11 или 12 бит.) На третьей синхронизирующей дорожке записывается номер сектора следующего слова на окружности барабана. ЭВМ читает номера секторов с этой дорожки, а когда считанный номер совпадает с номером сектора в адресе, ЭВМ может прочитать выбранное слово из следующего сектора, начинающегося следующим импульсом времени слова.

## 7.21. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА НА МАГНИТНЫХ ДИСКАХ

Другой тип памяти, называемый **памятью на магнитном диске**, очень напоминает по действию память на магнитном барабане. ЗУ на магнитном диске обеспечивает большую емкость памяти при умеренных рабочих скоростях. В настоящее время имеется довольно большое число различных типов ЗУ на маг-

<sup>1)</sup> Сектор окружности дорожки, содержащий слово. Время, необходимое для прохождения этого сектора под головкой считывания-записи, образует малый цикл. — *Прим. ред.*

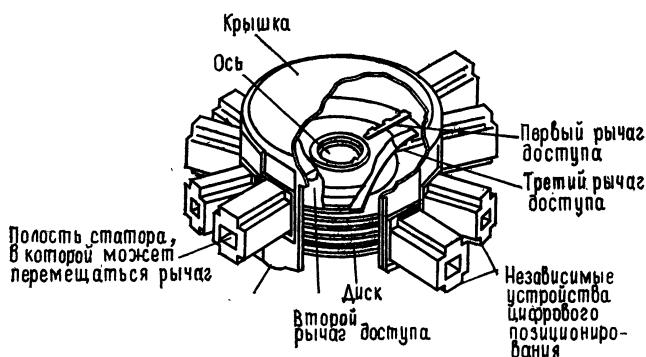
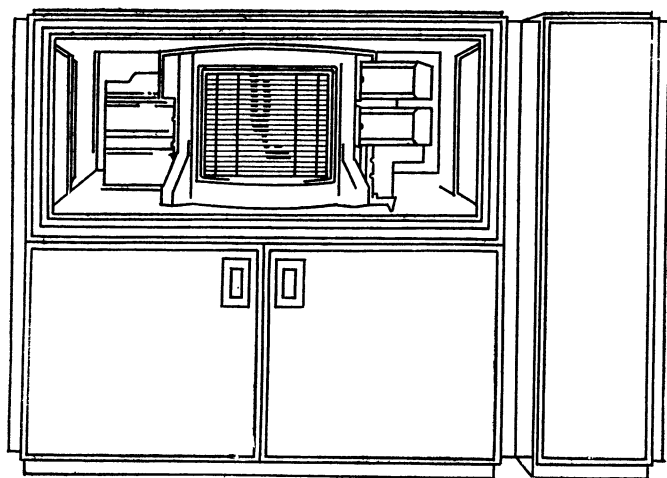


Рис. 7.41. Запоминающая система на магнитном диске.

нитных дисках. Отличаясь в отдельных деталях, все они основаны на одних и тех же принципах действия.

Накопитель на магнитном диске напоминает автоматический проигрыватель. Вращающиеся диски, покрытые магнитным материалом, собраны в пакет; диски отделены друг от друга небольшими промежутками (рис. 7.41). Информация записывается на поверхности вращающихся дисков с помощью магнитных головок, которые радиально перемещаются по дискам. (Информация записывается по концентрическим окружностям, а не по спирали.) Каждая полоса информации в виде окружности на диске называется **дорожкой**. На одной стороне типичного диска может быть от ста до нескольких тысяч таких информационных дорожек. Биты записываются на дорожке с плотностью пример-

но от 500 до 9000 бит/дюйм. В некоторых системах наружные дорожки содержат больше битов, чем внутренние, так как длина окружности наружной дорожки больше, чем внутренней, но большинство дисков имеют одинаковое число битов на каждой дорожке. Скорость вращения дисков, конечно, различна у разных изготовителей, но типичные скорости порядка 3600 об/мин.

Так как каждый диск содержит большое число дорожек, а в заданной памяти может быть несколько дисков, разработано несколько методов установки головки чтения-записи в правильное положение на выбранной дорожке. Поскольку для чтения и записи обычно используется одна и та же головка, возникает проблема правильной и быстрой установки этой головки на выбранной дорожке.

Существуют два основных типа систем установки дисковых головок. В системе первого типа головки находятся в фиксированном положении на каждой дорожке. Такие системы называются **системами с фиксированными головками**. У систем второго типа имеется одна или более пар головок чтения-записи для каждой пары соседних поверхностей диска (так как информация обычно записывается с двух сторон каждого диска). Такие головки чтения-записи установлены на рычагах, которые могут двигаться и выдвигаться. Эти системы называются **системами с перемещающимися головками**.

Позиционирование<sup>1)</sup> головок путем механического перемещения рычагов представляет сложную и искусную работу, особенно из-за того, что дорожки на диске часто записываются на расстоянии менее сотых долей дюйма одна от другой. Очевидно, что накопители на дисках со многими головками могут находить и записывать или читать с выбранной дорожки быстрее, чем при малом числе головок, так как общее число механических перемещений при поиске дорожки для систем с большим количеством головок будет меньше.

Общее время, требуемое для того, чтобы начать считывание выбранных данных или начать запись данных на определенном месте выбранной дорожки, называется **временем доступа**.

Время, требуемое для позиционирования головки на выбранной дорожке, называется **временем поиска**; обычно оно принимает значения от нескольких миллисекунд до долей секунды. Другой задержкой при нахождении выбранных данных является **время ожидания**, или **задержка вращения**, связанная с временем, которое требуется данным для того, чтобы достигнуть головки, когда головка уже установлена. Таким образом, общее время доступа для диска есть время поиска плюс время ожидания.

---

<sup>1)</sup> Термин «позиционирование» означает процесс установки в определенное местоположение. — *Прим. ред.*



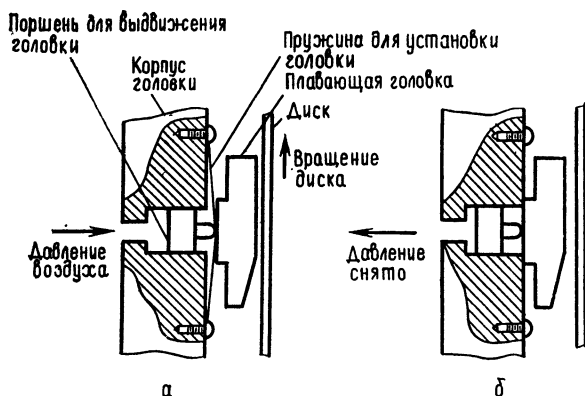


Рис. 7.42. Плавающая головка для запоминающего устройства на магнитном диске.

*а* — головка показана в рабочем положении; *б* — головка показана в отведенном положении.

При скорости вращения 2400 об/мин, например, максимальное значение времени задержки составляет 25 мс, а среднее значение равно 12,5 мс. Время ожидания представляет собой нижнюю границу для времени доступа в системах с фиксированными головками. По этой причине для получения минимального времени доступа используют фиксированные головки. Обычно головки объединены в группы по восемь или девять, включая, возможно, и запасную, и тщательно выравнены в фиксированных положениях относительно диска. Как правило, в группе на одном дюйме можно разместить 8—16 головок, однако за счет чередования групп можно достигнуть плотностей 30—60 дорожек на дюйм.

При равных площадях для записи на дисках системы с фиксированными головками имеют меньшую емкость памяти, чем системы с перемещающимися головками, так как в системах с перемещающимися головками число дорожек на один дюйм больше.

Важным преимуществом перемещающихся магнитных головок является возможность установки их на очень близко расположенные дорожки данных. Хотя расстояние между дорожками ограничивается взаимными помехами между соседними дорожками, а также механическими допусками, обычным является расстояние между соседними дорожками в 0,01 дюйма. Ширины дорожек порядка  $(2,5 - 5) \times 10^{-3}$  дюйма соответствует точность позиционирования головок порядка  $(0,5 - 1,5) \times 10^{-3}$  дюйма.

В накопителях на магнитных дисках, как правило, используются головки чтения-записи типа **плавающих головок**. Упрощенно

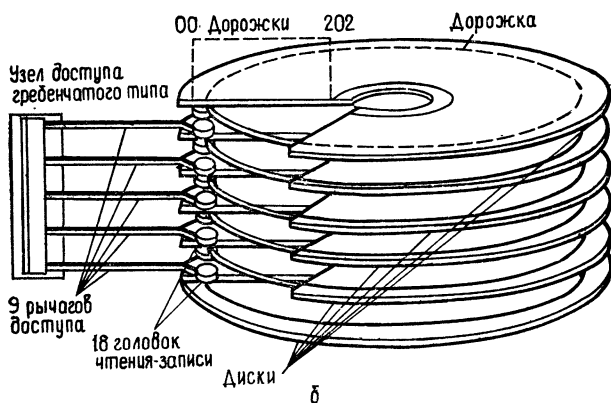
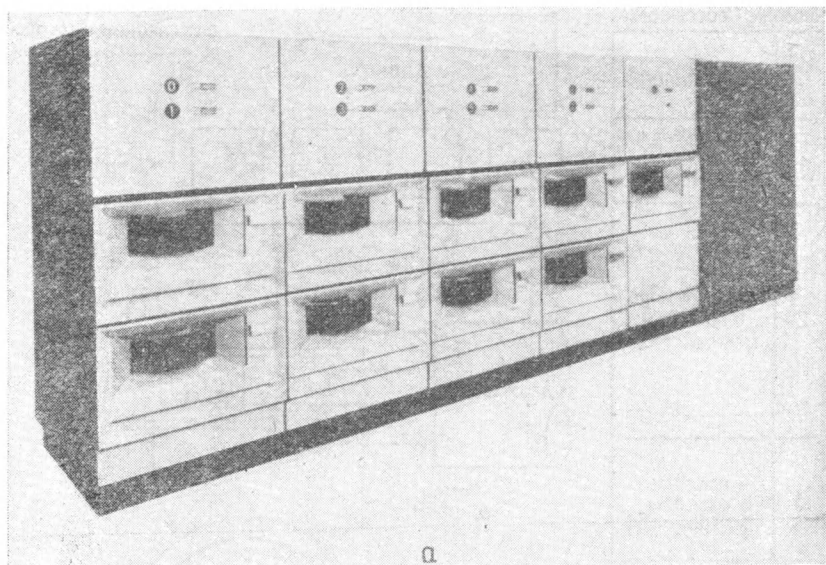


Рис. 7.43. а — память на магнитных дисках IBM 2314; б — пакет дисков для IBM 2314 (показаны только пять дисков).

щенный чертеж плавающей головки приведен на рис. 7.42. Когда диск вращается с большой скоростью, тонкий, но упругий пограничный слой воздуха вращается вместе с ним. Головка имеет такую форму, что она плавает в этом слое вращающегося воздуха, благодаря которому диск не соприкасается с головкой, и, таким образом, предотвращается износ поверхности диска. По существу слой воздуха, вращающийся вместе

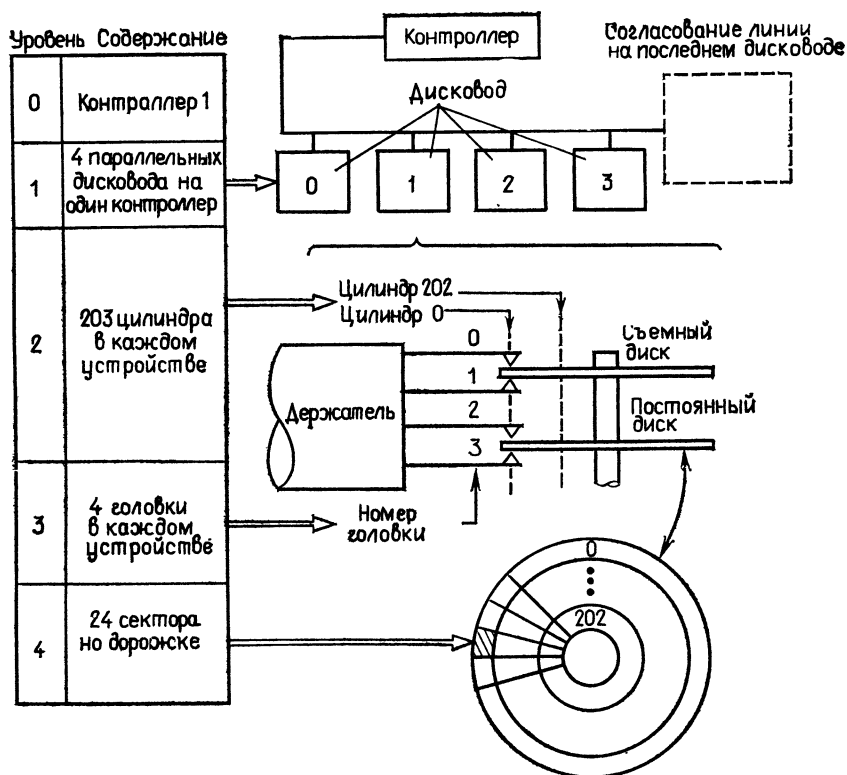


Рис. 7.44. Запоминающая система, содержащая до четырех дисководов, у каждого из которых по два диска с двумя сторонами для записи. (С разрешения фирмы Hewlett-Packard.)

*Примечание.* Дорожка выбирается, когда выбраны цилиндр и головка.

с диском, действует как пружина с жесткостью, превышающей несколько тысяч фунтов на один дюйм, удерживая головку от соприкосновения с поверхностью диска. Чтобы привести головку в необходимую близость с диском, используют различные устройства. Часто применяют приспособление типа «поршня» (рис. 7.42), нагнетая сжатый воздух в механизм, подающий головку к поверхности диска.

Дисковые ЗУ имеют разные размеры и скорости. Некоторые диски довольно большие, их диаметр достигает 4 футов. Другие, меньших размеров, вращаются быстрее, являются сменными и т. д. Благодаря тому что существует большое количество таких ЗУ, а при их изготовлении может быть обеспечено практически бесконечное разнообразие конфигураций, системному пользователю предоставляется значительная свобода выбора,

Таблица 7.8. Пакеты дисков

Характеристики дисковых механизмов	IBM 2314	CDC 215	UNIVAC 8440	IBM 3330	IBM 3330-11	IBM 3340
Тип конструкции пакета	IBM 2316	IBM 2316	2316 <sup>а)</sup>	IBM 3336	IBM 3336-11	IBM 3348
Число поверхностей	20	20	20	19	19	24
Емкость памяти, Мбайт	29	54	108	100	200	35—70
Число дорожек на дюйм	200	200	200	192	384	300
Число перемещений на дюйм	4400	4400	4400	4040	4040	5072
Скорость данных, млн. бит/с	2,5	2,5	5,0	6,5	6,5	7,1
Среднее время доступа, мс	60	30	30	30	30	25
об/мин	2400	2400	2400	3600	3600	3000
Фиксатор	Механический <sup>б)</sup>	Оптический	Оптический	Серво	Серво	Серво
Привод	Гидравлический <sup>б)</sup>	Звуковая катушка	Звуковая катушка	Звуковая катушка	Звуковая катушка	Звуковая катушка
Покрытие диска, минимальная толщина, мкдьюм	90	90	50	50	40	40
Зазор между плавающей головкой и поверхностью диска, минимальное значение, мкдьюм	80	80	70	50	30	30

<sup>а)</sup> Специальный пакет, выпущенный с аппаратурой 2316 и дисками 3336.

<sup>б)</sup> Гидридные устройства, выпускаемые не фирмой IBM; имеют звуковую головку и оптическую маску.

Очень популярны дисковые ЗУ со сменными пакетами дисков (или **модулями**). Каждый пакет содержит набор совместно вращающихся дисков. В дисковом устройстве IBM 2314, изображенном на рис. 7.43, а, применяется, например, сменный пакет из 10 дисков с 20 рабочими поверхностями (рис. 7.43, б). В этой конфигурации не используются верхняя поверхность верхнего диска и нижняя поверхность нижнего диска. Используется восемнадцать головок чтения-записи, по две для каждой поверхности, на которой записана информация. Диски вращаются со скоростью 2400 об/мин. Головки крепятся в каретке механизма позиционирования, похожей на гребенку. Поверхность каждого диска содержит 200 концентрических магнитных дорожек (треков), а каждая дорожка имеет примерно 4400 бит на дюйм.

На рис. 7.44 приведены характеристики недорогой дисковой системы, выпускаемой фирмой Hewlett-Packard.

В табл. 7.8 перечислены характеристики ряда больших дисковых систем.

В середине 1970-х годов дисковые устройства со сменными пакетами, или модулями, использовались наиболее широко. Однако примерно в это время появилось устройство IBM 3350 с фиксированными дисками, с большей плотностью записи, большим числом дорожек на поверхности диска и с более высокой скоростью передачи, созданное на основе новой дисковой технологии. Эта дисковая технология известна под названием **Винчестер**-технологии; она отличается низкой нагрузкой на головку (10 г по сравнению с 300 г в более ранних устройствах) и малой массой головки. Кроме того, поскольку диски не являются сменными, упрощаются проблемы, связанные с выравниванием и установкой головок на дорожку. На диски нанесено гладкое тонкопленочное покрытие, так что легкие головки могут

Таблица 7.9. Характеристики больших дисководов с фиксированными дисками (для CDC 9776 <sup>а)</sup>)

Характеристика	Спецификация
Число шпинделей в корпусе	2
Емкость на один шпиндель	400 Мбайт (подвижная головка) 1,72 Мбайт (фиксированная головка)
Скорость данных	9,6 МГц
Среднее время доступа	25 мс
Скорость вращения	3600 об/мин
Время ожидания	8,4 мс

<sup>а)</sup> С разрешения фирмы CDC.

*Таблица 7.10.* Характеристики дисководов типа Винчестер  
(для малых и средних систем)

Характеристика	Спецификация
Число дисков	1—4
Число поверхностей данных	1—7
Плотность битов	6400 бит/дюйм
Плотность дорожек	500 дорожек/дюйм
Число дорожек на поверхности	690
Емкость поверхности	7,6 Мбайт
Скорость вращения	3600 об/мин
Емкость четырех дисков	53,2 Мбайт

«ударяться», не нанося повреждений (они отскакивают). Первые дисководы «Винчестер» применялись для больших запоминающих систем; однако вскоре этой технологией воспользовались изготовители малых машин, так что теперь дисководы типа Винчестер производят многие фирмы и системы с фиксированными дисками используются как в больших системах, так и в системах мини- и микроЭВМ.

В табл. 7.9 приведены характеристики для больших дисководов CDC, а в табл. 7.10 даны характеристики для типичных малых дисководов типа Винчестер. Благодаря относительно невысокой стоимости хранения одного бита информации в дисковых ЗУ, относительно малым временам доступа и большим скоростям передачи, достигаемым при считывании или записи данных в файле на диске, накопители на магнитных дисках стали одними из важнейших запоминающих устройств в современных цифровых ЭВМ.

## **7.22. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА НА ГИБКИХ ДИСКАХ (ФЛОППИ-ДИСКАХ)**

В памяти на дисках нового типа, разработанной в IBM, вместо обычного жесткого диска на металлической основе используется гибкий «флоппи»-диск на пластиковой основе. Среда запоминания имеет примерно такие же размер и форму, как грампластинка на 45 об/мин, и ее можно вставлять так же просто, как катушку с пленкой. Каждый гибкий диск стоит всего несколько долларов.

Гибкие диски сменные, и каждый диск находится в конверте, показанном на рис. 7.45. Диски монтируются на дисковом вместе с конвертом; информация записывается и считывается через отверстие в конверте. (Существуют системы, где требуется снимать конверт.) Некоторые изготовители обеспечивают сейчас

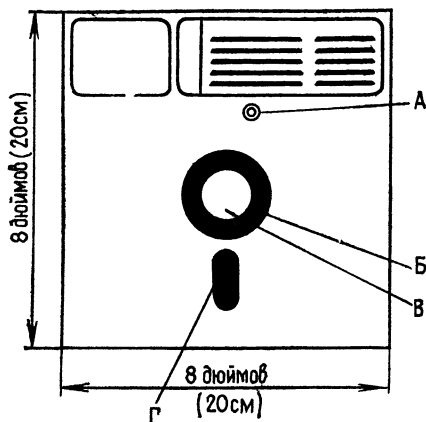


Рис. 7.45. Гибкий диск с конвертом.

А) Индексное отверстие. Внешняя окружность — это отверстие в конверте; внутренняя окружность — это индексное отверстие в диске. Когда эти два отверстия совмещаются, при вращении диска во время операций обработки данных луч света, освещающий одну из сторон дискетки, улавливается с другой стороны и используется для синхронизации.

Б) Отверстие в конверте для механизма привода.

В) Отверстие в диске для вала дискетода. После того как дискетка установлена в машине и вал дискетода введен в отверстие диска, механизм привода фиксируется на той части диска, которая открыта отверстием в конверте для механизма привода.

Г) Щелевое отверстие площадки контактного давления (зона считывания-записи). (Подобное отверстие на другой стороне дискетки называется *щелью головки*.) Щель головки открывает поверхность записи диска при поворотах диска в своем конверте. Устройство для записи и считывания данных в дисковом устройстве, называемое *головкой чтения-записи* и аналогичное головке записи-воспроизведения в магнитофоне, перемещаясь вдоль щели, устанавливается в указанные позиции. Это передвижение в заданную позицию называется *доступом к дорожке*. (Данные записываются только на той стороне дискетки, в которой имеется щель головки.)

такими дисками целые системы стоимостью ниже 500 долл. Удобство использования и невысокая стоимость гибких дисков способствуют широкому применению их в самых разных системах.

В большинстве дискетодов с гибкими дисками головки чтения-записи находятся в непосредственном контакте с поверхностью диска. (Для продления времени жизни контакт головки с дискеткой обычно имеет место только на время чтения или записи.) Время жизни дорожки на дискетке обычно порядка 3—5 млн. контактных оборотов. Стандартный гибкий диск заключен в 8-дюймовый квадратный пакет, а сам диск имеет диаметр 7,88 дюйма. Записывающая поверхность представляет собой слой ферромагнитного покрытия толщиной 100 микро-дюймов на полистироловой подложке толщиной 0,003 дюйма. Конверт защищает диск от прикосновений; кроме того, он имеет специальную прокладку, которая протирает диск, снимая с него продукты износа и другую грязь, которая может повредить носитель и головку.

В первоначальном варианте гибкого диска фирмы IBM для индикации начала дорожки использовалось индексное отверстие диаметром 0,100 дюйма<sup>1)</sup> на расстоянии 1,5 дюйма от

<sup>1)</sup> Индексное отверстие указывает начало первого сектора на гибком диске. — Прим. ред.

Таблица 7.11. Характеристики устройства с гибкими дисками IBM

Характеристика	Спецификация
Емкость, байт <sup>а)</sup>	400 000
Скорость вращения, об/мин	360
Скорость передачи, бит/с	250 000
Время доступа при переходе от дорожки к дорожке, мс	16—20
Среднее время доступа, мс	176
Плотность записи	
Внутренняя дорожка, бит/дюйм	3 268
Внешняя дорожка, бит/дюйм	1 836
Число дорожек на дюйм	48
Число дорожек	77

<sup>а)</sup> Это максимальное значение; используя стандартный формат системы записи IBM, на диске записывают только 250 000 байт данных.

центра диска. Ширина дорожки 0,012 дюйма, а стандартная плотность дорожек 48 на дюйм. Диск имеет 77 дорожек. Емкость поверхности при использовании стандартного кода и плотности расположения информации 3268 бит на 1 дюйм составляет около 400 000 байтов по 8 бит. В табл. 7.11 приведены некоторые характеристики дискового устройства с гибкими дисками. Диски выпускаются также размером 5<sup>1</sup>/<sub>4</sub> дюйма, или 3<sup>1</sup>/<sub>2</sub> дюйма, и популярность таких дисков все возрастает. Ряд

Таблица 7.12. Характеристики дисковых систем

	8-дюйм. дисковый механизм		5,25-дюйм. дисковый механизм	
	односторонний	двусторонний	односторонний	двусторонний
Емкость (формат нестандартный)				
Одинарная плотность, Кбайт	400	800	110	220
Удвоенная плотность, Кбайт	800	1600	220	440
Среднее время доступа, мс	225	100	450	300
Скорости передачи				
Одинарная плотность, Кбайт/с	250	250	125	125
Удвоенная плотность, Кбит/с	500	500	250	250
Число дорожек	77	154	35	70
Скорость вращения, об/мин	360	360	300	300
Число дорожек на дюйм	48	48	48	48



фирм предлагает системы с гибкими дисками, в которых используется принцип плавающих головок. В некоторых флоппи-дисках применяется система адресации (как в обычных дисках), при которой задаются дисковод, дорожка и сектор. В других системах каждый блок записываемых данных на дорожке содержит заголовок, и информация заголовка указывается при каждом обращении.

В системе IBM standard используется одна полная дорожка для информации о формате. На остальных дорожках биты начала и заголовков, а также биты проверки чередуются с данными.

Чтобы увеличить емкость диска, можно использовать дисководы с двусторонней удвоенной плотностью дорожек (100 дорожек на дюйм) и удвоенной плотностью информации (5—6 Кбит на дюйм). Чем дешевле дисковые механизмы, тем меньше дорожек они обычно имеют.

В табл. 7.12 приведены некоторые характеристики малых дисковых систем.

### 7.23. МАГНИТНАЯ ЛЕНТА

В настоящее время наиболее распространенной средой для хранения больших объемов информации является магнитная лента. Из-за большого времени доступа магнитной ленты нежелательно использовать ее в качестве быстродействующей основной памяти, однако современные методы серийного производства настолько понизили стоимость ленты, что запоминание огромного количества информации может обойтись недорого. Более того, так как информацию на ленте можно стирать и перезаписывать, одна и та же лента может использоваться многократно. Другое преимущество магнитной ленты заключается в том, что хранящаяся информация не исчезает, и, следовательно, данные или программы, записанные в одном месяце, могут быть вновь использованы в следующем.

Еще одно преимущество использования магнитной ленты для запоминания больших объемов данных вытекает из того факта, что можно заменять бобины лент на лентопротяжном механизме. Таким образом, один и тот же механизм управления магнитной лентой и связанная с ним схема могут быть использованы с множеством разных бобин ленты с различными данными.

Запоминающее устройство, использующее магнитную ленту, состоит из четырех основных частей:

1. **Магнитная лента.** Это обычно гибкая пластмассовая лента с тонким слоем покрытия из какого-нибудь ферромагнитного материала.

**2. Лентопротяжный механизм.** Он состоит из устройства, предназначенного для перемещения ленты относительно записывающих головок по команде ЭВМ. Сюда же относятся головки и средства для хранения ленты, такие, как бобины, на которые намотана лента.

**3. Система чтения и записи.** Эта часть системы включает усилители чтения и записи и преобразователи, которые преобразуют сигналы с ленты в цифровые сигналы, пригодные для использования в центральной вычислительной системе.

**4. Оборудование для переключения и буферизации.** Эта часть состоит из устройств, требуемых для выбора необходимого лентопротяжного механизма, если их несколько, для запоминания информации, поступающей с ленты, и информации, которая должна быть считана на ленту (обеспечивает буферизацию), а также для проведения таких операций, как ручная перемотка ленты.

Лентопротяжные механизмы, используемые в цифровых системах, обладают способностью очень быстро разгоняться, останавливаться и обеспечивать высокую скорость движения ленты. Способность очень быстро разгонять и останавливать ленту важна по двум причинам. Во-первых, ввиду того что процесс записи или чтения не может начаться, пока лента не будет двигаться с достаточной скоростью, задержка, связанная с нарастанием скорости, замедляет операцию. Во-вторых, информацию обычно записывают на магнитной ленте «блоками» или «записями». Так как лента может быть остановлена между блоками или записями, та часть ленты, которая проходит во время процесса остановки или разгона под головками, пропадает впустую. Это так называемый **межблочный промежуток**, или **промежуток между записями**. Быстрые разгон и остановка сохраняют ленту.

На рис. 7.46,а показан типичный лентопротяжный механизм. Для того чтобы можно было очень быстро разогнать и затормозить ленту, стремятся изолировать бобины ленты, обладающие большой инерцией, от механизма, который перемещает ленту относительно записывающих головок. На рис. 7.46,б показан быстродействующий стартстопный лентопротяжный механизм, использующий несколько натяжных рычагов, вокруг которых натянута лента. Верхний и нижний рычаги натяжения на рис. 7.46,б подвижные, и, когда ведущий вал начинает передвигать ленту относительно головок, механизм обеспечивает буферизацию подачи ленты. Для управления верхней и нижней бобинами используется сервомеханизм; он поддерживает длину участка ленты между ведущим валом и бобинами ленты достаточной для того, чтобы обеспечивать постоянную подачу ленты

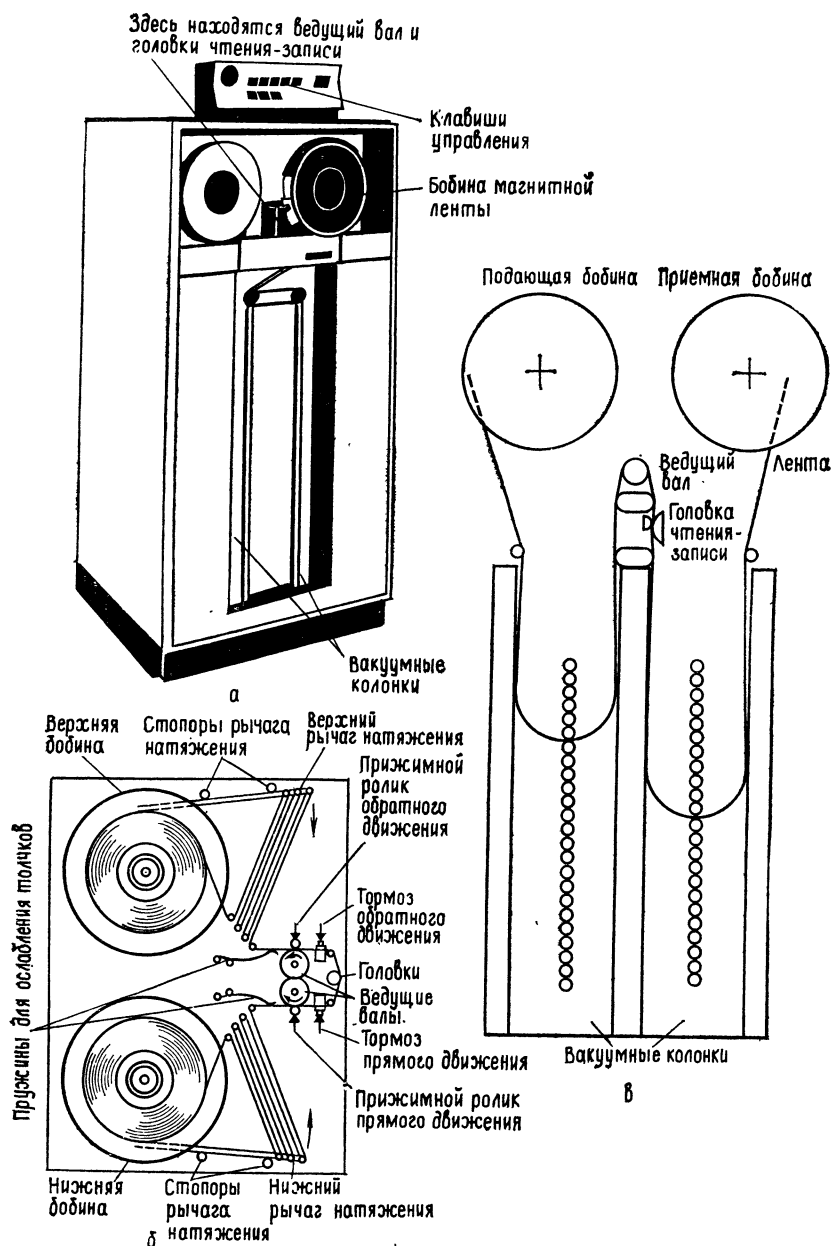


Рис. 7.46 а — лентопротяжный механизм IBM 3420; б — лентопротяжный механизм с вакуумными колонками.

Таблица 7.13. Характеристики типичного лентопротяжного устройства с рычагами натяжения

Размер бобины, дюйм	бит/дюйм	Скорость данных, Кбит/с	Время разгона-останова, мс	Емкость, Мбайт	Стоимость типичной бобины ленты, долл.
7	1600	40	15	11,5	7,50
8,5	1600	60	10	23	8,50
10,5	1600	72 Кбит/с при 45 дюйм/с; 120 Кбит/с при 75 дюйм/с	8,33 (5 мс при 75 дюйм/с)	46	11,00

на рычаги натяжения. В табл. 7.13 приведены некоторые характеристики системы такого типа.

Другое приспособление для изолирования бобин ленты, обладающих большой инерцией, от основного привода ленты показано на рис. 7.46, в. Такая система изолирует ленту от привода лентопротяжного механизма с помощью двух колонок, в которых ленты удерживаются на месте вакуумом. Сервомеханизм поддерживает нужную длину ленты между бобиной и ведущим валом. В обеих системах — этой и предыдущей — используются постоянно вращающиеся ведущие валы для продвижения ленты, а также «прижимные ролики» для того, чтобы прижимать ленту к ведущему валу, когда активизируется транспортировка ленты. Для быстрой остановки предусмотрены тормоза. В табл. 7.14 приведены некоторые типичные количественные характеристики системы такого рода.

При использовании подобных систем время запуска и время остановки может быть меньше 5 мс. Такое время требуется для того, чтобы разогнать ленту до скорости, необходимой для чтения или записи, либо для полной остановки движущейся ленты. Скорости, с которыми ленты движутся относительно головок, весьма разнообразны; для большинства лентопротяжных механизмов они имеют значения в пределах от 12,5 до

Таблица 7.14. Характеристики типичного лентопротяжного механизма с вакуумными колонками для бобин диаметром 10,5 дюйма

Скорость, дюйм/с	бит/дюйм	Максимальная скорость передачи данных, Кбайт/с	Время разгона-останова, мс	Емкость, Мбайт
50	1600	80	7,5	46
75	1600	120	5	46
200	1600	320	3	62
200	6250	1250	1,2	350

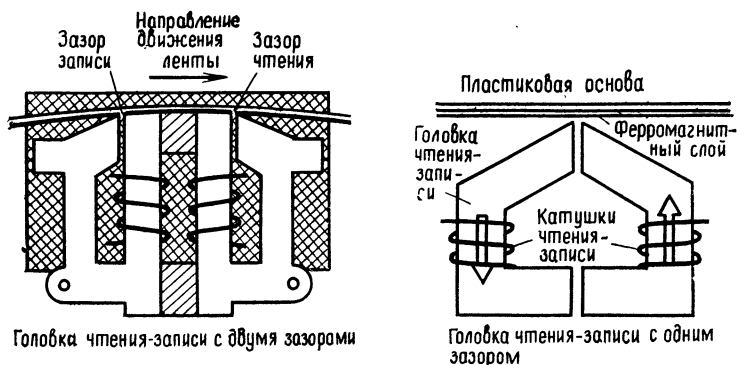


Рис. 7.47. Головки чтения-записи с одним (а) и двумя (б) зазорами.

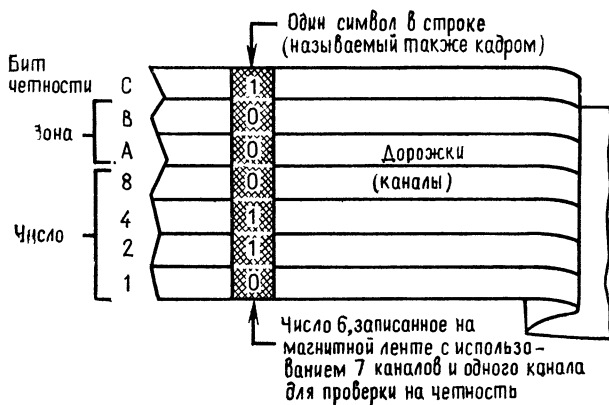


Рис. 7.48. Базовая схема магнитной ленты.

250 дюймов/с. В Лаборатории Линкольна сконструирован очень быстрый (900 дюймов/с) механизм, в котором лента управляется непосредственно бобинами. Система такого вида предназначена только для передачи больших объемов данных в основную быстродействующую память машины и из нее за одну операцию. Для систем, в которых должны передаваться меньшие количества данных, очевидно, более приемлемы меньшие скорости ленты в сочетании с быстрыми разгоном и остановкой. Некоторые системы имеют сменные кассеты с катушкой ленты в каждой кассете. Считается, что это предохраняет ленту и облегчает смену катушек. Эти системы будут рассмотрены в следующей главе.

Большинство ленточных систем имеют головки чтения-записи с двумя зазорами. Два зазора (рис. 7.47) удобны, так как во время записи зазор чтения устанавливается вслед за зазо-

ром записи и используется для проверки путем чтения и сравнения того, что было записано.

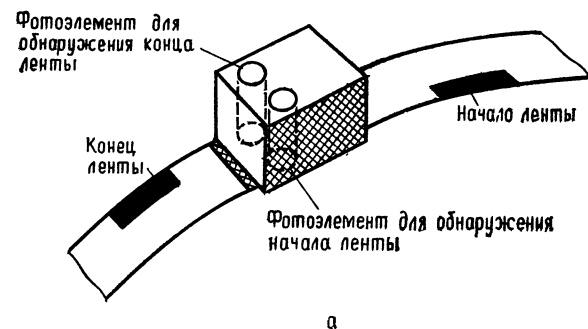
По ширине ленты варьируются от 1/4 до 3 дюймов; однако большинство лент — это ленты из майлара шириной 1/2 дюйма и толщиной  $1,5 \times 10^{-3}$  дюйма. В 10,5-дюймовой бобине обычно содержится 2400 или 3600 футов ленты. Обычно при ширине ленты в 1/2 дюйма используют около девяти каналов или дорожек. Поверхность ленты, как правило, находится в контакте с головкой чтения-записи. Выходные сигналы головок чтения лежат в основном в пределах от 0,1 до 0,5 В. Плотность записи может быть различной; однако до некоторой степени стандартными являются плотности 200, 556, 800, 1600, 6250 и даже 12 500 бит/дюйм в каждой дорожке.

Данные записывают на магнитную ленту, используя определенную систему кодирования. Обычно в каждом ряду вдоль ленты запоминается один символ (рис. 7.48). Лента на рис. 7.48 имеет семь дорожек, или каналов, один из которых используется для бита четности, добавляемого для того, чтобы сделать число единиц в каждом ряду нечетным. Эти вопросы, а также коды, используемые для магнитной ленты, будут рассмотрены в следующей главе. Данные записываются на магнитную ленту блоками и обычно с особыми символами для сигнализации о начале и конце блока. В начале и конце бобины к ленте прикрепляют небольшой кусочек фольги, отражающей свет, а фотоэлектрические элементы, чувствительные к этим маркерам, предохраняют от выхода за пределы ленты (рис. 7.49, а).

Таблица 7.15. Характеристики лентопротяжного механизма с вакуумными колонками

Модель KENNEDY 9300 <sup>а)</sup>	
Характеристика	Спецификация
Плотность данных	9 дорожек, 800 символов/дюйм, 1600 символов/дюйм
Скорость ленты	125 дюйм/с
Время разгона-останова	3 мс при 125 дюйм/с
Перемещение ленты при разгоне-останове	0,6825 дюйма
Диаметр бобины	10,5 дюйма
Лента	
длина	2400 фут
ширина	0,5 дюйма
толщина	$1,5 \times 10^{-3}$ дюйма
Скорость перемотки	300 дюйм/с, номинал

<sup>а)</sup> С разрешения фирм Kennedy, an Allegheny International.



		0123456789 ABCDEF GH IJKLMNOP																												
Контроль	{	С	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
		В	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
Зона	{	А	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
		8	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
Число	{	8	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
		4	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
		2	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											
		1	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																											

Дорожка №		Значения битов																δ									
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
9	4																										
8	6																										
7	0																										
6	1																										
5	2																										
4	8																										
3	3																										
2	7																										
1	5																										

\*Позиция Р используется для бита контроля на нечетность

в

Рис. 7.49. Коды магнитной ленты.

а — обозначение начала и конца ленты; б — магнитная запись на ленте семидорожечного кода BCD; в — сравнение девятидорожечного (EBCDIC) и семидорожечного форматов данных на ленте.

Таблица 7.16. Характеристики лентопротяжного механизма  
Hewlett-Packard 7090E

Характеристика	Спецификация
Число дорожек	9
Скорость чтения-записи системы на базе 2100	25, 37,5, 45 дюйм/с
система 3000	45 дюйм/с
Плотность	1600 символов/дюйм (8 бит/символ)
Скорость передачи данных	72 000 символов/с, макс.
Диаметр бобины	10,5 дюйма, макс.
Лента	
ширина	0,5 дюйма
толщина	$1,5 \times 10^{-3}$ дюйма
Скорость перемотки	160 дюйм/с
Время разгона-останова	8,33 мс/(чтение после записи) при 45 дюйм/с
Система обнаружения конца и начала ленты, использующая отражаю- щие маркеры	Совместима с IBM

Коды, используемые для записи на ленту, различны; на рис. 7.49, б, в показаны два широко используемых кода IBM. Стандартная лента IBM шириной 1/2 дюйма и толщиной 1,5 миллидюйма имеет семь либо девять дорожек. На рис. 7.49, б показан семидорожечный код; нули представлены пробелами, а единицы — вертикальными черточками. На рис. 7.49, в изображен девятидорожечный код. Плотности записи: 200, 556, 800, 1600 или 6250 бит (или рядов) на дюйм [что означает 200, 556, 800, 1600 или 6250 символов (или байтов) на дюйм, так как в каждом ряду записывается один символ].

В табл. 7.15 приведены некоторые характеристики лентопротяжного механизма с вакуумными колонками средней стоимости, а в табл. 7.16 даны характеристики недорогого лентопротяжного механизма фирмы Hewlett-Packard с рычагами натяжения.

## 7.24. КАССЕТЫ И ОБОЙМЫ

Сменная кассета с лентой, используемая в знакомом всем бытовом магнитофоне, представляет собой привлекательное средство для записи цифровых данных. Кассеты имеют небольшие размеры, сменяемы и недорогие; их часто используют в малых и «домашних» компьютерах. Механизм перемещения ленты в обычной бытовой кассете, используемый часто для малых систем, не обладает достаточным качеством, необходимым для использования его в больших ЭВМ, работающих в сфере науки



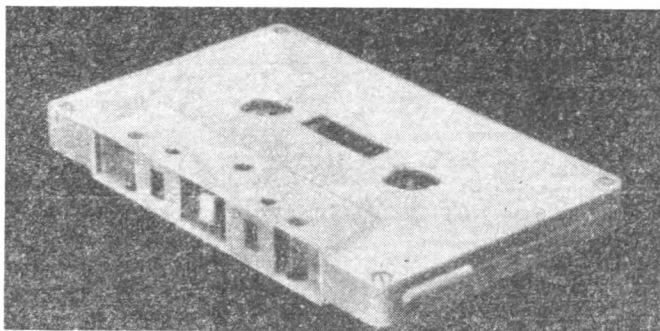


Рис. 7.50. Кассета магнитной ленты.

и производства. Тем не менее создано большое количество высококачественных цифровых кассет стоимостью порядка нескольких долларов (в основном от 2 до 15). Они небольшого размера — примерно такого, как знакомые кассеты, — и похожи на них по виду.

Имеются также большие **обоймы лент** (tape cartridge), содержащие длинные полосы магнитной ленты; они напоминают большие кассеты. Такие обоймы обеспечивают более удобный способ упаковки ленты и значительно упрощают установку бобины (что представляет проблему при работе с обычными бобинами ленты, когда лента должна устанавливаться на механизме вручную). Обоймы также предохраняют ленту от загрязнения и порчи, так как лента запечатана в катушке.

На рис. 7.50 показана типичная кассета. В настоящее время производится множество различных цифровых кассет и кассетных приводов с различными характеристиками. Так, например, фирма Data General предлагает кассетный привод со средней скоростью ленты 31 дюйм/с (кассета содержит 282 фута магнитной ленты шириной 0,15 дюйма), емкостью памяти ленты 800 000 бит и скоростью передачи (чтения) 12 800 бит/с. Кассета может перематываться за 85 с. Для обозначения начала и конца ленты используются 22-дюймовые отражающие маркеры начала и конца (фотодиод улавливает эти полосы).

Обоймы являются устройством для хранения на магнитных лентах, обладающим высокими эксплуатационными характеристиками. Существует несколько конструкций обойм. Они отличаются не только эксплуатационными характеристиками, но и распределением аппаратуры между обоймой и лентопротяжным механизмом. Часто используются обойма и привод фирмы 3М, изображенные на рис. 7.51. Обойма содержит 300 футов 1/4-дюймовой ленты, способной использовать до четырех дорожек с плотностью 1600 бит/дюйм при максимальной емкости

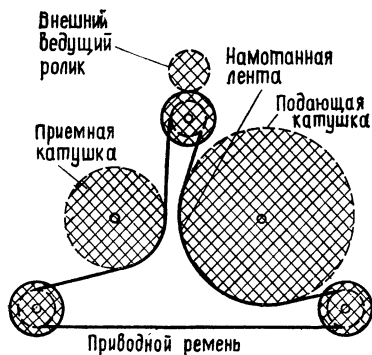
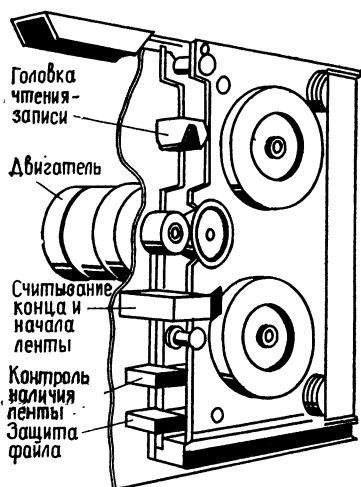


Рис. 7.51. Обойма и привод. (С разрешения фирмы 3М.)

памяти свыше  $2 \cdot 10^7$  бит. Лентопротяжный механизм 3М работает со скоростью 30 дюймов/с при чтении или записи и 90 дюймов/с в режиме поиска. Новейший привод на эластичных ремнях перемещает ленту, а также обеспечивает натяже-

Таблица 7.17. Спецификации обоймы и лентопротяжного механизма DCD-3 фирмы 3М

Характеристика	Спецификация
Скорость чтения-записи	30 дюйм/с вперед и назад
Скорость в прямом направлении, перемотка, при поиске по промежуткам	90 дюйм/с вперед и назад
Плотность записи	1600 бит/дюйм
Скорость передачи данных	48 Кбит/с, макс.
Межблочный промежуток	1,33 дюйма, типично; 1,2 дюйма, минимум по стандартам ANSI
Максимальная рекомендуемая частота разгона-останова	Три операции в секунду без принудительного воздушного охлаждения
Общее отклонение скорости от номинального значения	$\pm 4\%$ макс.
Головка ленты <sup>а)</sup>	1-, 2-, 4-канальные головки чтения-записи
Логика интерфейса	Совместима с ТТЛ
Питание	5 В пост. ток $\pm 5\%$ , $\pm 18$ В пост. ток $\pm 5\%$

<sup>а)</sup> Например, головки могут быть включены в каждую обойму.

Таблица 7.18. Сравнение характеристик запоминающих устройств

	5-дюйм. бобина	Кассета Phillips	Обойма типа 3М	Гибкий диск	Большой фиксиро- ванный накопитель на диске
Емкость, Кбайт	18 500	550	2500	250	571 000
Скорость пе- редачи, Кбит/с	180	9,6	48	250	14 000
Число дорожек	9	2	4	77	600
Плотность, бит/дюйм	880	880	1600	3200	1600
Межблочный промежу- ток, дюйм	0,6	0,8	1,3	Не приме- няется	Не приме- няется
Стоимость ме- ханизма	2000	400	500	400	30 000
Стоимость носителя среды, цент/байт	$0,06 \times 10^{-3}$	$1,2 \times 10^{-3}$	$0,6 \times 10^{-3}$	$2,6 \times 10^{-3}$	$0,185 \times 10^{-3}$

ние ленты. Привод ленты, катушки и направляющие компонен-  
ты относятся к основной части обоймы и не требуют никакого  
управления извне. Имеется несколько новых систем обойм, пред-  
назначенных для «поддержки» дисководов типа Винчестер.

В табл. 7.17 даны некоторые спецификации обоймы и ленто-  
протяжного механизма 3М. В табл. 7.18 сравниваются описан-  
ные стандартные запоминающие устройства. Заметим, что не-  
которые более современные устройства по своим характеристи-  
кам превосходят стандартные. В табл. 7.19 приводятся данные  
по новейшим, недорогим устройствам, пригодным для использо-  
вания в мини-ЭВМ и микроЭВМ.

Таблица 7.19. Характеристика недорогих запоминающих устройств

Характеристика	Емкость	Стоимость, цент/байт
Гибкий диск	2,4 Мбит	$2,6 \times 10^{-3}$
Кассета большой емкости	1 Мбайт	$5 \times 10^{-4}$
Кассета Phillips	1,44 Мбайт	$1,2 \times 10^{-3}$
Кассета малой емкости	200 Кбит	$20 \times 10^{-4}$
Обойма 3М	11,5 Мбайт	$0,6 \times 10^{-3}$
Бобины 7-дюймовой лен- ты	40 Мбит	$0,5 \times 10^{-4}$

## 7.25. ПАМЯТЬ НА ЦИЛИНДРИЧЕСКИХ МАГНИТНЫХ ДОМЕНАХ И ПАМЯТЬ НА ПРИБОРАХ С ЗАРЯДОВОЙ СВЯЗЬЮ

Вторичные, или вспомогательные, запоминающие устройства, которые до сих пор действительно пользовались успехом, были электромеханическими устройствами (барабаны, диски, лента, и т. д.), в которых биты хранились в виде магнитных полей на поверхности, а поиск данных осуществлялся с помощью механического перемещения. В настоящее время уже начали появляться два типа устройств для вторичной памяти, не имеющие никаких движущихся частей. Это память на цилиндрических магнитных доменах (ЦМД) — пузырьковая магнитная память и память на приборах с зарядовой связью (ППЗС).

ЗУ на ЦМД в первую очередь конкурируют с гибкими дисками, малыми дисками, обоймами и небольшими ленточными устройствами. ЗУ на ЦМД надежнее (так как не имеют движущихся частей), потребляют меньше энергии, компактнее и дешевле. Однако у дисков выше скорости передачи и ниже стоимость в расчете на один бит, за исключением малых систем.

Появление ЗУ на ЦМД связано с исследованиями в Bell Laboratories, которые показали, что биты можно запоминать в виде «пузырьков» в тонкой магнитной пленке на кристаллической подложке. Пузырьковое устройство функционирует подобно регистрам сдвига. Механизм запоминания состоит из магнитных доменов цилиндрической формы. Эти домены образуются в тонком слое пленки из монокристаллического синтетического феррита, когда перпендикулярно к поверхности пленки прикладывается магнитное поле. Вращающееся поле передвигает домены через пленку как в сдвиговом регистре. Наличие домена соответствует 1, отсутствие домена — 0. Домены перемещаются по пути, определяемому конфигурацией полосок мягкого магнитного материала, нанесенных на магнитную эпитаксиальную пленку.

Для пользователя физический принцип работы памяти на ЦМД менее важен, чем ее эксплуатационные характеристики. Запоминающие устройства подобны длинным сдвиговым регистрам, которые могут сдвигаться под внешним управлением. Память постоянна, так как если перемещение прекращается, биты в памяти будут оставаться бесконечно.

Чтобы лучше использовать характеристики сдвигового регистра и уменьшить время доступа, сдвиговые регистры обычно делают небольшой длины порядка 50—100 Кбит. В памяти может содержаться от нескольких сотен килобитов до нескольких мегабитов.

Скорость сдвига относительно невелика, примерно 200 кГц, так что время доступа порядка нескольких миллисекунд. (Чге-

ние и запись выполняются только на концах сдвигового регистра.)

ЗУ на ЦМД требуют относительно сложных интерфейсных схем, но изготовители ИС выпустили для этой цели специальные ИС.

Приборы с зарядовой связью (ПЗС) строятся с использованием технологии ИС. Биты хранятся на конденсаторах в виде зарядов, как в динамических ЗУ на ИС, за исключением того, что память собрана в форме сдвигового регистра с «пакетами» зарядов, которые сдвигаются от ячейки к ячейке под управлением синхронизирующих импульсов.

Так как запоминание осуществляется зарядом конденсатора, то при прекращении передвижения на длительный срок (несколько миллисекунд) произойдет утечка зарядов с конденсаторов и содержимое памяти будет потеряно.

Емкость памяти на приборах с зарядовой связью обычно составляет от 500 Кбит до нескольких мегабит. Сдвиговые регистры считываются и записываются с концов, так что время доступа зависит от длины сдвигового регистра. Скорость сдвига, как правило, равняется 200—500 кГц, так что для сдвиговых регистров небольшой длины время доступа измеряется миллисекундами.

Так как для производства запоминающих устройств на приборах с зарядовой связью используется технология ИС, для них требуется менее сложная схема интерфейса, чем для ЗУ на ЦМД. Метод, используемый для определения длины сдвиговых регистров как для памяти на ЦМД, так и для ПЗС, основывается на анализе соотношения «стоимость — производительность». В случае большого количества коротких цепочек время доступа меньше, но требуется больше схем для интерфейса и усложняется использование системы. Длинные цепочки более экономны, но приводят к большему времени доступа.

ЗУ как на ЦМД, так и на ПЗС находятся на ранней стадии своего развития, но уже сейчас рассматриваются наравне с более привычными дисковыми ЗУ.

## 7.26. МЕТОДЫ ЦИФРОВОЙ ЗАПИСИ

Хотя характеристики и конструкция таких запоминающих устройств, как магнитные барабаны, устройства для записи на ленту, дисковые ЗУ, могут быть очень разными, в основе процесса хранения для каждого из этих устройств лежит запоминание 0 или 1 на небольшом участке магнитного материала. В каждом случае запоминающая среда динамическая, так как носитель информации перемещается относительно считывающего или записывающего устройства.

Хотя процесс записи 0 или 1 на поверхности может показаться простым, тем не менее значительные исследования проводились в направлении совершенствования как знаков, используемых для представления 0 и 1, так и средств определения записанных значений. Необходимо, чтобы выполнялись два условия: 1) плотность записи должна быть как можно больше, т. е. каждая ячейка или бит должны занимать как можно меньшую площадь, что приводит, например, к экономии ленты, используемой для запоминания заданного объема информации, и 2) процедура чтения или записи должна быть выполнена как можно надежнее. Эти условия противоречивы, так как с уплотнением записи битов значительно увеличивается искажение считываемого сигнала.

При записи информации на магнитной поверхности цифровая информация подается записывающим схемам, которые затем преобразуют эту информацию в кодовую комбинацию, записываемую головкой записи. Методы записи информации на магнитном носителе можно разделить на несколько категорий: **с возвращением к нулю, с возвращением к смещению и без возвращения к нулю.** Методы чтения информации, записанной с помощью этих способов, также различны. Основные методы записи будут описаны ниже наряду с формами записываемых импульсов и импульсов, которые позже считывают, используя головки чтения, и транслируют с помощью системы чтения.

### **7.27. МЕТОДЫ ЗАПИСИ С ВОЗВРАЩЕНИЕМ К НУЛЮ И С ВОЗВРАЩЕНИЕМ К СМЕЩЕНИЮ**

На рис. 7.52 иллюстрируется метод записи с возвращением к нулю. На рис. 7.52, *а* ток через обмотку головки записи идет только тогда, когда следует записать 0 или 1. Когда нужно записать 1, в обмотку головки записи подается импульс тока положительной полярности, а когда должен быть записан 0, на обмотку подается отрицательный импульс тока. В обоих случаях после прекращения действия поданного импульса ток в обмотке головки записи отсутствует до тех пор, пока не будет записываться следующий бит. Второй ряд импульсов на этом рисунке иллюстрирует отпечаток (рельеф) остаточного потока на магнитной поверхности после того, как она прошла мимо головки записи. Этот отпечаток искажается из-за растекания потока вокруг головки.

Если этот отпечаток намагниченности пройдет под головкой чтения, то часть магнитного потока будет захвачена сердечником головки. Поток шунтируется магнитным сопротивлением материала сердечника головки, вместо того чтобы переходить через зазор в головке (рис. 7.39); при изменении величины потока

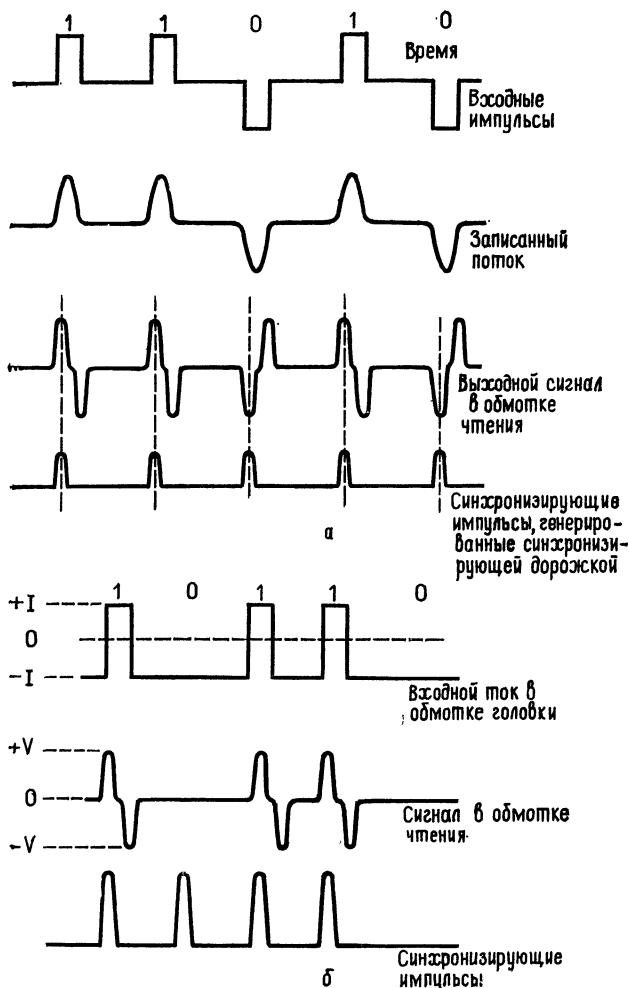


Рис. 7.52. Запись с возвращением к нулю (а) и запись с возвращением к смещению (б).

через материал сердечника в обмотке сердечника индуцируется э. д. с. Таким образом, изменение амплитуды записанного магнитного поля приводит к индуцированию э. д. с. в обмотке головки чтения. Формы импульсов на рис. 7.52, а, б иллюстрируют типичные выходные сигналы на обмотках головок чтения для каждого из этих методов. Заметим, что форма импульса на головке чтения не является точным воспроизведением ни входного тока во время процесса записи, ни фактического отпечатка намагниченности на магнитном материале.

Проблема, следовательно, состоит в том, чтобы различать на обмотке считывания значения, соответствующие 1 или 0. Для этого используют несколько способов. Один из них заключается в первоначальном усилении сигнала с головки чтения в линейном усилителе. Выходной сигнал этого усилителя затем стробируется так же, как выходной сигнал из обмотки считывания платы сердечников. Для барабанных систем правильная синхронизация для стробирования, которое должно быть очень точным, определяется синхронизирующими сигналами, записанными на синхронизирующей дорожке. Если выход усилителя чтения соединен с вентилем И, а стробирующий импульс также подается на этот вентиль, то выход будет положительным, когда записанный сигнал представляет собой 1.

Важно, чтобы синхронизирующий импульс был очень крутым и появлялся в точный момент времени по отношению к чтению или записи битов.

Основная особенность метода записи с возвращением к нулю заключается в том, что для 1 выходной сигнал в течение первой половины времени каждого бита будет положительным относительно второй половины, а для 0 выходной сигнал в течение первой половины времени каждого бита будет отрицательным по сравнению со второй половиной. Это обстоятельство иногда используется для считывания после записи (контрольного считывания).

При использовании метода записи с возвращением к нулю (рис. 7.52, а) магнитное поле возвращается в состояние с нулевым потоком, когда отсутствует импульс 1 или 0. Это делает невозможной новую запись по ранее записанной информации, если очень точно не установлена позиция каждой ячейки. Если импульс, соответствующий 0, записывается непосредственно на ранее записанную 1, то генерируемый поток изменит направление поляризации записанного поля на противоположное только тогда, когда головка записи будет находиться точно в том же положении, где она находилась при записи нуля. Следовательно, критическим фактором для такой системы является синхронизация записи информации и поэтому она редко используется, за исключением магнитных барабанов, в которых синхронизация может быть тщательно установлена с помощью синхронизирующих дорожек. При другом методе используется стирание всего потока перед записью новой информации, но он требует дополнительной головки стирания и поэтому редко применяется.

Второй метод записи информации — с возвращением к смещению — иллюстрируется на рис. 7.52, б. В этом случае ток через обмотку поддерживает головку в состоянии насыщения в отрицательном направлении, если не нужно записывать 1. Когда записывается 1, в середине битового времени в обмотку



подается импульс тока противоположной полярности. На рисунке показаны также выходные сигналы с обмотки считывания. В этом случае выходной сигнал появится в обмотке считывания, если только была записана 1. Выходной сигнал можно усилить и стробировать так же, как в предыдущем случае. Синхронизация здесь не является столь критичной, как в случае, когда информация «записывается поверх», так как отрицательный поток головки намагнитит поверхность в правильном направлении независимо от того, что было записано ранее. Предполагается, что ток через обмотку в этом и во всех последующих случаях достаточен для насыщения материала, на котором записываются сигналы. Основная проблема здесь состоит в считывании последовательности нулей. На магнитной ленте либо должна использоваться дорожка синхронизации, либо используемый код должен быть таким, чтобы в каждой строке ленты появлялась по крайней мере хотя бы одна 1. Заметим, что это связано с тем, что только единицы генерируют изменения магнитного потока и, следовательно, выходные сигналы на головке чтения.

## 7.28. МЕТОДЫ ЗАПИСИ БЕЗ ВОЗВРАЩЕНИЯ К НУЛЮ

На рис. 7.53 иллюстрируются три метода записи, каждый из которых классифицируется как система записи без возвращения к нулю. При использовании первого из них в течение всего битового времени, когда записывается 0, ток обмотки отрицательный, а в течение всего битового времени, когда записывается 1, положительный. Следовательно, ток обмотки будет оставаться постоянным при записи последовательности нулей или единиц и будет изменяться только тогда, когда за 0 следует 1 или когда вслед за 1 записывается 0. В этом случае сигнал в обмотке считывания будет индуцироваться только тогда, когда записанная информация меняется с 1 на 0 или наоборот.

Второй метод иногда называют **модифицированным без возвращения к нулю**, или методом **без возвращения к нулю с инверсией**. В этой системе записи полярность тока в обмотке записи меняется каждый раз, когда записывается 1, и остается постоянной, когда записывается 0. Если записывается последовательность единиц, полярность записанного потока будет меняться для каждой 1. Если записывается последовательность нулей, никаких изменений не будет происходить. Заметим, что в этой системе имеет значение изменение полярности, а не сама полярность. Таким образом, сигнал будет считываться после записи, если только записана 1. Такая система часто используется для записи на ленте, когда для генерации синхронизирую-

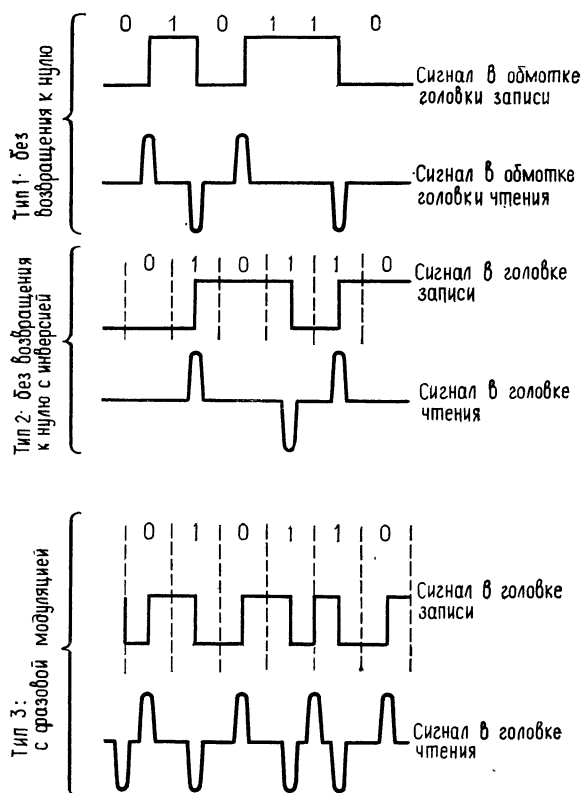


Рис. 7.53. Три типа записи без возвращения к нулю.

щих, или строб-импульсов, в каждой ячейке вдоль ширины ленты должна быть записана 1. Таким образом, если вдоль ленты записано 10 дорожек, одна из них должна быть синхронизирующей дорожкой, на которой записана последовательность единиц, каждая из которых определяет различные группы ячеек для считывания, или же информация должна быть закодирована таким образом, чтобы в каждой группе из 10 считываемых ячеек хотя бы в одной была 1. На ленте часто записывают буквенно-цифровую кодовую информацию; этот код может быть организован так, чтобы в каждом кодовом наборе была 1.

Третий метод записи без возвращения к нулю, иллюстрируемый на рис. 7.53, называют иногда **системой с фазовой модуляцией, фазового кодирования, гарвардской, манчестерской или системой расщепления частоты**. В этом случае 0 записывается в виде отрицательного импульса за  $1/2$  битового времени, вслед за которым за  $1/2$  битового времени записывается поло-

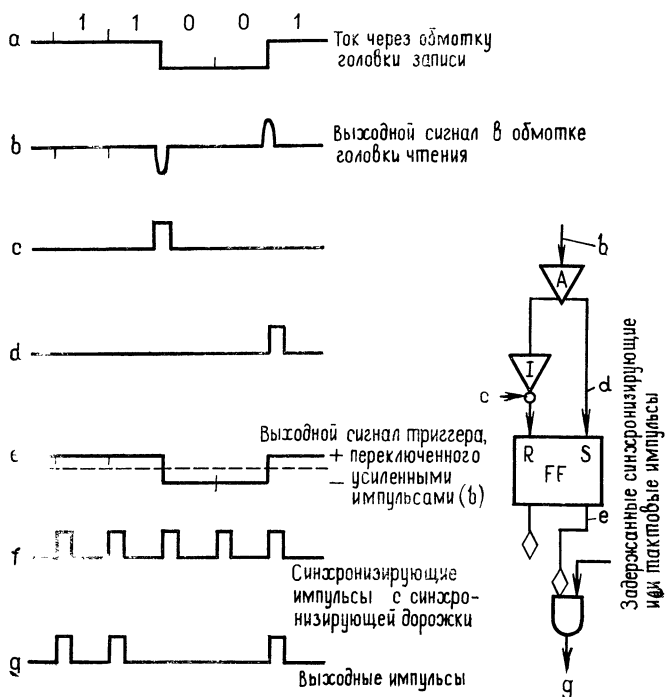


Рис 7.54. Запись методом без возвращения к нулю.

жительный импульс, а 1 записывается в виде положительного импульса за  $1/2$  битового времени, вслед за которым за  $1/2$  битового времени записывается отрицательный импульс. Такой метод записи часто используется в быстродействующих системах.

Чтение записанной информации состоит из двух этапов. Сначала выходной сигнал из головки чтения усиливается, а затем усиленные сигналы преобразуются логическими схемами. На рис. 7.54 иллюстрируется способ преобразования для первой из систем без возвращения к нулю, показанных на рис. 7.53. Выходные сигналы могут иметь форму либо выходного сигнала триггера, либо представлять собой последовательность импульсов. Синхронизирующие импульсы появляются каждый раз, когда ячейка проходит под головками чтения.

Триггер (рис. 7.54) реагирует только на положительные импульсы. Следовательно, сигналы положительных импульсов на записывающей головке будут «устанавливать» триггер в 1. Инвертор на входе С преобразует отрицательные импульсы в положительные. Эти положительные импульсы будут сбрасы-

вать триггер в состояние 0. Непосредственно в ЭВМ могут использоваться сигналы с выхода триггера или же импульсы, полученные с вентиля И, на входы которого подаются единичный выход триггера и задержанные синхронизирующие импульсы. Можно сформировать также последовательное представление числа, записанного вдоль поверхности.

## ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

В. Гамахер, З. Вранесик, С. Заки

### 8.1. ВВЕДЕНИЕ

Термин «программное обеспечение» относится ко всем программам, составленным для выполнения на ЭВМ. Эти программы могут быть записаны на любом из языков программирования. Программы бывают самых разных размеров. Студенческие программы для решения небольших численных задач могут включать от 20 до 50 операторов языка высокого уровня, например Фортрана. С другой стороны, программы учета данных и управления ими, которые используются крупными фирмами или административными учреждениями, насчитывают тысячи операторов. Эти примеры относятся к программам, которые обычно называют прикладными или программами пользователей. Такие программы состояются лицами, применяющими ЭВМ для решения научных и управленческих задач.

Изготовители вычислительных систем и соответствующих средств обслуживания поставляют программы другого класса, которые в отличие от программ пользователей называются программами вычислительной системы, или системным программным обеспечением. Системное программное обеспечение включает программы, которые транслируют программы пользователей в программы на машинном языке. Другие системные программы используются для загрузки этих оттранслированных программ в оперативную память перед их выполнением. Программы-трансляторы иногда называют процессорами языка программирования. Набор подпрограмм, применяемых для управления работой аппаратных ресурсов [центрального процессора (ЦП), оперативной памяти (ОП), массовой памяти большой емкости, устройств ввода-вывода и т. д.], составляет важную часть системного программного обеспечения в вычислительной системе. К числу таких подпрограмм относятся и программы операционной системы.

С целью уменьшения стоимости вычислений для индивидуальных пользователей ресурсы больших вычислительных систем почти всегда разделяются между рядом независимых пользовательских программ. Программы операционной системы предназначены для управления этим разделением ресурсов таким образом, чтобы вычислительная система функционировала наиболее эффективным образом и вместе с тем была удобной для пользователя. Стремление к столь идеализированной цели способствовало проведению научно-исследовательских работ в широком масштабе, что привело к проектированию и созданию операционных систем.

## 8.2. ЯЗЫКИ И ТРАНСЛЯТОРЫ

Наиболее элементарным языком является машинный язык, в котором программы представляются с помощью двоичных кодов машинных команд и элементов данных (байтов или слов). На рис. 8.1 приведена программа на машинном языке. Двоичные наборы программ на машинном языке представляются как в восьмеричной, так и шестнадцатеричной системе. Шестнадцатеричная система очень удобна для записи байтов и слов для 16- и 32-разрядных машин, поскольку наборы из 8, 16 и 32 бит могут быть закодированы соответственно двоичными, четверичными, восьмеричными и шестнадцатеричными цифрами.

Программа на машинном языке для конкретной ЭВМ может быть выполнена на ней без использования какой-либо другой

Позиция в оперативной памяти		Содержимое слов в памяти (16 бит)		
Восьмеричный адрес байта	Шестнадцатеричный адрес байта	Вариант на машинном языке		На языке ассемблера
		восьмеричный	шестнадцатеричный	
212	8A	005000	0A00	CLR R0
214	8C	012703	15C3	MOV #—16., R3
216	8E	177760	FFF0	
220	90	006300	0CC0	MLOOP: ASL R0
222	92	006101	0C41	ROL R1
224	94	103002	8602	BCC NOADD
226	96	060200	6080	ADD R2, R0
230	98	005501	0B41	ADC R1
232	9A	005203	0A83	NOADD: INC R3
234	9C	001371	02F9	BNE MLOOP
236	9E	000000	0000	HALT

Рис. 8.1. Программа для умножения чисел на машинном языке и языке ассемблера.

программы. Однако программирование на машинном языке является трудоемким, и поэтому программы обычно пишутся на языке, имеющем более символическую и стилизованную форму. Простейшими являются языки ассемблера. Проблемно-ориентированные языки, такие, как Фортран, ПЛ-1, Кобол, Алгол и др., часто называют языками высокого уровня в связи с тем, что они имеют широкий набор операционных средств и управляющих операторов, которые существенно превосходят основные типы команд в языках ассемблера.

Программа, написанная на любом из языков, имеющих более высокий уровень, чем машинный язык, называется **исходной программой**. Исходные программы транслируются в программы на машинном языке с помощью системных программ, называемых **трансляторами**. Для языков ассемблера эти трансляторы называются ассемблерами. Когда исходная программа представлена на языке высокого уровня, транслятор называется **компилятором**. Результат процесса трансляции называется **объектной программой**. В простейшем случае объектная программа получается на машинном языке; она может быть загружена в оперативную память и выполнена сразу. Иногда части больших исходных программ транслируются отдельно в отдельные объектные программы. Для соединения этих объектных программ в единую программу на машинном языке, которая может непосредственно выполняться, требуется их дальнейшая обработка.

Следует отметить ряд особенностей языков ассемблера и языков высокого уровня. В основном язык ассемблера представляет собой структурированное множество мнемонических обозначений для соответствующего машинного языка. Однако в некоторых расширенных языках ассемблера простые операторы (с параметрами) могут соответствовать коротким последовательностям машинных команд. Указанные типы операторов часто называются **макрокомандами** или просто **макрос**. Ассемблер, который генерирует версию машинного языка этих типов операторов, автоматически распространяет их на соответствующую последовательность машинных команд. Такой ассемблер иногда называют **макроассемблером**. Языки ассемблера, даже те, которые располагают макросредствами, тесно связаны с машинными языками. С другой стороны, многие операторы в языках высокого уровня должны транслироваться в отдельные, но взаимосвязанные последовательности машинных команд. Этот процесс сложнее процесса расширения макрокоманд. Использование языков высокого уровня облегчает работу программиста при описании желаемых действий ЭВМ, исключая необходимость явного определения способа компоновки машинных команд для их выполнения. Эти языки характеризуются наличием управляющих структур, таких, как циклы **DO**, операторов

IF ... THEN ... ELSE, PROCEDURES с параметрами, а также структур данных вида ARRAYS, различных типов данных.

Для выполнения программ на языке высокого уровня не обязательно транслировать их в программы на машинном языке. Иногда желательно произвести трансляцию программы в программу на промежуточном языке, которая затем выполняется с помощью другой программы, называемой **интерпретатором**. Интерпретатор — это программа, которая считывает интерпретируемую программу и выполняет каждый ее оператор. Этот процесс осуществляется значительно медленнее, однако в ряде случаев метод интерпретации является наилучшим с системной точки зрения. Например, при создании коротких студенческих программ, выполняемых лишь один раз после отладки, этот метод может оказаться более эффективным. Управляемое выполнение программы в соответствии с методом интерпретации позволяет предоставить в распоряжение пользователя более обширную и содержательную диагностику ошибок в программе. Интерактивный процесс вычислений, при котором пользователь осуществляет ввод и выдает запросы на выполнение нескольких операторов в системе, а также получает ответы через терминал до перехода к дальнейшим вычислениям, естественным образом обрабатывается с помощью метода интерпретации. С другой стороны, предпочтение обычно отдается компиляции программ в машинный язык, если они должны использоваться многократно с различными входными данными. Примерами подобного рода являются широко распространенные пакеты математических подпрограмм для численного интегрирования и решения линейных уравнений или же большие программы обработки управленческой и экономической информации.

### 8.3. ЗАГРУЗЧИКИ

Предположим, что данная программа оттранслирована в программу на машинном языке. Для выполнения эта программа должна быть загружена в оперативную память ЭВМ. Так как этот процесс не простой, рассмотрим некоторые его этапы.

Принципы, используемые при загрузке программы в ОП для ее выполнения, удобно описать на примере. Рассмотрим программу на машинном языке, пробиту на бумажной ленте в формате, показанном на рис. 8.2. Символы на бумажной ленте имеют ширину 8 бит (один байт), так что для описания содержимого слова в 16-разрядной ЭВМ потребуются два символа. Пустая (без пробивок) лента воспринимается устройством считывания как набор нулей, поэтому начало информации отмечается посредством байта начала, имеющего значение 1. Сле-



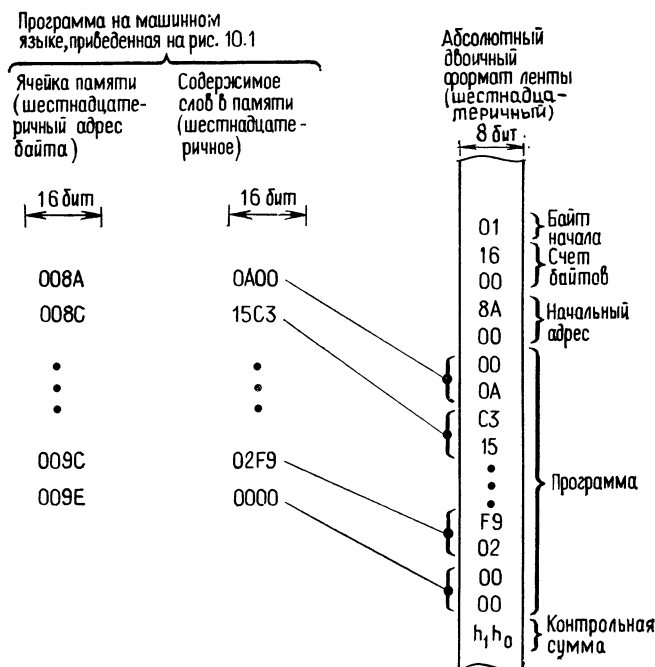


Рис. 8.2 Расположение программы на бумажной ленте в формате абсолютного двоичного загрузчика.

дующие два байта задают шестнадцатеричное число, которое указывает общее число байтов в данном сегменте программы. Начальный адрес программы указывается с помощью следующих двух байтов. Затем в сегменте программы перечисляются все слова загружаемой программы, начиная каждое слово с младшего байта. В конце сегмента программы помещается контрольная сумма, образуемая 8 бит, которая представляет собой сумму байтов программы по модулю 256. Контрольная сумма дает программисту некоторую возможность для обнаружения ошибок при электромеханическом преобразовании пробивок на бумажной ленте в элементы байта в устройстве считывания с бумажной ленты. Любая одиночная ошибка при считывании сегмента программы выявляется путем сравнения контрольной суммы со считанным значением суммы (внутренний подсчет).

Для загрузки такой программы в ОП необходимо использовать другую программу, уже находящуюся в памяти. Эта программа называется **абсолютным двоичным загрузчиком**. Она относительно небольшого размера и ее можно ввести в память

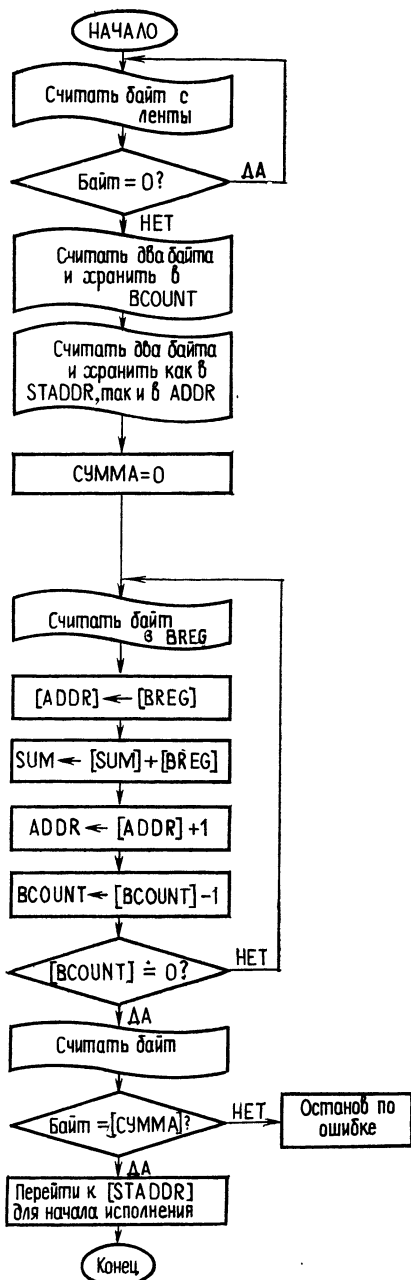
вручную при помощи переключателей лицевой панели, которые имеются во всех ЭВМ. Ранее был рассмотрен программно-управляемый ввод-вывод. Поэтому для читателя не составит большого труда создать программу абсолютного двоичного загрузчика, которая считывает байты с бумажной ленты, пробитой в формате, приведенном на рис. 8.2. Загрузчик размещает слова программы в необходимую область памяти, осуществляя подсчет внутренней контрольной суммы при загрузке программы. На рис. 8.3 показана блок-схема подобной программы. Программа для типичной мини-ЭВМ, соответствующая этой блок-схеме, может насчитывать примерно 50 команд. Это весьма небольшое количество битов для ручного ввода в ОП с переключателей панели.

Для загрузки программ такого вида как абсолютный двоичный загрузчик в ОП можно составить более простые программы загрузки. Обычно эту более простую программу называют **загруз-**

Рис. 8.3. Блок-схема абсолютного двоичного загрузчика.

*Примечания*

1. Используемая ЭВМ является 16-разрядной с адресуемым байтом.
2. BCOUNT (ячейка памяти для слова) содержит число байтов в загружаемой программе.
3. STADDR (ячейка памяти для слова) содержит начальный адрес загружаемой программы.
4. ADDR (ячейка памяти для слова) содержит адрес, куда должен загружаться следующий байт программы.
5. BREG — регистр для хранения байта.
6. СУММА (ячейка памяти для байта) содержит сумму всех байтов программы по модулю 256.



**чиком программы раскрутки** (или загрузчиком предввода). Он не производит накопления контрольной суммы и может быть построен так, чтобы отпала необходимость в прямом подсчете загружаемых байтов. Для обеспечения этого формат программ, вводимых с помощью загрузчика предввода, должен быть сложнее по сравнению с форматом на рис. 8.2. Подробное рассмотрение работы загрузчиков предввода не входит в наши намерения, так как они до некоторой степени зависят от типа ЭВМ и с целью их сокращения обычно прибегают к хитроумным приемам программирования.

Выше были представлены идеи, используемые в простейших загрузчиках для программ на машинных языках. Во многих вычислительных системах, в частности в больших системах, возникает потребность в более сложном загрузчике, называемом **настраивающим загрузчиком (перемещающим загрузчиком)**. Сначала следует обосновать необходимость в таком загрузчике. В случае абсолютного двоичного загрузчика точка начала загружаемой программы фиксируется в момент написания программы. В общем случае программа будет выполняться правильно только тогда, когда она введена в ОП именно с этой ячейки памяти. Для многих ЭВМ характерно то, что при обычных рабочих условиях в их ОП одновременно находится множество различных программ. Эти программы отличаются по размерам, так что желательно располагать некоторой степенью свободы при определении местоположения конкретной программы в момент ее загрузки. Поэтому функция настраивающего загрузчика заключается в приеме программы на машинном языке, составленной в предположении, что она будет загружаться, начиная с ячейки памяти  $x$ , которая обычно принимается за 0 (для удобства), и загрузке ее, начиная с ячейки памяти  $y$ . Вообще для такого загрузчика потребуются внесение некоторых изменений в программу, чтобы она правильно выполнялась в новой области памяти.

Рассмотрим формат, который должна иметь объектная программа, если она обладает свойством перемещаемости. Когда все адреса в программе заданы относительно счетчика программы, она будет выполняться правильно независимо от ее местоположения в памяти. С другой стороны, допустим, что в некоторых частях программы используется абсолютная адресация памяти. Такие адресные значения называются **адресными константами**. Настраивающий загрузчик должен откорректировать эти адресные константы в соответствии с областью памяти, куда будет загружаться программа. Если объектная программа составлена так, что она будет правильно выполняться при условии ее загрузки с ячейки памяти 0, то корректировка просто сводится к прибавлению адреса фактической начальной ячейки

памяти ко всем адресным константам в программе, что вызывает их смещение. Ясно, что позиции адресных констант в объектной программе должны быть указаны транслятором. В следующем разделе будет приведен пример, иллюстрирующий этот способ.

Некоторые ЭВМ обладают особенностями, которые упрощают перемещение программ. Предположим, что все адреса генерируются относительно базового регистра. Во время выполнения программы содержимое базового регистра используется как смещение адресов, порождаемое информацией о способе адресации. В этом случае перемещение программы может осуществляться просто вводом начального адреса в базовый регистр. Таким образом, сложность обработки адресов транслятором и загрузчиком может быть в значительной степени уменьшена благодаря особенностям организации конкретной ЭВМ. Указанное взаимодействие аппаратных средств машины с основным системным программным обеспечением составляет важный аспект разработки вычислительных систем.

#### 8.4. РЕДАКТОРЫ СВЯЗЕЙ

До сих пор рассматривался простой случай загрузки программ, которые были оттранслированы в машинный язык как единое целое. На практике чаще встречается случай, когда большая программа состоит из определенного числа подпрограмм, написанных отдельно друг от друга и, возможно, различными людьми. Часто оказывается полезным транслировать эти подпрограммы независимо друг от друга в машинный язык. Результат трансляции каждой из указанных независимых подпрограмм называется **объектным модулем**. Объектные модули должны иметь формат, который позволяет соединить и связать их в единую программу на машинном языке, передаваемую загрузчику. Это означает, что некоторые символы в исходной программе должны быть выделены программистом как **внешние символы**. Внешними считаются те переменные или строки меток, на которые ссылаются из одной или нескольких отдельно оттранслированных программ или подпрограмм. Программа, соединяющая объектные модули в единую программу на машинном языке, которая может быть передана загрузчику, называется **редактором связей**. Программа на его выходе называется **загрузочным модулем**. Следует отметить, что, как правило, функции редактора связей и загрузчика совмещаются в единой системной программе, которая называется связующим загрузчиком. В этом случае генерация загрузочного модуля является промежуточным шагом при выполнении связующего загрузчика.

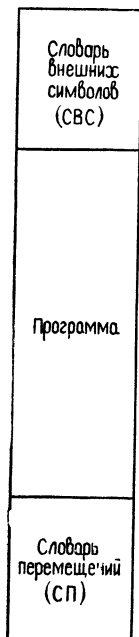


Рис. 8.4. Объектный модуль.

Возможны некоторые компромиссные варианты. Вместо того чтобы пытаться связать и загрузить отдельно оттранслированные программы, иногда целесообразно объединить все входные программы и подпрограммы, требуемые для некоторого сложного задания, и представить их в виде единого целого для трансляции, в результате чего генерируется загрузочный модуль. В этом случае разрешение перекрестных ссылок возлагается на транслятор. Однако в данном разделе будем считать оправданной раздельную трансляцию подпрограмм.

Для дальнейшего изложения будут использованы те же соображения, что и в случае абсолютного двоичного загрузчика. Будет задан формат для объектных модулей, порожденных в результате раздельных трансляций. Основной новой идеей, используемой в этом формате, является спецификация внешних символов, определенных так же, как и выше. На рис. 8.4 показан возможный формат для объектных модулей. Этот общий формат начинается с таблицы внешних символов, называемой словарем внешних символов (СВС), за которым следует текст программы на машинном языке. Список местоположения адресных констант, которые должны быть перемещены загрузчиком, размещается после программы в секции, называемой

словарем перемещений (СП). Адресные константы должны быть помечены. Будем считать, что эти константы отождествлены с их местоположением в списке.

Рассмотрим конкретный пример двух программ А и В, которые оттранслированы раздельно. Программа А определяет две ячейки памяти DATAWORD1 и DATAWORD2, к которым следует иметь доступ из обеих программ. Программа А обращается также к программе В как к своей подпрограмме, поэтому она должна ссылаться на точку входа SUBRB программы В. На рис. 8.5 приведен возможный формат представления этих программ для редактора связей. В каждой строке таблицы СВС имеется признак для идентификации. В первой строке указаны длина программы вместе с ее именем, обозначенным через Р. Во второй и третьей строках СВС для программы А описаны DATAWORD1 и DATAWORD2 как внешние символы. Они являются адресами ячеек памяти 120 и 121 этой программы. Четвертая строка SUBRB предназначена для ссылки (R) на адрес в программе В. Эта ссылка содержится в ячейке памяти 180 программы А. Адрес SUBRB должен быть задан в

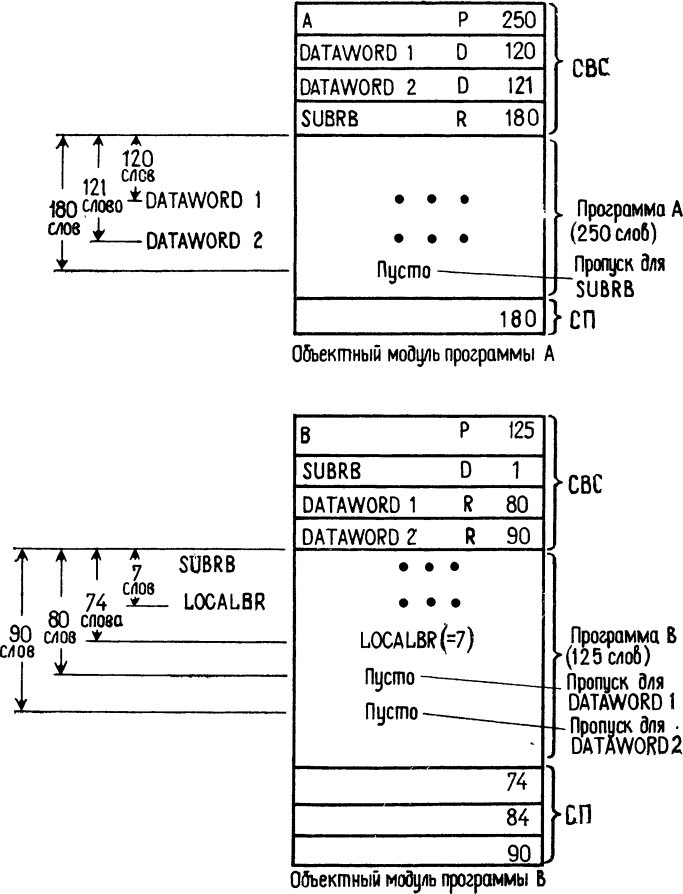


Рис. 8.5. Объектные модули для программ А и В.

СВС программы В. Когда ячейка памяти для слова с меткой SUBRB определяется во время выполнения процесса связывания, его значение должно быть записано в ячейку памяти 180 программы А. Тот факт, что это значение должно быть перемещено во время ввода окончательного загрузочного модуля в ОП, также подчеркивается включением ячейки памяти 180 в СП. Это справедливо и для объектного модуля программы В. В этот модуль включается дополнительная локальная ссылка, которая представлена внутренней адресной константой LOCALBR. Ссылка производится в слове 74, что отмечается в СП. На рисунке эта адресная константа равна 7. Остальные

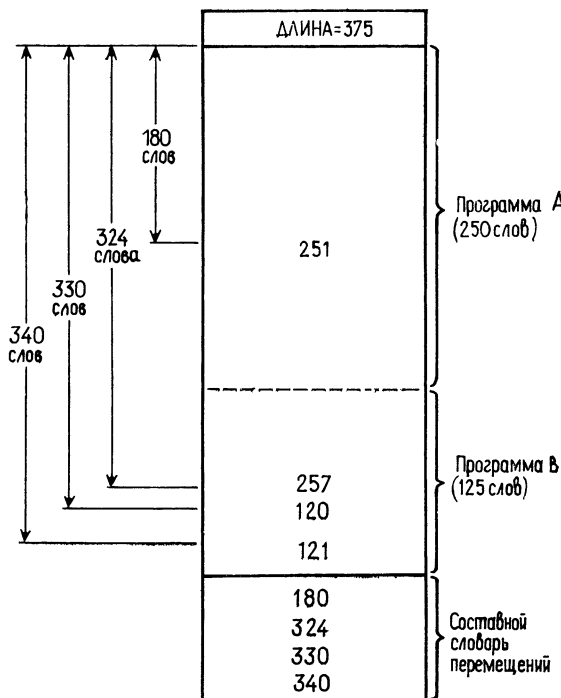


Рис. 8.6. Загрузочный модуль, полученный после связывания объектных модулей для программ А и В.

два элемента СП указывают местоположение адресных констант DATAWORD1 и DATAWORD2.

Пусть эти два объектных модуля передаются редактору связей в следующем порядке: сначала модуль А, а затем модуль В. Редактор связей должен построить загрузочный модуль, состоящий из 375 слов программ А и В, в начале которого находится описание длины программы, а в конце — составной СП. Очевидно, что в отдельных СВБ и СП содержится достаточно информации для построения загрузочного модуля, показанного на рис. 8.6. Например, с помощью связывающего процесса устанавливается, что SUBRB занимает 251-ю ячейку памяти составного загрузочного модуля. Поэтому адресная константа 251 помещается в слово 180 тела программы загрузочного модуля. Аналогично адресная константа 7 в 74-й строке программного модуля В заменяется на адресную константу 257 ( $= 250 + 7$ ), когда она помещается в 324-ю строку ( $= 250 + 74$ ) окончательного загрузочного модуля. Теперь этот загрузочный модуль может передаваться программе загрузчика. Как только для

загрузчика определяется начальный адрес S, он загружает программу длиной в 375 слов в ОП, начиная с указанной ячейки памяти. В этом случае он прибавляет значение S — 1 к константам 251, 257, 120 и 121 в ячейках памяти 180, 324, 330 и 340 загрузочного модуля. Программа, в окончательном виде насчитывающая 375 слов в ОП, показана на рис. 8.7, где в качестве начального адреса выбран 3401.

Модель процессов связывания и загрузки, рассмотренная в данном разделе, соответствует упрощенному варианту основных принципов, используемых при спецификации СВС и СП. Для ее практической реализации требуется, как правило, введение некоторого числа различных классов внешних символов. Благодаря этому пользователь получает большую степень свободы при написании программ, которые должны быть связаны друг с другом, но транслируются отдельно.

	⋮
3401	
3580	3651
3724	3657
3730	3520
3740	3521
3775	
	⋮

## 8.5. ОПЕРАЦИОННЫЕ СИСТЕМЫ

Во введении данной главы было отмечено, что ресурсы вычислительной системы обычно разделяются между некоторым числом программ пользователей с целью снижения стоимости вычислений для каждого пользователя. Это снижение стоимости вычислений будет реальным, только если разделение ресурсов повышает производительность системы, т. е. увеличивает число программ пользователей, выполняемых за единицу времени. Данный раздел посвящен рассмотрению способов разделения ресурсов между заданиями пользователей и описанию программ операционной системы (ОС), которые пытаются эффективно осуществить это разделение, тем самым повышая производительность.

Пусть вычислительная система состоит из центрального процессора, оперативной памяти, устройства считывания с карт, построчно-печатающего устройства и вспомогательного запоминающего устройства на магнитных дисках. Устройства считывания, печати и дисковое запоминающее устройство связаны с ОП тремя логически независимыми каналами. Эти каналы позволяют производить обмен данными между ОП и указанными тремя внешними устройствами одновременно с выполнением программ центральным процессором. Именно эта возмож-

Рис. 8.7. Размещение загрузочного модуля, представленного на рис. 8.6, в основную память, начиная с ячейки 3401.



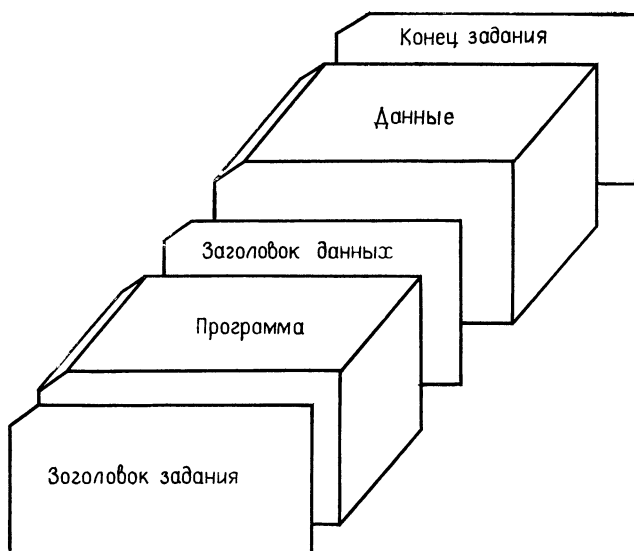


Рис. 8.8. Колода задания.

ность обеспечения связи ОП и внешних устройств с одновременным выполнением программ должна стать источником повышения скорости обработки программ пользователей. Основная функция программ ОС заключается в планировании, иницировании и управлении работой как каналов, так и центрального процессора.

Под термином **задание** будут подразумеваться все операции ввода-вывода и вычисления, связанные с заданной программой пользователя. Для этапов ввода, вычисления и вывода, требуемых для выполнения задания, удобно пользоваться термином **шаг**. Пусть пользователь выдал системе полный запрос на выполнение задания в виде колоды перфокарт, определяющих программу и данные, и ожидает получить выходную распечатку после полного выполнения задания. Сам пользователь не принимает никакого участия в процессе обработки задания.

Рассмотрим процесс обработки ряда заданий, структура колоды перфокарт которых показана на рис. 8.8. Функции карт «Заголовок задания», «Заголовок данных» и «Конец задания» очевидны. Карты «Заголовок задания» выполняют функцию, отличную от функции описателя задания. В табл. 8.1 приведена последовательность действий, которая может потребоваться во время обработки типичного задания. На первом шаге задания программа считывается в ОП из устройства считывания с карт. На рис. 8.8 форма программы не указана. Если она записана на входном языке высокого уровня, то ОС направит

Таблица 8.1. Пример последовательности выполнения задания

Номер шага задания	Операция	Используемые устройства системы
1	R1 — ввод программы	Устройство считывания с карт
2	C1 — вычисления	Центральный процессор
3	R2 — ввод данных	Устройство считывания с карт
4	C2 — вычисления	Центральный процессор
5	R3 — ввод данных	Устройство считывания с карт
6	C3 — вычисления	Центральный процессор
7	W1 — вывод данных	Построчно-печатающее устройство

текст программы в соответствующий компилятор для его трансляции в машинный язык. Можно допустить, что любая операция обработки, требуемая для преобразования программы к виду, готовому к выполнению, включена в шаг вычислений C1. Этот шаг также содержит начальную фазу выполнения программы. Выполнение программы продолжается до тех пор, пока на шаге R2 не потребуются входные данные. На этом шаге ввода данных необходима передача устройством считывания информации с некоторых карт в ОП для последующей обработки во время выполнения шага вычислений C2. Остальная часть данных считывается во время выполнения шага R3; окончательные результаты печатаются на шаге W1.

При **пакетной обработке** задания выполняются по одному строго последовательно в таком порядке, в котором задания пакета размещаются в устройстве считывания с карт. Существуют определенные возможности для совмещения операций ввода-вывода с вычислениями в отдельных заданиях. Например, в задании, приведенном в табл. 8.1, устройство считывания с карт может выполнить шаги 3 и 5, как только заканчивается шаг 1. Данные считываются в буферную область ОП каналом устройства считывания с карт при одновременном выполнении шага вычисления C1. В этом задании печать выходных данных является последним шагом. Если допустить, что он не может начинаться до завершения шага вычисления C3, то исключается возможность совмещения вывода данных с вводом или вычислениями. В других заданиях частично могут перекрываться шаги вывода данных и вычислений. Отсюда можно заключить, что в пределах отдельных заданий возможно некоторое совмещение шагов ввода-вывода данных и вычислений. Однако степень подобного перекрытия значительно изменяется от задания к заданию.

Если рассматривать одновременно запросы на выполнение ряда заданий, вместо того чтобы в каждый момент иметь дело с запросом на выполнение только одного задания, то можно

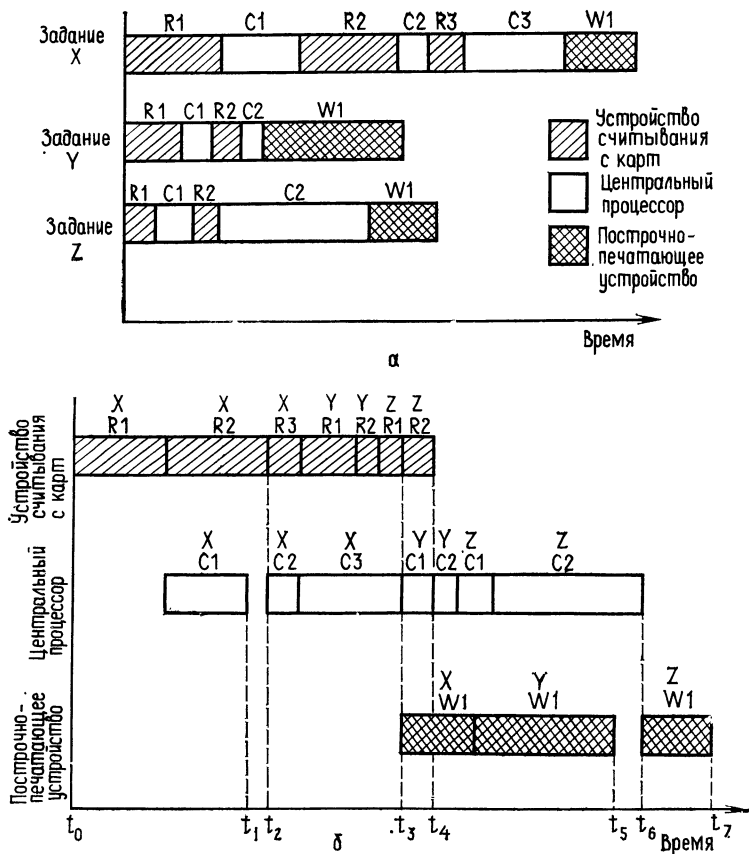


Рис. 8.9. Совмещение шагов ввода, вывода и вычислений в системе, работающей в режиме пакетной обработки.

$a$  — линейные временные диаграммы выполнения трех заданий;  $b$  — совмещение шагов заданий.

располагать большими возможностями для совмещения работы физических ресурсов системы. Поэтому прямым расширением простой системы пакетной обработки, описанной выше, является считывание колоды карт для заданий с номерами  $N+1$ ,  $N+2$ , ... в буферную область ОП во время выполнения вычислений, предусмотренных в задании с номером  $N$ . Предполагается, что колода карт для  $N$ -го задания уже считана. По мере выполнения вычислений  $N$ -го задания выходные данные заданий с номерами ...,  $N-2$ ,  $N-1$  могут быть переданы из другой буферной области ОП в постстрочно-печатающее устройство. Задания по-прежнему обрабатываются в порядке их раз-

мещения в пакете, но при этом появляется потенциальная возможность эффективного совмещения шагов смежных заданий.

Рассмотрим подробнее на конкретном примере особенности совмещения. Удобно ввести временную линейную диаграмму, показывающую время, требуемое для выполнения каждого шага задания пользователя. На рис. 8.9, *а* приведены диаграммы для задания пользователя, описанного в табл. 8.1, которое обозначим через  $X$ , а также для двух других заданий  $Y$  и  $Z$ . Как уже упоминалось, имеется возможность совмещения шагов как в пределах отдельных заданий, так и относящихся к различным заданиям. Однако существуют определенные пределы совмещения, так как некоторые шаги на данном устройстве могут быть начаты только после выполнения соответствующих шагов на других устройствах. Например, выполнение шага  $S_2$  задания  $X$  нельзя начать до тех пор, пока не закончится шаг  $R_2$  этого задания. Учет подобных ограничений иллюстрируется на рис. 8.9, *б*, где указаны совмещения, достигаемые при выполнении различных шагов трех заданий  $X$ ,  $Y$  и  $Z$ . Предполагалось, что эти задания следуют друг за другом в пакете в указанном порядке и система начинает работать в момент  $t_0$ . Максимальное совмещение происходит в промежутке времени от  $t_3$  до  $t_4$ . Интервал времени от  $t_1$  до  $t_2$  является временем простоя центрального процессора, так как, согласно упомянутому ограничению, шаг  $S_2$  задания  $X$  может начинаться лишь после завершения шага  $R_2$ . Аналогично промежуток времени от  $t_5$  до  $t_6$  является временем простоя печатающего устройства, поскольку шаг  $W_1$  задания  $Z$  должен выполняться после шага  $S_2$  задания  $Z$ .

Сделаем несколько замечаний относительно рис. 8.9б. Если промежуток времени от  $t_0$  до  $t_7$  рассматривать как период времени, в течение которого система продолжает работать, в промежутке от  $t_0$  до  $t_3$  может осуществляться вывод данных на память для заданий, обработанных до задания  $X$ . Кроме того, колоды заданий, следующие за колодой для задания  $Z$ , могут считываться после момента  $t_4$ .

### Спулинг<sup>1)</sup>

До сих пор мы исходили из допущения, что имеется достаточный объем ОП для входных и выходных буферов, требуемых для достижения разумного уровня совмещения. Если оно не

---

<sup>1)</sup> Термин «спулинг» (Spooling, Simultaneous Peripheral Operation on Line) имеет значение ввода (или вывода) в реальном масштабе времени в промежуточный носитель (магнитную ленту, диски) вместо ввода с перфокарт и вывода на печать. Этот термин часто переводится как подкачка (от качка) данных. — *Прим. ред.*

выполняется, то дополнительный объем памяти, необходимый для совмещения, может обеспечиваться магнитным диском. Теперь следует выяснить, приведут ли к общему повышению эффективности дополнительные затраты, требуемые для буферизации информации из ОП на диск и, обратно, с диска в ОП. Хотя подробный анализ этого вопроса не является нашей целью, отметим, что подобные системы должны способствовать достижению большей эффективности работы в целом. Процесс непрерывного считывания колод заданий в ОП и буферизация их на диске до назначения им времени центрального процессора называется **спулингом ввода**. Аналогичный процесс буферизации выходных данных на диске и их обратный ввод в ОП с целью передачи затем в печатающее устройство называется **спулингом вывода**.

Вообще спулинг более равномерно распределяет запросы отдельных заданий на устройства ввода-вывода, создавая устойчивую нагрузку для устройства считывания с карт и построочно-печатающего устройства. В уже рассмотренной системе с пакетным режимом обработки могут быть моменты, когда устройство считывания с карт простаивает, а также моменты, когда оно считывает следующее задание, а остальные компоненты системы простаивают. Аналогичные замечания касаются и построочно-печатающего устройства. Поскольку оба этих электромеханических устройства работают существенно медленнее по сравнению с передачей данных с диска или на диск, используя спулинг, можно добиться более эффективного функционирования системы. Во время выполнения программ, когда требуются входные данные для них, образы карт считываются из дискового файла, когда же создаются выходные данные, образы печатаемых строк передаются в дисковый файл. При этом скорость вычислений в отдельных программах не снижается из-за более медленной работы устройств ввода-вывода.

## Мультипрограммирование

Как в простой системе с пакетным режимом, так и в системе пакетной обработки со спулингом центральный процессор последовательно предоставляется для выполнения заданий  $X$ ,  $Y$  и  $Z$ . До начала вычислений по заданию  $Y$  полностью завершаются вычисления по заданию  $X$  и т. д.

Теперь рассмотрим более общий случай, когда подпрограмма ОС выбирает несколько (возможно, три или четыре) заданий из входной очереди заданий на диске, образованной в результате спулинга, и передает их в ОП с целью подготовки для выполнения. Затем одному из этих заданий предоставляется время центрального процессора для начала вычислений (кого-

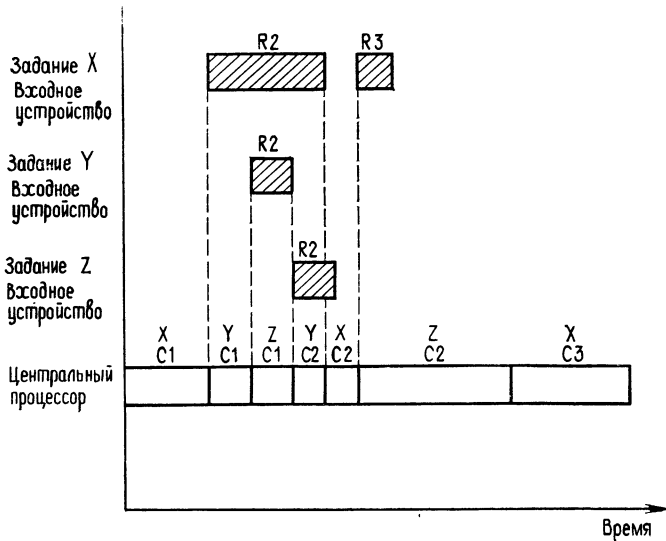


Рис. 8.10. Возможная последовательность действий в системе с мультипрограммированием.

рые могут включать трансляцию, связывание с системными подпрограммами и т. д.). После этого в некоторой точке вычислений задание запрашивает операцию ввода-вывода, приводящую к передаче данных на диск или с диска. Это может быть диск спулинга или другие диски, выделенные пользователю для его файлов. Операционная система инициирует эту передачу и затем предоставляет центральный процессор одному из других заданий в ОП.

Эта ситуация, заключающаяся в том, что в каждый момент времени в ОП имеется несколько заданий, способствует уменьшению времени простоя центрального процессора во время обменов с диском (или любого другого типа независимых операций ввода-вывода). Центральный процессор в такой ситуации может с большей эффективностью использоваться для вычислений по программам пользователей. Этот метод называется методом **мультипрограммирования**. Ясно, что для него требуются приемлемые процедуры ОС для достижения желаемой степени эффективности. Существует опасность, что требуемые программы ОС будут занимать слишком много времени центрального процессора. Однако системы с режимом мультипрограммирования оправдали себя на практике и непрерывно совершенствуются.

В качестве простого примера работы системы в режиме мультипрограммирования снова рассмотрим шаги заданий X,

$Y$  и  $Z$ , приведенных на рис. 8.9, *а*. Предположим, что все шаги  $R1$  соответствуют загрузке программы в ОП с диска спулинга ввода. Относительные затраты времени на них могут быть различными, но теперь это несущественно. Далее пусть остальные шаги считывания  $R2$  и  $R3$  задания  $X$  относятся к входным данным с диска спулинга, но шаги  $R2$  заданий  $X$  и  $Z$  относятся к вводу данных из двух других вспомогательных устройств памяти, которые могут взаимодействовать одновременно через другие каналы с диском спулинга. Как и прежде, логическая последовательность событий для каждого задания должна протекать так, как показано на рис. 8.9, *а*. Предположим, что выполнены все шаги  $R1$ . Тогда любой из шагов  $C1$  может инициироваться в центральном процессоре операционной системы. На рис. 8.10 показана возможная последовательность действий, начинающаяся с предоставления времени центрального процессора шагу  $C1$  задания  $X$ . После того как заканчивается шаг  $C1$  задания  $X$ , начинается выполнение шага  $R2$  задания  $X$  и одновременно с ним — шага  $C1$  задания  $Y$ . Остальные шаги выполняются в соответствии с диаграммой. В данном примере центральный процессор все время занят.

### **Управление операционной системой режимом мультипрограммирования**

В предыдущих разделах было показано, как спулинг и мультипрограммирование могут использоваться для достижения высокой производительности системы при обработке заданий. Были приведены примеры, иллюстрирующие способы совмещения шагов ввода и вывода данных и вычислений, относящихся к различным заданиям. Такое совмещение шагов увеличивает загрузку компонент вычислительной системы (устройств ввода-вывода, каналов, центрального процессора) за счет увеличения параллелизма в их работе.

Цель настоящего раздела состоит в рассмотрении некоторых аспектов подпрограмм ОС, которые являются важными для планирования, инициирования и управления параллельной работой компонент системы. Как правило, в системе с мультипрограммированием на различных стадиях обработки находится несколько заданий. Группировка заданий осуществляется с помощью подпрограммы планирования ОС. Это действие называется **долгосрочным планированием**; оно выполняется на основе информации об управлении заданием, которая указывается пользователем в начале задания. Подобная информация обычно объединяется в виде **списка заданий** во время входного спулинга. Пользователь объявляет ресурсы, требуемые для выполнения задания, при помощи управляющих операторов задания,

которые обычно содержат оценки необходимого объема ОП, времени центрального процессора, а также требования в отношении внешних устройств. Для заданий, которые оперируют файлами данных на магнитных лентах или дисках, следует, кроме того, указать необходимый объем памяти на конкретном запоминающем устройстве (ленте или диске).

Рассмотрев требования к ресурсам заданий из списка заданий, программа планирования ОС может сгруппировать задания для мультипрограммирования, с тем чтобы системные ресурсы использовались наиболее эффективным образом. Задания, которые включают много операций ввода-вывода, могут быть сгруппированы с заданиями, требующими большой нагрузки центрального процессора, так что можно выработать сбалансированные требования ко всем компонентам системы. Существует один аспект планирования выполнения заданий, который не связан непосредственно с эффективностью системы. Часто пользователи могут запросить обслуживание с высоким приоритетом за дополнительную плату. Это приводит к введению классов приоритетов заданий, основанных на скорости их обслуживания. Приоритет задания также должен учитываться программой планирования, если он не задан вручную в момент представления заданий в процессе входного спулинга.

После того как задания сгруппированы для мультипрограммирования, они должны загружаться в ОП. Для этого заданиям должны быть выделены определенные участки ОП. (Эту функцию выполняет **программа управления памятью**.) Процедуры ОС для указанного процесса размещения заданий могут быть достаточно сложными и в дальнейшем не рассматриваются. После того как несколько заданий в ОП подготовлены для обработки, необходимо иметь подпрограмму ОС, осуществляющую **кратковременное планирование**. Эта подпрограмма предназначена для разрешения конфликтов между заданиями, размещенными в ОП, в отношении распределения времени центрального процессора, доступа к каналам и т. д.

При рассмотрении ОС удобно иметь общее название для любой задачи, находящейся под управлением ОС, включая любые подпрограммы, которые могут быть выполнены как часть запланированного задания. Подобные задачи принято называть **процессами**. Вообще процесс — это основная единица работы, выполняемая под управлением ОС. Представление процесса включает информацию о текущем содержимом слова состояния программы (PSW), о содержимом регистров центрального процессора и т. д. по мере выполнения программы, связанной с процессом. Необходимо иметь достаточно информации для возобновления программы и ее правильного выполнения, если она была прервана. Приведем несколько примеров этого неформального



представления процесса. Рассмотрим задание пользователя, содержащее программу PROG. Во время выполнения программы PROG она запрашивает операцию ввода данных с диска через канал ввода-вывода А. При выполнении операции ввода программа PROG временно приостанавливается и время центрального процессора предоставляется некоторому другому заданию. После того как передача входных данных закончена, PROG готова к возобновлению и ей повторно отводится время центрального процессора. После завершения вычислений подается запрос на печать некоторых выходных данных. Канал ввода-вывода В передает данные в дисковый файл, который накапливает все выводимые на печать данные, связанные с этим заданием. Позже выходной спулинг отпечатает этот файл, используя отдельную операцию. С этим заданием связаны следующие три процесса:

- $P_1$  программа PROG,
- $P_2$  входная подпрограмма, выполняемая каналом А,
- $P_3$  выходная подпрограмма, выполняемая каналом В.

Отдельные входные и выходные процессы, такие, как  $P_2$  и  $P_3$ , относительно простые с точки зрения управления ими операционной системой. После инициализации они обычно выполняются непрерывно до конца благодаря особенности работы канала. С другой стороны, такие процессы, как  $P_1$ , могут осуществляться сложным образом. Они могут многократно блокироваться из-за обращений к операциям ввода-вывода или прерывания другими процессами с более высоким приоритетом. В результате процесс может переходить в одно из следующих состояний: «готовность для выполнения», «выполнение», «заблокирован».

Диаграмма **состояний** представляет собой полезное средство графического отображения возможных переходов из одного состояния в другое, которые могут иметь место при выполнении данного процесса. На рис. 8.11 показана общая диаграмма состояний процесса. Стрелки между состояниями указывают на возможность соответствующих переходов. Одна из функций ОС заключается в **управлении процессами**, т. е. в управлении действиями, которые следуют из модели диаграммы состояний. Например, рассмотрим ситуацию, когда во время выполнения программы X программа Y с более высоким приоритетом заблокирована в ожидании внешнего прерывания. Когда это прерывание произойдет, управление будет передано операционной системе и программа Y перейдет в состояние разблокирования. Поскольку программа Y имеет более высокий приоритет, чем программа X, то диспетчер ОС прервет выполнение программы X и возобновит выполнение программы Y,

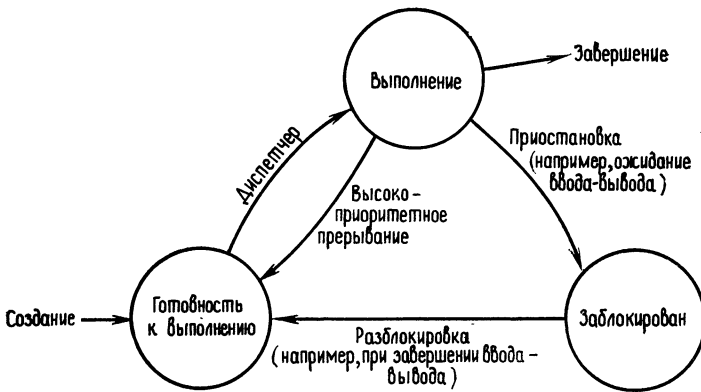


Рис 8.11. Диаграмма состояний процесса

Теперь читатель, возможно, обратит внимание на то, что в системе с мультипрограммированием пользователь не занимается составлением машинных команд ввода-вывода и/или команд для каналов и не включает их в свою программу. Он также не рассматривает обслуживание прерываний, возникающих в результате выполнения его запросов на ввод-вывод данных. Пользователь просто выдает запрос операционной системе на выполнение операции ввода-вывода и возврат управления после завершения этой операции. Подпрограммы ввода-вывода данных операционной системы осуществляют их фактическую передачу. Пользователь не составляет подпрограммы для фактического ввода-вывода данных по ряду причин. Прежде всего большинство пользователей предпочитают выражать свои логические входные и выходные данные посредством только простых операторов языка высокого уровня. Пользователи такого типа заботятся лишь о считывании информации с карт и выводе результатов на построочно-печатающее устройство. Эти пользователи полностью изолированы от механизмов входного и выходного спулинга и мультипрограммирования. Даже те пользователи, которые могли бы программировать некоторые из машинно-зависимых функций, все же нуждаются в центральном управлении посредством ОС, чтобы иметь гарантию того, что 1) данные или программы разных пользователей не будут влиять друг на друга, 2) ни один пользователь не сможет в отдельности монополизировать ресурсы системы и снизить ее эффективность. Таким образом, можно утверждать, что подпрограммы ОС должны выполнять все операции ввода-вывода и управлять использованием разделяемых ресурсов для защиты отдельных заданий и поддержания высокой производительности системы.

Следует отметить, что центральные процессоры некоторых ЭВМ имеют как супервизорный, так и пользовательский режимы. Одно из отличий супервизорного режима от режима пользователей заключается в том, что некоторые команды могут быть выполнены только тогда, когда центральный процессор находится в супервизорном режиме. К числу таких команд относятся команды, модифицирующие PSW<sup>1)</sup>. Данное обстоятельство помогает обеспечить защиту пользовательских программ и данных. Когда центральный процессор находится в пользовательском режиме, его работу может прервать любое внешнее устройство, вызывая переход к вспомогательной подпрограмме прерывания ОС, выполняемой в супервизорном режиме. Когда центральный процессор находится в супервизорном режиме, его работу могут прервать лишь наиболее критические по времени срабатывания устройства ввода-вывода.

Здесь необходимо более четко выделить роль приоритетов прерывания устройств ввода-вывода в противоположность более общему понятию приоритетов процесса-задания пользователя. Приоритет задания при планировании распределения разделяемых ресурсов обычно определяется требованиями, заявляемыми на управляющих картах задания, а также отношением пользователя к вопросу более быстрого обслуживания за дополнительную плату. Допустим, что ОС определяет приоритет запросов на выполнение заданий на основе других характеристик заданий. Можно предложить следующую стратегию: высокий приоритет получают запросы на обработку заданий на центральном процессоре, требующие большого числа операций ввода-вывода данных; соответственно высокий приоритет получают также запросы на ввод-вывод от заданий, требующих большой загрузки центрального процессора. Цель этой стратегии состоит в обеспечении такого способа обработки заданий, при котором эти задания будут максимально подготовленными к использованию ресурсов, от которых они зависят в наибольшей степени. Таким образом, можно получить устойчивый спрос на все системные ресурсы, в результате чего обеспечивается максимальное использование компонент системы и высокая производительность обработки заданий. В некоторых случаях, когда это будет продемонстрировано в следующем разделе на примере интерактивных систем, приоритеты при планировании определяются посредством учета срочности запросов на выполнение отдельных процессов и система уже не будет иметь ту степень свободы при планировании процессов, которая характерна для нее, когда единственным критерием полезного действия является производительность обработки заданий.

---

<sup>1)</sup> Команды, выполняемые только в супервизорном режиме, называются привилегированными командами. — *Прим. ред.*

Теперь вернемся к рассмотрению приоритетов прерывания устройств ввода-вывода, связанных с запросами прерывания работы аппаратуры, которые порождаются внешними устройствами. Эти приоритеты обычно не зависят от приоритетов планирования выполнения процессов заданий пользователей. Приоритеты прерывания устройств определяются по степени срочности ответа обслуживающих подпрограмм, являющихся частью операционной системы. Эти запросы на ответы внешних устройств зависят от их характеристик и не зависят от задания пользователя.

### **Управление операционной системой интерактивным режимом вычислений**

Интерактивные вычисления стали весьма распространенным методом использования ЭВМ. Общая идея этого метода заключается в том, что пользователи составляют свои программы, используя телетайп или внешнее устройство с электронно-лучевой трубкой в **режиме реального времени**. Это означает, что устройство ввода-вывода пользователя находится под управлением центрального процессора и по мере ввода знаки с клавиатуры устройства непосредственно передаются в вычислительную систему. В некоторых случаях простые вычисления выполняются построчно по мере ввода строк. В других случаях пользователь может назвать имя подпрограммы или функции и напечатать его, а затем выдать запрос на выполнение. Для этой цели разработаны различные языки программирования; наиболее известными являются APL и BASIC. Ввиду того что для обработки знаков и строк, введенных пользователем, требуется относительно низкая скорость, обычная большая ЭВМ может обслуживать от 25 до 50 абонентских терминалов, производя вычисления в небольшом объеме.

При пакетной обработке главная цель состоит в достижении высокой производительности системы. В этом случае время, требуемое для завершения отдельного задания, является существенным фактором. С другой стороны, главная цель интерактивной вычислительной системы заключается в обеспечении быстрого ответа отдельным пользователям. Это требование накладывает более жесткое ограничение на планирование работы центрального процессора по сравнению с общей схемой мультипрограммирования. Существенно, что, после того как пользователь введет команду в терминал, послав запрос в ЭВМ на выполнение некоторого вычисления и получение отпечатанного результата, ЭВМ ответит на этот запрос через несколько секунд. Величину этой задержки ответа обозначим через  $T$ .

Рассмотрим большую вычислительную систему с одним центральным процессором, выполняющим вычисления для терминалов, число которых может дойти до 50. Для обеспечения требуемого времени ответа всем пользователям можно применить метод планирования, который называется **методом квантования времени** или **циклического обслуживания**. При использовании этого метода центральный процессор предоставляется для обслуживания каждого терминала по очереди в течение короткого интервала времени. Для удовлетворения требований к времени ответа длина этих интервалов должна быть ограничена сверху. Этот верхний предел называется **квантом** и обозначается через  $t$ . Для прерывания процесса пользователя, требующего для своего выполнения более чем  $t$  секунд, используется интервальный таймер (датчик временных интервалов). При прерывании управление передается ОС. Затем ОС выделяет квант времени ЦП следующему терминалу. Если не учитывать временные затраты на работу ОС, то значение  $t$  можно положить равным  $T/n$ , где  $n$  — число работающих терминалов. Рассмотрим конкретный пример, в котором допустимая задержка  $T$  не превосходит двух секунд и используются все 50 терминалов. Тогда через каждые 2 с каждому пользователю предоставляется 40 мс на вычисления. На ЭВМ, которая выбирает и исполняет команду в среднем за 4 мкс, 40 мс соответствуют выполнению 10 000 команд. Этого должно быть достаточно для полного исполнения большинства подпрограмм, используемых в интерактивных вычислениях.

Приведенный выше численный пример позволяет получить лишь оценку времени вычислений, предоставляемого любому пользователю через каждые  $T$  секунд. Многие детали в нем опущены. Параметр  $T$  был определен как интервал времени от момента подачи запроса пользователя на вычисления до момента получения ответа от ЭВМ. Предполагалось, что все время работы центрального процессора предоставляется пользователю для его вычислений. На самом деле это условие не соблюдается. Время центрального процессора может потребоваться также для передачи отдельных символов с терминалов в буферы памяти. Другой способ состоит в использовании отдельного процессора для накопления входных символов и размещения их в его запоминающем устройстве. Всякий раз, когда в терминале формируется полная строка, она передается в устройство памяти главной ЭВМ. Однако сначала рассмотрим случай, когда главный центральный процессор выполняет все задания, связанные с накоплением и буферизацией символов. Символы с конкретного терминала накапливаются во входном буфере, выделенном этому терминалу. ЭВМ может также передать обратно каждый символ для печати на терминале (его эхо) сразу

же после его получения. Эхо должно быть возвращено пользователю как бы мгновенно, что на практике означает доли секунды.

Буферизация входных строк и соответствующее им обратное эхо отдельных символов для всех интерактивных терминалов должны рассматриваться как независимый процесс, осуществляемый ОС. То же самое относится к буферизации выходных строк, которые являются реакцией (ответами) ЭВМ, и их последующей печати. Для вызова процессов ОС, связанных со считыванием и печатью символов, можно использовать прерывания. С другой стороны, центральный процессор может упорядочить передачу отдельных символов с терминалов. Ясно, что это должно осуществляться со значительно большей скоростью, чем предоставление кванта с помощью планировщика циклического обслуживания.

На считывание и печать отдельных символов требуется дополнительное время, из-за которого уменьшается время центрального процессора, предоставляемое для вычислений. Таким образом, фактическая длина кванта меньше значения  $t$ , определенного ранее. Однако, по-видимому, через каждые  $T$  секунд не со всех терминалов поступают запросы на вычисления. Это означает, что через каждые  $T$  секунд в среднем каждому терминалу может отводиться время вычисления, большее, чем один квант.

Поучительно получить грубую оценку времени, дополнительно требуемого для обработки отдельных символов. С этой целью удобно описать «типичную» работу (сеанс)<sup>1)</sup> «среднего» пользователя на терминале. Допустим, что студент хочет выполнить вычисления, связанные с определением напряжений в электрической схеме. Пусть основная часть вычислений включает решение системы четырех линейных уравнений. Для решения такой задачи имеется системная подпрограмма. Студент должен ввести в ЭВМ параметры подпрограммы и команду обращения к ней. Каждый параметр представляется в формате десятичного числа с плавающей точкой не более чем 10 символами. Поскольку максимальное число требуемых параметров равно 16, студент отпечатает самое большее 160 символов при вводе данных о задаче. Если предположить, что для инициирования задания и вызова подпрограммы требуются еще 40 символов, то студент во время своей работы на терминале должен отпечатать около 200 символов. Ответы ЭВМ будут состоять из четырех значений, каждое из которых имеет примерно

---

<sup>1)</sup> Сеанс (session) — непрерывный период времени, в течение которого пользователь активно взаимодействует с ЭВМ в интерактивном режиме. — *Прим. перев.*

10 символов; различные ответы во время инициирования и при завершении работы могут насчитывать от 50 до 60 символов. Во время этого сеанса (взаимодействия пользователя с ЭВМ) ЭВМ отпечатает около 100 знаков. Таким образом, общее число печатаемых символов составит 300. Пусть продолжительность всего сеанса 5 мин. Следовательно, каждый терминал осуществляет загрузку 60 символов в минуту. Если с ЭВМ взаимодействуют 50 терминалов, то общая оценка скорости обработки символов равна 3000 символов в минуту.

Теперь определим время, требуемое для обработки одного символа. Предположим, что рассматриваемая система обладает средствами прерывания. После получения запроса на прерывание с терминала текущее состояние центрального процессора должно быть сохранено. Допустим, что эта процедура содержит 10 команд. Обслуживание прерывания включает идентификацию терминала, передачу символа в соответствующий входной буфер, установку указателя буфера и передачу символа обратно в печатающее устройство терминала. Это может потребовать еще 20 команд. Наконец, для восстановления состояния центрального процессора и возвращения к прерванной программе потребуются примерно 10 команд. Поэтому при грубой оценке для обработки одиночного символа нужно выполнить 40 команд. Если допустить, что на выполнение каждой команды нужно 4 мкс, то время на обработку всех символов составит  $3000 \times 40 \times 4 = 480\,000$  мкс/мин, а затраты времени на обработку символов — лишь 0,8 % общего времени работы.

Выше предполагалось, что для выполнения всех процессов, требуемых при осуществлении интерактивных вычислений, включая обработку отдельных символов, используется лишь один центральный процессор. Поскольку последняя операция не зависит от остальных, она может быть передана отдельному процессору предварительной обработки данных. Этот процессор собирает символы, поступающие из всех терминалов, и пересылает их в виде более крупных блоков главному процессору. Главный процессор будет прерывать свою работу только при передаче блоков, а не при запросах на прерывание от отдельных терминалов. Хотя расчеты, приведенные в данном примере, показали, что на эти прерывания тратится лишь около одного процента времени работы главного центрального процессора, рассмотренная ситуация была весьма упрощена. Предполагалось, что все терминалы — медленные телетайпные устройства, и трансляция символов не производится. Процессоры предварительной обработки данных используются для выполнения ряда других задач, связанных с буферизацией ввода и вывода символов. Возможны различные типы индивидуальных терминалов, каждый из которых имеет собственные протоколы пере-

дачи данных. Идентификация отдельных терминалов и передача символов в обоих направлениях могут оказаться более сложными по сравнению с рассмотренными способами. Кроме того, коды множества символов могут различаться при переходе от одного типа терминала к другому. Тогда с помощью процессора предварительной обработки данных можно транслировать все коды в некоторый стандартный формат до их передачи в главную ЭВМ.

Отметим еще один практический вопрос относительно использования ЭВМ, которые обеспечивают интерактивное обслуживание. Если при загрузке интерактивных вычислений занимает только часть времени работы центрального процессора, вполне допустимо перевести ЭВМ в режим неинтерактивной пакетной обработки в свободные интервалы времени. В подобных случаях режиму интерактивной обработки должен предоставляться больший приоритет. Такая пакетная обработка осуществляется на уровне наиболее низкого приоритета и часто называется фоновым вычислением.



# УСТРОЙСТВА ВВОДА, ВЫВОДА И ДОПОЛНИТЕЛЬНОЙ ПАМЯТИ

*К. Гизер*

## 9.1. ВВЕДЕНИЕ

Функцией вычислительной системы является обработка информации. Вычислительная система должна обладать способностью обмениваться информацией с внешним миром. Чтобы обработать информацию, ее необходимо ввести в систему, а результаты обработки вывести во внешний мир. Окружение ЭВМ определяет тип устройств ввода-вывода, обеспечивающих требуемый обмен информацией. В большинстве известных систем предусмотрена возможность обмена информацией ЭВМ с людьми, которые подготавливают задачу для ее решения, обеспечивают исходные данные и предполагают использовать ответы ЭВМ. Иногда ввод может быть произведен непосредственно с прибора. Измерения в сложных физических экспериментах могут быть произведены автоматически с помощью ЭВМ. В таких случаях вводимая информация представляет собой цифровые показания различных приборов, которые используются для управления экспериментом. В некоторых применениях устройства как ввода, так и вывода могут быть непосредственно связаны с механической или электрической аппаратурой. Например, бортовая ЭВМ может применяться для определения положения самолета в любой момент (используя ввод с гироскопа или другого инерциального прибора), которое сравнивается с ожидаемым положением, вычисленным на ЭВМ; затем формируются корректирующие сигналы для управления механизмами самолета. ЭВМ, которые используются в режиме реального времени для работы с контрольно-измерительными приборами и для управления процессами, обычно меньше и более специализированы по сравнению с ЭВМ общего назначения. В этой главе будут рассмотрены устройства ввода и вывода, которые являются обычными для универсальных систем. В принципе они не отличаются от подобных устройств, используемых в ЭВМ реального масштаба времени.

---

Adapted from Computer Organization and Programming, by C. William Gear. Copyright © 1980, 1974, 1969, Used by permission of McGraw-Hill, Inc. All rights reserved.

ЭВМ общего назначения или обменивается информацией с пользователем или осуществляет обмен с носителем информации. Устройство, производящее обмен информацией с пользователем, называется устройством **ввода-вывода**, тогда как устройство, осуществляющее обмен с носителем информации, называется **вспомогательной**, или **вторичной, памятью**, так как информация сохраняет машинное представление. Вспомогательную память называют также **запасной** памятью. Вспомогательная память может также использоваться для передачи данных в другую ЭВМ немедленно или спустя некоторое время. Непосредственная передача данных может быть осуществлена как прямой обмен данными между двумя ЭВМ. При прямом обмене данными каждая ЭВМ воспринимается другой ЭВМ как устройство дополнительной памяти. Обмен данными этого типа часто производится между малыми и большой ЭВМ. Задержанные передачи выполняются следующим образом: сначала информация запоминается на машинном носителе (например, на магнитной ленте), а затем передается с этого носителя в другую машину. Важной особенностью этой памяти является возможность возвращения информации в записавшую ее машину. Это обеспечивает длительное хранение большого количества данных без использования дорогой основной памяти. Будет показано, что некоторые устройства имеют возможность хранить информацию на носителе, который может передаваться от одной ЭВМ к другой. Это может быть положено в основу классификации вспомогательной памяти. Устройства ввода-вывода могут быть отнесены к упомянутой категории. Устройства различаются также по скорости и емкости; это положено в основу используемой ниже классификации. В данной главе будут рассмотрены запоминающие устройства долговременного хранения, которые часто используются для хранения информации в течение длительного времени, и устройства со средним сроком хранения, которые часто применяются для передачи информации от одной короткой программы к другой в качестве дополнительной памяти во время однократного выполнения программы или для обеспечения быстрого доступа к часто используемым системным программам. Вообще высокоскоростные устройства имеют меньшую емкость и в меньшей степени способны к смене носителя информации.

## 9.2. УСТРОЙСТВА ВВОДА-ВЫВОДА

Устройства ввода-вывода бывают двух основных типов: устройства, выдающие твердую копию, и устройства, не выдающие твердой копии. Последние обеспечивают пользователя информацией непосредственно в несохраняемой форме или счи-

тывают вводимую информацию непосредственно как результат некоторых действий пользователя, например нажатия клавиши на клавиатуре пульта.

### Оперативные терминалы

Одним из наиболее простых устройств ввода-вывода в настоящее время является оперативный терминал. Он имеет клавиатуру, на которой пользователь может набрать буквенные и числовые данные, а также устройство ввода. Устройство вывода может быть или печатающее устройство, подобное пишущей машинке, или электронно-лучевая трубка (ЭЛТ) буквенно-цифрового дисплея. Последний может отображать ряд строк символов, обычно от 20 до 24 строк по 60—80 символов в строке. При использовании ЭЛТ ЭВМ обычно отображает на ней последние 20 или 24 строки, причем она работает в **скользящем режиме**, так что каждая новая строка выводится в нижнюю строку экрана, остальные строки экрана сдвигаются на одну строку вверх, а верхняя строка исчезает с экрана.

Скорость набора символов на клавиатуре ограничена (максимум 10—15 символов в секунду в очень короткие интервалы времени и в среднем до 8 символов в секунду для очень хорошей машинистки). Скорость вывода определяется пропускной способностью линии, связывающей терминал с ЭВМ. Для печатающей машинки эта скорость обычно составляет 15—30 символов в секунду, хотя некоторые печатающие устройства работают быстрее. Скорость вывода на ЭЛТ много больше и достигает 120—960 символов в секунду. Пользователи этих терминалов могут установить требуемую скорость в бодах (бит в секунду) переключателем. Так как большинство передающих систем передают 10 бит, чтобы послать один 8-битовый символ ASCII, то скорость в бодах в 10 раз больше скорости передачи в символах. Многие терминалы соединяются с ЭВМ телефонной линией. В этом случае скорость в бодах ограничена величиной примерно 1200.

Большинство терминалов связаны с ЭВМ двумя независимыми каналами: один соединяет клавиатуру с ЭВМ для ввода, а другой соединяет ЭВМ с устройством вывода терминала. (Даже при использовании одной телефонной линии обеспечиваются две независимые линии связи. Они разделяют одну телефонную линию.) Режим с использованием двух независимых линий связи называется **дуплексным режимом передачи**; он позволяет одновременно осуществлять передачу в обоих направлениях. Хотя во многих случаях пользователю кажется, что каждый символ, переданный с клавиатуры, немедленно печатается или отображается на устройстве вывода, фактически

этот символ посылается в ЭВМ и возвращается на терминал. Получается так называемое **эхо**. В некоторых системах вывод на печать может осуществляться быстрее, чем ЭВМ сгенерирует ответ, так что эхо в этом случае может восприниматься как сигнал предупреждения. В других системах используется **полудуплексный** режим, когда имеется только одна передающая линия. В таких системах пользователь должен быть осторожен и не нажимать на клавиши, пока ЭВМ передает информацию обратно на терминал, так как если в линию связи в одно и тоже время поступят два сигнала, то это приведет к конфликту между ними и информация будет потеряна. Существует также система, которая называется **симплексной**. В этой системе лишь одна линия связи, и она может быть одновременно использована только в одном направлении. Для связи между терминалом и ЭВМ необходимы некоторые виды **протоколов**, чтобы было определено, когда каждое из связанных между собой устройств имеет право на передачу, и обеспечивалась гарантия правильной работы. Симплексные системы редко используются для терминалов.

### Построчно-печатающее устройство

Печатающим устройством вывода в терминалах является устройство посимвольной печати. Это ограничивает его скорость, так как трудно напечатать отдельный символ быстрее, чем за 4 мс. **Построчно-печатающее устройство** печатает за такт одну строку информации. ЭВМ передает целую строку информации в устройство построчной печати, где она затем печатается. Для каждой печатной позиции строки имеется печатающий механизм; ширина строки в распространенных печатающих устройствах может содержать от 80 до 150 символов. Так как в таких устройствах параллельно используется группа печатающих механизмов, при механических способах печати можно достичь скорости от 1000 до 2000 строк в минуту. Немеханические способы печати (например, фотографические) могут обеспечить еще более высокую скорость печати (до 30 000 строк в минуту).

**Графопостроители и графические дисплеи.** Хотя печатающие устройства вывода очень удобны для многих целей, часто пользователь нуждается в графическом представлении информации, чтобы можно было понять ее смысл. Это может быть график зависимости одной переменной от другой или сложное изображение, предоставляющее многомерный объект. Например, трехмерные изображения часто бывают представлены двумерными контурными картами, где линии являются траекториями постоянства третьего измерения (на метеорологической

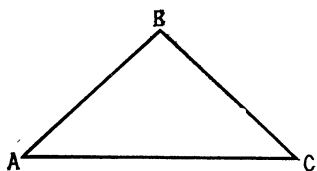


Рис. 9.1. Построение чертежа треугольника.

Движение пера начать от А.  
Перо опустить.  
Вверх, направо (100 шагов).  
Вниз, направо (100 шагов).  
Налево (200 шагов).  
Перо поднять.

карте нанесены контуры постоянного давления, которые указывают области высокого и низкого давления).

Чтобы построить графики вручную по числовым выходным данным ЭВМ, человек должен затратить на это много времени, если объем данных большой. Поэтому для использования в ЭВМ были разработаны устройства прямого графического вывода. Существуют два основных типа устройств вывода: одни устройства позволяют пользователю ставить точку в любом

месте выводной страницы, а другие дают возможность пользователю перемещать **перо** в любом направлении на заданное расстояние, причем перо при этом может быть либо установлено на бумаге и делать пометки, либо не соприкасаться с бумагой и не делать пометок. Слова **перо** и **бумага** не следует понимать буквально, такой вывод может осуществляться на любую среду, которая способна фиксировать след. Для многих устройств эти два основных способа одинаково приемлемы, но удобно рассмотреть их отдельно.

**Инкрементный графопостроитель** является примером реализации второго способа вывода информации. Это устройство с пером и большой лентой бумаги. Перо можно перемещать небольшими шагами в любом из нескольких направлений. В типичном графопостроителе возможно перемещение в любом из восьми направлений, под углом  $45^\circ$  друг к другу, на расстояние  $5/1000$  дюйма вертикально или горизонтально, или на расстояние  $7/1000$  дюйма в любом из четырех направлений под углом  $45^\circ$  друг к другу. Все линии должны быть образованы при помощи этих коротких прямолинейных сегментов. ЭВМ управляет графопостроителем при помощи набора символов, каждый из которых выполняет базовый шаг чертежа. На рис. 9.1 показано использование графопостроителя для черчения равнобедренного прямоугольного треугольника, лежащего на гипотенузе. Библиотека стандартных программ ЭВМ обычно содержит подпрограммы для простого использования графопостроителя. Общий пакет программ для графопостроителя позволяет пользователю перемещать перо, производящее или непроизводящее пометки, из текущего положения в любое заданное положение для того, чтобы «нарисовать» многие из известных кривых, например эллипс, и буквы или цифры. Подпрограммы преобразуют запросы пользователя в последовательность базовых шагов графопостроителя. Время вычерчивания фигуры зависит от линейного расстояния, на

которое перемещается перо. Скорость пера обычно порядка 1—10 дюйм/с.

Устройство вывода с ЭЛТ в основном позволяет пользователю высвечивать точки на экране ЭЛТ, задавая устройству вывода координаты  $X$  и  $Y$ . В таком устройстве линия строится из множества малых точек. Обычно в каждом направлении может быть высвечено около 500—4000 точек и соседние точки настолько близки, что сливаются для глаза. Дисплей на ЭЛТ другого типа обладает встроенными генераторами **векторов** и **символов**, которые позволяют непосредственно воспроизводить прямые линии и стандартные символы. Это уменьшает время ЭВМ, необходимое для генерации изображения, но с точки зрения пользователя не увеличивает гибкость в построении изображения. Подпрограммы используют основные операции для обеспечения тех же возможностей, какие имеются у графопостроителя. Важным дополнением дисплея на электронно-лучевой трубке является **световое перо**. Это фотоэлектрическое устройство, используемое подобно обычному перу, которое можно поместить в любую точку экрана ЭЛТ. Когда под управлением программы световое перо при коротком прикосновении высвечивает на экране светящееся пятно, фотоэлемент генерирует сигнал. Этот сигнал поступает в ЭВМ обычно в форме сигнала прерывания. Таким образом, программист может указать, где находится перо. С помощью соответствующей программы возможно проследить ход светового пера по экрану и запомнить внутри ЭВМ координаты различных точек, указанных пользователем. Таким образом, пользователь может **рисовать** на экране ЭЛТ и использовать световое перо как устройство ввода. Чаще всего с помощью светового пера производится выделение частей изображения.

Для указания точки на экране могут использоваться и другие приборы. Один из них представляет собой специальный планшет, чувствительный к электронному перу, которое держит пользователь. Другим прибором является небольшой блок, который пользователь перемещает по поверхности стола. Датчик измеряет положение руки и передает эту информацию в ЭВМ, которая затем изображает круг или крест на экране в позиции, соответствующей положению руки.

Еще одной формой вывода является ЭЛТ, спаренная с камерой, где прямо на выходе ЭВМ получается микрофильм или микрофиша. В этом случае, кроме команд для вычерчивания линий и изображения текста на экране ЭЛТ, имеются команды, предназначенные для управления фотопленкой или положением негатива фотофиши. Эти устройства имеют очень высокую скорость вывода и другим их достоинством является то, что физические размеры вывода очень малы по сравнению с огромными

распечатками (листингами). (В некоторых высокоскоростных ЭВМ при больших числовых расчетах используется печатающее устройство со скоростью выше 30 000 строк в минуту. В таких печатающих устройствах механизм перемещения бумаги должен за небольшой промежуток времени передвинуть огромную кипу бумаги; действительно, каждую минуту печатается восьмидюймовая пачка бумаги размером  $11 \times 14$  дюймов. Микрофиша с этой информацией может быть унесена в руках. Сомнительно, чтобы это можно было прочесть.)

Следующие два типа устройства ввода-вывода выходят из употребления. Перфокарты еще используются в некоторых приложениях, когда небольшое количество информации получается по почте из многих источников (например, идентифицирующая информация, возвращаемая с платежными счетами). Перфоленты иногда применяются для низкоскоростной записи данных эксперимента и на микроЭВМ чрезвычайно низкой стоимости. Мы рассмотрим эти устройства, потому что они имеют важное историческое значение и понимание принципа записи на перфоленту существенно упрощает понимание работы устройств записи на магнитной ленте.

## Перфокарты

Наиболее распространенными являются перфокарты прямоугольной формы из плотного картона (обычно  $7,5$  дюйма  $\times$   $3,25$  дюйма). На перфокарте могут быть пробиты отверстия на любой из 960 позиций прямоугольного массива позиций размером  $12 \times 80$ . Кроме вышеупомянутых 80-колонных перфокарт, используются карты с 51 или 90 колонками, но 80-колонные перфокарты наиболее широко применяются. Перфокарты использовались для управления машинами в текстильной промышленности задолго до того, как их стали применять в системах обработки данных. Первым существенным применением перфокарт в системах обработки данных является использование их Германом Холлеритом для переписи населения 1890 г. Благодаря этой работе карты стали называться **картами Холлерита** и способ, по которому они пробивались, называется **кодом Холлерита**. Этот код является средством представления буквенно-цифровых данных (т. е. букв и 10 цифр), а также нескольких специальных знаков, таких, как точка, запятая и т. д. Для представления каждого символа используется одна колонка, так что на одной перфокарте может быть пробито до 80 символов. Как показано на рис. 9.2, строки карты нумеруются в следующем порядке: 12, 11, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Символ пропуска, или пробел, представляется в виде отсутствия пробивки, цифры от 0 до 9 пробиваются в соответствующую

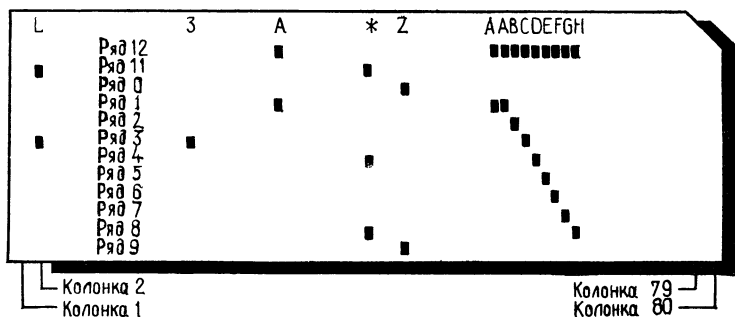


Рис. 9.2. Перфокарта Холлерита.

L — пробивки 11, 3; 3 — пробивка 3; A — пробивки 12, 1; \* — пробивки 11, 4, 8; Z — пробивки 0, 9.

щей строке, буквы имеют две пробивки. Буквы от A до I представляются пробивкой в 12-й строке с цифровой пробивкой от 1 до 9 соответственно, буквы от J до R — пробивкой в 11-й строке и от 1 до 9 соответственно и буквы от S до Z — пробивкой в 0-й строке и от 2 до 9 соответственно. Другие символы представляются другими комбинациями пробивок.

Когда карта, пробитая в этом формате, считывается в машину, ее содержимое преобразуется в строку символов в памяти в соответствующем коде. Если используется стандартный 48-символьный набор, то для представления 48 символов необходимо 6 бит. Поэтому можно использовать один из 6-битовых кодов, например двоично-кодированный десятичный код. Это преобразование обычно производится аппаратурно, так что программист не имеет дела с кодом Холлерита. Во многих современных ЭВМ используется 8-битовый код, позволяющий кодировать до 256 символов. Наиболее распространенными 8-битовыми кодами являются код ASCII (американский стандартный код для обмена информацией) и EBCDIC (расширенный двоично-десятичный код для обмена информацией). Недавно появился внутренний стандарт фирмы IBM.

Каждая колонка перфокарты может содержать до 12 пробивок для  $2^{12} = 4096$  различных комбинаций. Однако при 6- или 8-битовом коде используются весьма немногие комбинации. В некоторых системах для преодоления этой неэффективности вводятся **двоичные карты**. Двоичная карта может быть пробита в любой позиции, так что каждая колонка содержит 12 бит информации. Если двоичная карта считывается в ЭВМ, каждая колонка непосредственно отображается в 12-битовую память, и наоборот. Так как перфокарты, пробитые таким образом, не могут быть прочитаны на большинстве устройств перфокарточного ввода, они используются исключительно для хранения информации, которая должна считываться обратно



в ЭВМ на более позднем этапе. Довольно часто, например на двоичных картах, производится хранение программ после их трансляции в ЭВМ (во избежание их ретрансляции). Если в ЭВМ используется 8-битовый код, то существует отображение 8-битового байта в комбинацию пробивок в отдельной колонке. Использование 8 из 12 бит информации неэффективно, но и не слишком плохо. Например, в IBM 370 для хранения двоичной информации на картах применяется код EBCDIC, а не специальный набор команд для управления двоичными картами. При использовании 6-битового кода в каждой колонке двоичной карты можно разместить по два 6-битовых символа. Однако в настоящее время двоичная информация редко хранится на перфокартах, так что эта неэффективность не критична.

Перфокарты могут считываться со скоростью от 1000 до 2000 в минуту и пробиваться со скоростью от 200 до 400 в минуту. Вследствие того что при считывании перфокарт возможно появление ошибок, считывающее устройство для контроля ошибок считывания обычно считывает каждую карту дважды, а перфорирующее устройство для выполнения подобной проверки также считывает результат перфорации.

### Перфолента

Перфолента очень похожа на перфокарту тем, что это рулон жесткой бумаги, на котором информация представляется наличием или отсутствием пробитых отверстий. Ее отличие от перфокарты состоит в том, что она представляет собой непрерывную узкую полосу произвольной длины, тогда как перфокарты имеют фиксированные размеры. Обычно используются ленты трех типов по ширине, поперек которой имеются пять, семь и восемь возможных позиций информационных пробивок. Каждая такая позиция называется **дорожкой**. На рис. 9.3 показана однодюймовая 8-дорожечная лента. Заметим, что в дополнение к восьми информационным дорожкам имеется девятая, вдоль которой в каждой позиции пробиты отверстия<sup>1)</sup>. Такие отверстия делаются для того, чтобы можно было установить место позиции вдоль ленты, где должна появляться информация. Без таких сплошных пробивок ряд символов, соответствующих непробитым отверстиям, воспринимался бы как непробитая лента (пробел). Тогда было бы невозможно сказать, сколько символов находится в такой строке, не измеряя длины ленты; процедура была бы сопряжена со многими ошибками. Цепочка отверстий используется также для уста-

<sup>1)</sup> Эта дорожка называется синхронизирующей, или ведущей; отверстия на ней имеют меньший диаметр, чем «кодовые» отверстия — *Прим. ред.*

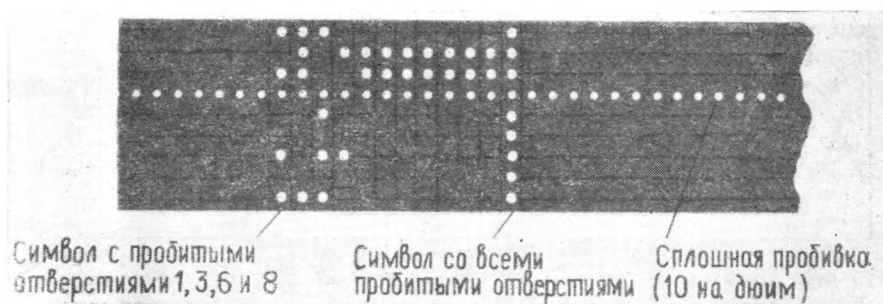


Рис. 9.3. Перфолента.

новки цепной шестерни на более медленном устройстве чтения. Вследствие того что при считывании перфоленты возможны ошибки, довольно часто одна из информационных дорожек используется в качестве бита контроля четности для других информационных дорожек. Перфолента может считываться со скоростью от 1000 до 2000 символов в секунду и пробиваться со скоростью 300 символов в секунду, хотя самые дешевые устройства работают со скоростью только 10 символов в секунду в одном направлении.

Для специальных целей разработано много других видов устройств ввода и вывода. Оптические сканирующие устройства могут **считывать** группы типовых символов и некоторые из них способны считывать символы, написанные от руки. Уже имеется трехмерное перо, которое позволяет ЭВМ фиксировать след руки пользователя в трехмерном пространстве; преобразователи аналогового ввода и вывода не только считывают информацию непосредственно с измерительной аппаратуры экспериментальных установок, но и непосредственно управляют экспериментом. Возможен голосовой вывод, осуществляемый группами речевых записывающих устройств, подобных магнитофону, или подачей сигналов на речевой генератор, который непосредственно создает основные звуки. В порядке эксперимента уже осуществлен ограниченный акустический ввод, использующий ЭВМ для анализа речевых команд.

### 9.3. УСТРОЙСТВА ДОЛГОВРЕМЕННОЙ ПАМЯТИ И ПРОМЕЖУТОЧНОГО ВВОДА-ВЫВОДА

При решении многократно выполняемых задач, в которых используется очень большое количество данных, требуются устройства долговременной памяти. К этому классу задач относятся экономические задачи, например ведение записей

счетов покупателей, и научные задачи, такие, как сравнение данных, полученных во время определенного физического эксперимента, с данными предыдущих экспериментов. Для подобных задач характерно наличие очень большого количества данных (типичная проектируемая система для задач этого типа должна использовать 100 млн. символов) и то, что эти данные изменяются незначительно (изменяются только некоторые счета покупателей или к старому файлу добавляются некоторые дополнительные экспериментальные данные). Все это приводит к необходимости в запоминающем устройстве, имеющем фактически неограниченную общую емкость и обеспечивающем последовательный доступ. Так как электроника, связанная с запоминающими устройствами, является дорогостоящей частью устройства, желательно, чтобы средой запоминания был такой достаточно простой носитель, который можно удалять, хранить и вновь устанавливать. Перфокарты в некоторой степени являются примером долговременного запоминающего устройства, так как в принципе обеспечена возможность читать и перфори́ровать произвольное количество карт. Однако содержимое карты не может быть изменено (отверстия не могут быть забиты) и карты имеют относительно высокую стоимость (два цента за карту). К тому же перфокарточное оборудование работает с невысокой скоростью. Тем не менее имеется ряд других устройств, лишенных этих недостатков, в частности магнитные ленты и комплекты дисков, а также их разновидности. Все это различные формы знакомого магнитофона, который используется для хранения звуков. В этом разделе будут рассмотрены принципы действия ряда таких устройств. Этим не исчерпывается перечень устройств. Хотя и появляются сведения о некоторых новых устройствах, принципы их действия остаются теми же; скорость и емкость могут увеличиваться, а стоимость уменьшаться.

### **Магнитная лента**

Память на магнитной ленте основана на принципах звукозаписи на ленте. Средой хранения является длинная полоса ленты фиксированной ширины. Обычно используется лента шириной 0,5 и 1 дюйм и длиной около 2400 футов. Лента намотана на катушку для облегчения управления. В системах мини-ЭВМ часто применяются ленты в кассетах меньших размеров. Функционально они идентичны. Толщина ленты от 1/500 до 1/1000 дюйма; она изготавливается из гибкого материала, такого, как майлар, и имеет покрытие из магнитного материала (окись железа). Информация хранится на ленте в виде слоя окисла, намагниченного в одном или другом направ-

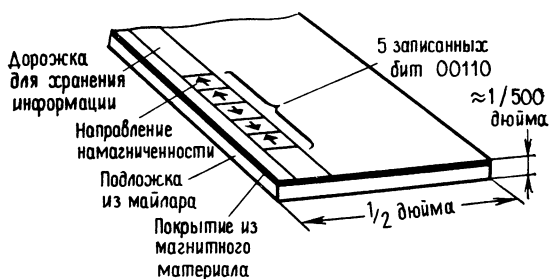


Рис. 9.4. Магнитная лента.

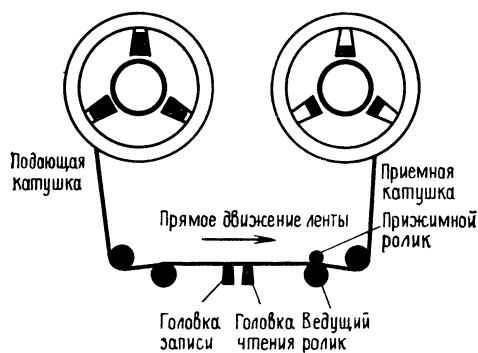


Рис. 9.5. Лентопротяжное устройство

ленте. **Дорожка** на ленте — это часть ширины ленты, на которой хранится закодированная намагничиванием покрытия информация (рис. 9.4). Так как дорожка имеет ширину менее  $1/16$  дюйма, на ленте можно разместить несколько дорожек. Полудюймовая лента обычно имеет 7 или 9 дорожек и, следовательно, можно **параллельно** хранить 7 или 9 бит, другими словами, один 6- или 8-битовый символ плюс бит четности. Для того чтобы считать или записать на ленту, необходимо установить катушку ленты в **лентопротяжное устройство**, которое представляет собой механизм с двумя вращающимися катушками, головкой чтения, головкой записи и ведущим роликом, к которому лента прижимается **прижимным роликом**. Ведущий ролик протягивает ленту мимо головки приблизительно с постоянной скоростью (рис. 9.5). Когда на ленте производится запись информации, она продвигается под головкой лентопротяжного устройства и усилители записи управляют током записывающей головки для намагничивания ленты. При считывании лента протягивается под считывающими головками, индуцированное напряжение с которых поступает на усилители считывания.

На практике биты информации упаковываются очень плотно вдоль ленты, например 200, 556, 800 и 1600 бит/дюйм на каждой дорожке; это стандартные значения плотности записи информации. Так как лента движется со скоростью 120 дюйм/с, остановка ее между символами невозможна. Поэтому информация обычно пишется на ленту непрерывными **записями**, или **блоками**, с промежутками между ними для остановки и запуска движения ленты. Эти промежутки между записями имеют длину около  $3/4$  дюйма. Блоки могут иметь фиксированную или переменную длину в соответствии с требованиями программиста. Важно понять разницу между этими двумя возможностями, поэтому рассмотрим причину их использования.

Уже упоминалось, что перфолента имеет специальную дорожку со сплошной пробивкой для того, чтобы позволить устройству считывания узнать, где на перфоленте был пробит символ. Аналогичным образом ЭВМ должна узнать, где на магнитной ленте написана группа битов (символ). Здесь не используется дорожка со сплошной пробивкой (трудно пробить 1600 отверстий на дюйм), но в некоторых системах применяется эквивалент подобной дорожки — **дорожки синхронизации**. Дорожка синхронизации — это дорожка на ленте, которая содержит магнитный сигнал в каждой позиции бита. Устройство считывания с ленты использует эту дорожку, чтобы указать, когда выявлена позиция символа на ленте, и тогда оно может считать настоящие информационные биты. Другим вариантом является требование, чтобы каждый символ, написанный на ленте, содержал по крайней мере один магнитный сигнал, с тем чтобы при чтении символа хотя бы в одной дорожке был записан сигнал. Это сделать не слишком трудно, так как каждому символу соответствует бит четности. Используя подходящую четность, можно быть уверенным в наличии по крайней мере одного сигнала. Могут применяться и другие методы записи, гарантирующие, что сигнал будет верно воспринят в каждой позиции, даже если дорожки рассматриваются отдельно. Важно, что при использовании дорожки синхронизации лента должна быть предварительно **размечена**; для этого любым способом на ленту записывают дорожку синхронизации и она должна оставаться на ленте постоянно. Считывая сигналы синхронизирующей дорожки, можно всегда вернуться к одной и той же точке ленты. Однако при отсутствии синхронизирующей дорожки лентопротяжное устройство должно обеспечить собственную информацию синхронизации во время записи новых данных, используя электронным образом генерируемые сигналы синхронизации. Так как невозможно гарантировать, что лента будет двигаться точно с одинаковой скоростью во время выполнения двух последовательных опера-

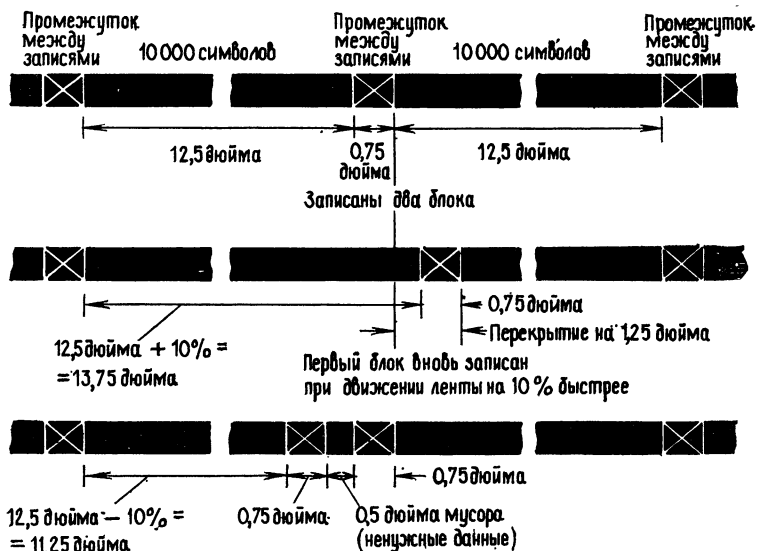


Рис. 9.6. Влияние непостоянной скорости ленты.

ций, записанный блок с заданным числом символов может менять длину между двумя выполнениями операции ЗАПИСЬ. Это показано на рис. 9.6. В верхней части рисунка показана лента с двумя записанными блоками информации по 10 000 символов с плотностью 800 бит/дюйм (около 12,5 дюйма каждый). В середине рисунка показан первый блок, переписанный при движении ленты со скоростью, большей на 10 %. В результате произошло перекрытие на 1,25 дюйма следующего блока. В нижней части рисунка показан эффект перекрытия при переписи первого блока на ленту, которая двигалась на 10 % медленнее. На этот раз на ленте осталось почти 0,5 дюйма информации от предыдущего блока. При считывании второго блока в обоих случаях будет обнаружена ошибка.

На ленте, имеющей дорожку синхронизации, возможна повторная запись данных в любой блок. В общем случае описанный способ не может быть реализован для лент без дорожки синхронизации; такая лента используется последовательно, т. е. перед записью данного блока необходимо записать все блоки, предшествующие ему. Однако некоторые изготовители утверждают, что можно создать настолько точные лентопротяжные устройства, что блоки могут быть записаны в произвольных точках.

Сравнивать размеченные и не размеченные ленты трудно. С одной стороны, содержимое размеченной ленты можно изме-

нить в любой точке. С другой стороны, часть ленточного пространства используется для синхронизирующей информации, а оно могло бы использоваться для хранения информации. (Точная запись при наличии бита в дорожке синхронизации сложнее, чем запись, когда лентопротяжное устройство определяет синхронизацию.) Если необходима перезапись информации в любой точке ленты, то для программиста легче использовать **блоки фиксированной длины**, и в действительности лента часто размечается таким образом. Однако по системным соображениям имеется тенденция использовать блоки фиксированной длины на неразмеченной ленте. Крупнейший изготовитель ЭВМ, фирма IBM, использует неразмеченные ленты и поэтому такие ленты наиболее распространены.

Основные операции с лентой включают запись и чтение блоков информации. Кроме того, необходимы операции управления лентой, чтобы дать возможность программе получить доступ к конкретному блоку. После того как блок уже записан на ленту, необходимо продвинуть (перемотать) ленту на один шаг назад, чтобы его вновь считать. (Некоторые системы позволяют производить считывание и запись на ленте в любом направлении, но в основном допускается считывание и запись только в прямом направлении.) Иногда необходимо продвинуть ленту на один или несколько блоков, чтобы считать блок, записанный на ленте где-то за ними; в этих случаях желательна операция продвижения на шаг вперед.

К содержимому магнитной ленты возможен только последовательный доступ, т. е. это больше напоминает перематывание рулона, чем листание книги. Рулон читают, перематывая его с одного конца; чтобы достигнуть параграфа в середине, необходимо просмотреть все предыдущие параграфы. Вообще не существует операции, обеспечивающей доступ к произвольному блоку на ленте. Для этого необходимо пропустить все предыдущие блоки.

Многие термины, вводимые ниже, появились при обработке экономических данных. Базовым элементом информации в этом случае является **символ**, который может быть частью имени, адреса или числовой записи. Символы группируются в **поля**, представляющие имена. Группы полей собираются в **запись**, которая может представлять основной блок, записанный на ленту. Исторически слово **запись** использовалось в смысле записанного блока информации. Мы будем применять это слово для обозначения логической группы информации, обычно управляемой пользователем как элементом, а термин **блок** будем использовать для обозначения записанной группы информации. Запись может, например, содержать всю информацию, связанную с одной работой. Наконец, все используемые

записи группируются в **файл**, или **набор данных**. Иногда один файл может быть многоленточным, тогда как в некоторых случаях он занимает только небольшую часть одной ленты. (Катушка ленты может содержать 30 млн. символов.) В файле на магнитной ленте используется специальная метка **конец файла**, которую ЭВМ может записать вместо обычного блока. При считывании эта метка воспринимается программой как указание на то, что достигнут конец файла. Наиболее общим аналогом для файлов, записей и символов на ленте является книга. Параграфы подобны записям, а файл — всей книге. Здесь нет деления на разделы и главы. Программист может их создать, используя метки конца файла или другие методы. Следуя этой аналогии, блоки фиксированной длины являются аналогами страниц книги. Блоки не вполне соответствуют естественному делению информации на разделы, параграфы и т. д., но они обеспечивают простой способ определения местоположения частей информации, если они снабжены индексами. На ленте фиксированные блоки могут не соответствовать требованиям информации, но они могут использоваться с успехом, если информация индексирована.

Обычно две операции обеспечивают использование маркеров файла — это продвинуть назад на файл и пропустить метку конца файла. Они упрощают прохождение через несколько файлов с минимальными программными затратами. Эти операции, допускаемые в типичных системах с размечаемыми и неразмечаемыми лентами, указаны в табл. 9.1. Во многих лентопротяжных устройствах с раздельными головками чтения и записи производится контроль информации при записи посредством ее последующего считывания. Если используется бит четности (почти все магнитные записывающие устройства имеют этот бит) и лента проходит под головкой считывания, после того как она прошла под головкой записи, контроль информации может быть осуществлен по четности. Это приводит к обнаружению любой одиночной ошибки. Если ошибка обнаружена, операцию можно повторить; для этого ленту следует продвинуть назад к началу блока и повторно его записать.

Когда корректируется информационный файл на магнитной ленте (т. е. изменяется некоторая информация), необходимо считать оригинал с одной ленты и сделать скорректированную копию на другой ленте второго лентопротяжного устройства. Это не является необходимым только тогда, когда используется размеченная магнитная лента с блоками фиксированной длины и при этом скорректированный элемент информации не длиннее первоначального. Для любой операции требуется как минимум два лентопротяжных устройства с неразмеченной лентой, хотя для большинства задач обработки данных их должно



Таблица 9.1. Размеченная и неразмеченная ленты

Операция	Неразмеченная лента	Размеченная лента
Запись	Программа указывает количество слов	Длина блока кратна фиксированному количеству слов
Чтение	Режим: либо весь записанный блок, либо несколько слов, указанных программой. Ограничение: невозможно считывание в направлении, обратном операции записи	Фиксированный блок, как он был записан, или заданное количество слов
Продвижение назад	На блок или, возможно, на файл	Только на блок <sup>а)</sup>
Продвижение вперед	На блок или, возможно, на файл	Только на блок <sup>а)</sup>
Перемотка к началу ленты	Есть	Есть
Запись метки конца файла	Есть	Не допустимо <sup>а)</sup>
Считывание метки конца файла при чтении	Есть	Не допустимо <sup>а)</sup>

<sup>а)</sup> Допускаются указания дополнительного символа в начале каждого блока для представления метки файла; в этом случае возможны все файловые операции.

быть как минимум четыре. Две ленты необходимы для того, чтобы информацию с одной ленты можно было в скорректированном виде записать на другую. В течение любого времени можно хранить неограниченное количество катушек ленты, обрабатываемых на ЭВМ. Таким образом, верхняя граница времени хранения не ограничена. Скорость операции у разных устройств и изготовителей различна и составляет от нескольких тысяч символов в секунду до 20 000 символов в секунду. Для просмотров (считывания) записей и файлов требуется столько же времени, сколько необходимо для считывания или записи, тогда как на простую перемотку уходит максимум 1—2 мин. Время на разгон и остановку ленты на межблочном промежутке составляет от 1 до 20 мс. В табл. 9.1 приведены сравнительные данные для размеченных и неразмеченных лент.

### Дисковые устройства

Диск, называемый также **дисковым файлом** (или файлом на дисках), — это устройство, которое похоже на грампластинки и проигрывающий механизм. Дисковая пластина представ-

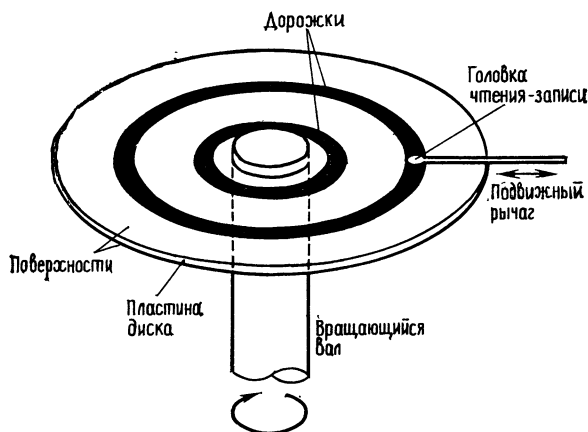


Рис. 9.7. Диск.

ляет собой круглый лист металла диаметром около 10—20 дюймов и толщиной  $1/16$ — $1/4$  дюйма. Обе поверхности покрыты магнитным материалом, так что толщина диска определяется только требованием сопротивления механическим усилиям. Информация записывается на дисковую поверхность за счет изменения ее магнитных характеристик (так же как и на магнитную ленту). Различие между записью на грампластинке и диске (помимо хранения информации) состоит в том, что грампластинка имеет один непрерывный желобок, или дорожку, тогда как на дисковой поверхности много дорожек в виде концентрических окружностей. Головки чтения-записи (одна и та же головка используется для обеих целей) смонтированы на движущихся рычагах, благодаря чему они могут быть установлены на любой дорожке; на поверхности может быть от 100 до 1000 дорожек. Диаграмма дисковой поверхности показана на рис. 9.7.

В дисковом устройстве могут быть доступны все записи одной поверхности. Пакет дисков может содержать несколько пластин (дисков), каждая с собственной головкой чтения-записи, на собственном рычаге, так что в таком пакете одновременно доступна любая поверхность. Если дисковый пакет съемный, то дисковый файл будет удовлетворять нашему определению долговременной и, что более существенно, неограниченной памяти. Он имеет характеристики, подобные магнитным лентам; ему может потребоваться относительно большое время для перемещения головки с текущей позиции на другую, подобно тому как при чтении ленты много времени занимает переход от одной позиции к другой. Однако это происходит у диска

гораздо быстрее; типичное максимальное время перемещения головки составляет около  $1/10$  с, тогда как для нахождения нужной позиции на ленте требуется до 4 мин. Скорость передачи символов диском обычно составляет от 100 КС (КС — килосимвол = 1000 символов) до 400 КС в секунду, тогда как для ленты — от 30 до 180 КС. Большой дисковый пакет содержит примерно столько же информации, что и лента. (Лента содержит около  $30 \times 10^6$  символов при плотности записи 1000 символов на дюйм и длине 2400 футов; диски обычно содержат от  $10 \times 10^6$  до  $400 \times 10^6$  символов.)

Запись на диске почти всегда осуществляется в формате фиксированных блоков, так как длина каждой дорожки фиксирована. Дисковый пакет содержит метки сектора, которые представляют собой физические метки и могут быть считаны дисковым устройством для определения текущей позиции на диске. Обычно дисковый пакет содержит от 20 до 40 меток секторов. Одна из меток является специальной (часто используется двойная метка) и называется **меткой индекса**. Она указывает начало дорожки. Блоки могут быть записаны, начиная с любой из меток секторов. При записи дисковое устройство использует синхросигналы для синхронизации. Во время чтения данные с диска сами содержат достаточно информации для обеспечения синхронизации. Скорость дискового файла поддерживается достаточно точной, так что возможна запись в любой блок, не изменяя других.

Во многих дисковых устройствах используется довольно сложная процедура, позволяющая гарантировать, что под головкой еще до чтения или записи находится необходимая дорожка или сектор. В большинстве дисков применяется **головная метка**, содержащая примерно 50 бит информации о дорожках, секторах и длине блока. Эта информация записывается на дисковый пакет, когда он обрабатывается впервые; эта операция называется **разметкой диска**. Системный программист выбирает для системы размеры блока и передает их в головную метку, которая записывается перед каждым блоком. Если диск будет использоваться позже, программа должна вначале установить головку на требуемую дорожку (так называемая операция поиска), затем указать информацию головной метки до того, как может быть осуществлено чтение или запись. Дисковое устройство сравнивает эту информацию с информацией головной метки и не выдает разрешение на выполнение операции пока не произойдет совпадение. Большинство пользователей не осведомлены об этом процессе, так как он проводится системной программой во время выполнения дисковой операции.

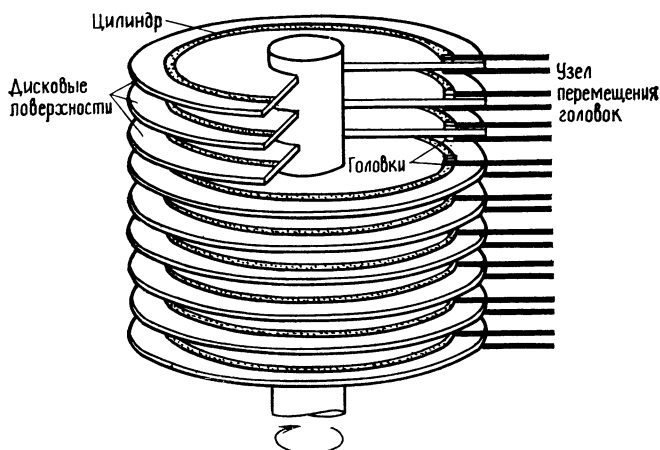


Рис 9.8. Дисковый файл с несколькими поверхностями.

Когда головки зафиксированы в каком-нибудь положении, ЭВМ имеет доступ к одной дорожке на каждой поверхности. Из рис. 9.8 видно, что дорожки на поверхностях дисков находятся на поверхности цилиндра. По этой причине при адресации к дискам часто используют **адрес цилиндра** и **адрес поверхности**. Первый указывает на местоположение головок, второй — на местоположение используемых поверхностей. Вместе эти компоненты образуют адрес дорожки. Для обмена несколькими блоками информации между ЭВМ и дисковым устройством потребуется меньше времени, если эти блоки будут находиться на одном и том же цилиндре. Головка в этом случае не перемещается.

Один из современных типов дисков называется **гибким (флоппи) диском**; это гибкая круглая пластина из майлара, покрытого магнитным окислом. Такое устройство имеет очень низкую цену — примерно 1/10 цены стандартных дисковых устройств небольших размеров — и часто используется в системах мини-ЭВМ.

### Разновидности лент и дисков

Многие устройства принадлежат к тому же классу, что и ленты и диски, т. е. допускают смену носителя информации (их память виртуально не ограничена), а также имеют приемлемые скорости и емкости для отдельной загрузки. Эти устройства имеют форму пакета дископодобной или лентоподобной среды памяти, доступ к которой осуществляется таким образом, что отдельный участок среды размещается под механизмом

чтения-записи. Устройства такого рода способны передавать более  $10^9$  символов в **оперативном режиме** (online), т. е. сразу в машину.

Одним из таких устройств является **ячейка данных**. Это устройство содержит катушки, в каждой из которых находятся полоски гибкой основы, покрытые магнитным материалом. Механизм чтения имеет барабан, вокруг которого может быть обернута одна из этих полосок. Барабан вращается под головками чтения-записи для записи или считывания данных. Механическая система доступа может загрузить или разгрузить любую из полосок ленты. Скорость загруженной полоски ленты такая же, как скорость диска, но требуется значительное время доступа к ленте, если она находится на некотором расстоянии от механизма доступа.

Главным достижением последнего десятилетия является уменьшение стоимости устройств и некоторое увеличение скорости. Не уменьшая физические размеры механических частей механизмов, трудно перемещать их со скоростью, значительно большей скорости, достигаемой в настоящее время, хотя увеличение плотности записи допускает некоторое повышение скорости. Однако уменьшение стоимости интегральных схем и значительное увеличение их производства приводят к значительному удешевлению устройств. Малые медленные устройства, использующие кассеты магнитофонных лент шириной 1/8 дюйма, стоят до 100 долл., а гибкие дисковые устройства — от 500 до 1000 долл. Большие дисковые устройства на треть миллиарда символов стоят в 20 раз дороже.

#### **9.4. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ПРОМЕЖУТОЧНОГО ХРАНЕНИЯ**

Под **промежуточным хранением** будем понимать хранение информации от нескольких секунд до нескольких дней, когда непрактично извлекать среду хранения и помещать ее для хранения в шкаф. Для устройств этой категории характерны скорости в диапазоне от скоростей, которыми обладают устройства долговременного хранения, до примерно нескольких миллионов символов в секунду. Емкости этих устройств находятся в пределах от тех, которые имеют устройства долговременного хранения, до примерно нескольких миллионов символов. Эти устройства применяются для расширения непосредственно адресуемой основной памяти, емкость которой недостаточна, а также для сохранения информации от одной выполняемой задачи до другой, когда неудобна физическая перезагрузка информации запоминающей среды в считывающее устройство. Типичными ситуациями, в которых возникает потребность в таких

применениях, являются: 1) сортировка большого количества вводимой информации (скорее всего на магнитной ленте), которая требует многократного просмотра данных, копирования их из одной области памяти в другую, и 2) сохранение программ пользователей и данных в системе разделения времени с дистанционно удаленными пультами, где почти каждому пользователю необходимо иметь непосредственный доступ к личной информации.

Потребности вышеприведенных примеров могут быть удовлетворены использованием устройств долговременного хранения, о которых упоминалось выше, но для некоторых задач недостаточна максимальная скорость данных или велико среднее время доступа (т. е. время движения относительно головок или лент), поэтому для этих случаев нужны усовершенствованные устройства. Однако в типичной установке такие устройства могут использоваться в основном как буферная память, т. е. для временного хранения информации только во время выполнения программ. Помимо этого могут применяться и другие устройства.

## Диски

Диски рассматривались выше как устройство со сменными дисковыми пластинами для долговременного хранения. Если пластины не сменные, то на отдельном устройстве можно разместить больше пластин. Фактически такими были первые разработанные диски. Физически они подобны сменным дискам и используются таким же образом. Однако емкость их памяти может быть увеличена; значительно возрастает скорость доступа, если одновременно пользоваться более чем одной головкой чтения-записи на нескольких дорожках. Существуют диски, в которых имеется одна головка на каждую дорожку. Такие диски называются **барабанами**, хотя обычно различаются **диски с фиксированными головками** и барабаны. Так как большой диск может иметь 8000 дорожек, расположение одной головки на одной дорожке возможно лишь для дисковых устройств небольших размеров. Однако такое расположение устраняет перемещение рычагов головки — самую медленную часть операции дискового файла. Среднее время доступа сокращается до половины времени оборота диска (примерно  $35/2$  мс), так как в среднем необходима половина оборота вращения требуемой информации до положения, когда она окажется под читающей головкой, и только тогда информация может быть передана в ЭВМ. Средняя задержка, вызванная вращением, называется **временем ожидания**.

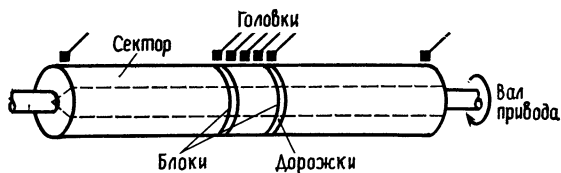


Рис. 9.9. Барабан.

## Барабаны

Исторически барабаны появились до дисков, но их удобно рассматривать как вариант дискового файла с фиксированными головками. Вместо того чтобы хранить информацию на группе дисковых поверхностей, ее хранят на поверхности барабана диаметром 10 дюймов и длиной по крайней мере 20 дюймов (хотя разработаны барабаны и значительно больших размеров). Так как барабаны более компактны, они могут иметь большие скорости вращения, чем дисковые механизмы, следовательно, среднее время доступа может быть сокращено. Типичное время вращения составляет от 8 до 35 мс. Информация на барабане обычно хранится в блоках фиксированной длины. На рис. 9.9 схематично изображен барабан. Он напоминает цилиндр дискового файла, но по техническим причинам возможно получение более высокой плотности битов на барабане по сравнению с диском. Для достижения высокой скорости передачи данных несколько дорожек могут считываться параллельно, так что за один оборот может быть прочитано большое количество слов. Чтобы блоки имели управляемые размеры, дорожки делятся на **сектора**. Каждый сектор содержит блок для считывания и записи. Доступ ЭВМ к барабану обеспечивается указанием адреса дорожки и сектора. Как только считан или записан один сектор, немедленно становится доступным сектор со следующим по порядку номером, так как переключение от одного набора головок к другому происходит почти мгновенно.

## Массовая память

Первичная память ЭВМ конструируется из многих идентичных элементов, каждый из которых может хранить 1 бит и может быть доступен за почти одинаковое время. В качестве таких элементов используются сердечники или магнитные тонкие пленки или элементы интегральных схем. Скорости считывания или записи обычно составляют от 0,1 до 1 мкс для считывания или записи одного слова, содержащего от 16 до

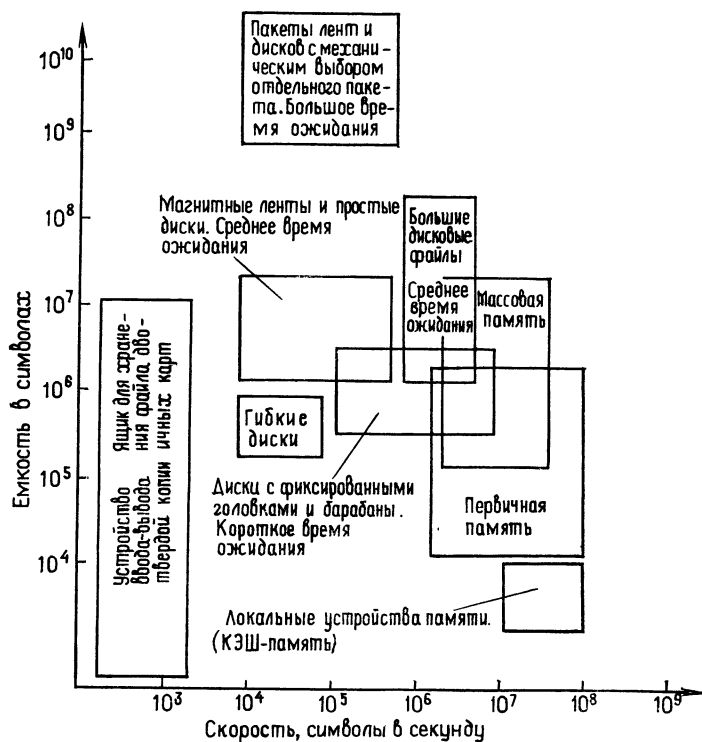


Рис. 9.10. Современные устройства памяти.

64 бит. Массовая память (bulk store<sup>1)</sup>) — это такое же устройство, но в котором количество слов значительно увеличено по сравнению с типичной основной памятью емкостью от 10<sup>4</sup> до 10<sup>6</sup> слов. Такая память может хранить до 10 млн. символов. Вследствие увеличения размеров ухудшается скорость, для доступа обычно требуется 2 мкс и более. Устройство, использованное фирмой CDC в расширенной памяти на сердечниках, имеет значительно большее слово массовой памяти. Обычная память содержит 60-битовые слова, а массовая память — 480-битовые слова. Таким образом, с каждого из 4 блоков массовой памяти параллельно считываются 8 слов, что обеспечивает максимальную скорость 100 нс на слово, которая является максимальной скоростью основной памяти. Такая скорость достигается за счет перекрытия времени работы четырех блоков памяти со временем доступа 400 нс. Так как такие

<sup>1)</sup> В иностранной литературе для обозначения подобной памяти чаще используется термин mass storage. — *Прим. ред.*



скорости могут быть достигнуты, только когда пересылается большое количество последовательно расположенных в памяти слов (на первое слово требуется 3,2 мкс), описываемое устройство может использоваться только как устройство передачи блоков между основной памятью и массовой памятью. Эта память больше похожа на барабан или диск, но она обладает бóльшим быстродействием и не имеет времени ожидания, обусловленного временем оборота. Такую память нельзя использовать для передачи команд и данных непосредственно в центральный процессор. Однако программа может считывать данные (и команды) из расширенной памяти на ферритах, выполняя команду копирования последовательно расположенных блоков слов из расширенной памяти на ферритах в первичную память. Затем эти данные или команды могут быть использованы в следующей части программы.

### **9.5. СРАВНЕНИЕ СКОРОСТИ И ЕМКОСТИ**

На рис. 9.10 проведено сравнение скорости и емкости памяти различного типа со скоростью и емкостью основной памяти.

## СИСТЕМЫ С РАЗДЕЛЕНИЕМ ВРЕМЕНИ

*Г. Хеллерман, Т. Конрой***10.1. ВВЕДЕНИЕ**

Первые вычислительные системы с разделением времени, обладающие основными чертами систем такого рода в современном виде, появились в начале 60-х годов. Хотя в какой-то степени к их числу могут быть отнесены системы военного назначения для «командования и управления», а также системы резервирования авиабилетов, в настоящее время под системой с разделением времени принято понимать вычислительную систему коллективного пользования, в которой каждый пользователь имеет возможность создавать и выполнять свои собственные программы, полагая, что имеет дело с универсальной ЭВМ, полностью находящейся в его распоряжении, причем система должна реагировать на некоторые запросы пользователя достаточно быстро, чтобы взаимодействие человека с машиной было удобным. По сравнению с более известными системами пакетной обработки системы с разделением времени предоставляют пользователю другой способ взаимодействия с ЭВМ. Так как этот способ в большей степени обеспечивает контакт пользователя с ЭВМ, становятся более эффективными такие виды деятельности, как решение задач, техническое проектирование, а также все виды разработки и отладки программ. Значительное расширение возможностей творческой деятельности человека не только приводит к повышению производительности его труда благодаря более продуктивному использованию творческих способностей человека, но и служит стимулом развития этих способностей.

Обеспечивая более полную реализацию потребностей пользователя, системы с разделением времени позволяют повысить и эффективность использования ресурсов ЭВМ (по крайней мере, потенциальных). Одна из предпосылок этого основана на предположении, что высокая скорость поступления запросов

к системе от многих пользователей, каждый из которых обращается к системе со своим пакетом, улучшит эффективность использования ЦП; в обычных пакетных системах она, как правило, не превышает 30—40 %. Однако эти потенциальные возможности могут быть реализованы не полностью из-за необходимости выполнения дополнительных вспомогательных операций, связанных с обменом информацией (см. ниже).

«Разделение времени» центрального процессора, от которого происходит название всей системы, представляет собой лишь одну из разновидностей практического применения принципа совместного использования ресурсов несколькими пользователями. В качестве других примеров следует назвать совместное использование основной памяти и других устройств ЭВМ, а также программ, главным образом, трансляторов (интерпретаторов и компиляторов), библиотек подпрограмм и прикладных программ. Возможности доступа к ресурсам ЭВМ за время, соизмеримое с временем реакции человека, превращают системы с разделением времени в эффективное средство коммуникации, посредством которого пользователь может легко воспользоваться результатами, полученными другими пользователями.

Кратко перечислив основные преимущества систем с разделением времени, перейдем к рассмотрению некоторых характерных особенностей их реализации. Одна из особенностей заключается в том, что в большинстве средних и больших вычислительных систем при организации разделения времени предъявляются относительно слабые требования к числу и характеристикам нужных для этого устройств. Наличие у центрального процессора трех принципиальных особенностей: **схемы прерывания, генератора сигналов истинного времени и механизма защиты памяти**, было достаточно характерно для большинства ЭВМ с 60-х годов, поскольку все они были необходимы для обеспечения работоспособности операционных систем мультипрограммной пакетной обработки. Существенной и даже в некоторой степени отличительной принадлежностью систем с разделением времени является так называемый **терминал**, который в современном виде включает устройство типа пишущей машинки, такое, как телетайп или пишущая машинка с шариковой головкой фирмы IBM, и соответствующее электронное оборудование для подключения этого устройства к телефонной линии. Терминалы этого типа сравнительно недорогие и пользователи могут с их помощью подключаться к любой системе с разделением времени, которая способна обслуживать терминалы данного типа (при условии, конечно, что заключены соответствующие соглашения о пользовании системой). Не составляет принципиальной трудности подключение

одного терминала с помощью обычной техники телефонной связи в различные моменты времени к удаленным от него центральным процессорам для удовлетворения многообразных потребностей в средствах для решения задач, написанных на различных языках программирования, и в других средствах вычислительной системы. Обычные широко разветвленные телефонные системы часто представляют собой характерную составляющую часть систем с разделением времени. Сигналы, поступающие в вычислительную часть системы по входным линиям, электрически преобразуются в модем (модулятор — демодулятор) и фиксируются посредством мультиплексирования в одном регистре. Затем они распределяются по буферным областям основной памяти, в которых находятся системные программы, предназначенные для обработки информации в соответствии с этими сигналами. Функция мультиплексирования реализуется с помощью устройства, которое обычно называется **устройством управления передачей сигналов**.

Окупаемость системы с разделением времени достигается за счет одновременного обслуживания достаточно большого количества пользователей. Типичная средняя ЭВМ обслуживает примерно 50 пользователей. Увеличение числа пользователей является существенным фактором с точки зрения оправдания экономических затрат. Их одновременная работа технически осуществима, так как новые пользователи могут легко подключаться к системе. Это возможно благодаря тому, что в процессе длительной работы за терминалом пользователь много времени тратит на обдумывание своих действий и получаемой информации, а также на работу с клавиатурой. Разумеется, одним из основных объективных критериев эффективности системы с разделением времени является достаточно быстрая реакция на некоторые важные команды пользователя, такие, например, как команды редактирования программ, для выполнения которых в ЭВМ с достаточным быстродействием требуется относительно небольшое время.

Для совместного обслуживания множества пользователей с их программами и данными нужна значительная память. Требуемые размеры этой памяти обычно существенно превышают экономически оправданные размеры быстродействующей основной памяти. В любое время можно удовлетворить требования только нескольких, а иногда только одного пользователя на пространство памяти, а для информации остальных пользователей необходимо на это время предусмотреть более дешевую и более медленную, но и более емкую вспомогательную память, реализуемую в виде запоминающих устройств на магнитных барабанах или дисках. Если система приступает к работе над отдельным заданием, когда оно не находится

в основной памяти (что случается довольно часто), прежде всего необходимо переместить текущее выполняемое задание из основной памяти, например на барабан, сохранив при этом информацию о его состоянии для последующего возобновления обработки. Затем нужное задание пересылается с барабана на освободившуюся часть основной памяти. Эта последовательность действий, называемая **свопингом**<sup>1)</sup>, типична для большинства систем с разделением времени. Свопинг производится автоматически программой операционной системы. Существует много других способов организации обслуживания множества пользователей, некоторые из которых будут рассмотрены ниже. Все эти способы должны удовлетворять одному главному требованию — обеспечить быстрый ответ на те запросы, которые имеют существенное значение для взаимодействия человека с машиной.

Введем некоторые основные термины. При рассмотрении диалоговых систем под **трактом** (tract) будем понимать однократное взаимодействие между пользователем и системой; этот термин является сокращением от слова **транзакция** (transaction) (запрос — ответ). Понятие тракта во многом аналогично понятию **задания**, но выполнение задания отдельного пользователя (понимаемого как выполнение некоторой программы) обычно связано со множеством взаимодействий пользователя с системой и, следовательно, со множеством трактов.

Некоторые системы с разделением времени предоставляют пользователям возможность работы на одном языке программирования; в этом случае систему иногда называют **специализированной** (dedicated). Примерами специализированных систем являются QUIKTRAN фирмы IBM и некоторые системы, использующие языки APL и BASIC. Иногда теоретики используют применительно к вычислительной системе термин «специализированная», имея в виду, что пользователь может применять различные языки программирования, но совместное выполнение пользовательских и фоновых (пакетных) заданий невозможно. Поскольку выполнение программ в пакетном режиме часто требует относительно больших машинных ресурсов, необходимо, чтобы языки и компиляторы, использование которых возможно при работе за терминалами, были **совместимы** с теми, которые служат для пакетной обработки. Если такая совместимость не обеспечена, это может вызвать много неприятностей, особенно в ситуациях, когда задание подразумевает применение небольших по объему алгоритмов для обработки больших объемов данных. В настоящее время наблюдается

---

<sup>1)</sup> В оригинале swapping — чередующаяся загрузка или перекачка.

тенденция к созданию многоязычных неспециализированных систем с разделением времени, языковые средства которых совместимы с языками пакетной обработки.

Резюмируя наиболее существенные из отмеченных выше соображений, отметим, что наиболее важными качествами систем с разделением времени являются эффективность, экономичность и удобство взаимодействий человека с ЭВМ. Совместное использование общих ресурсов, включая аппаратуру, программы и данные, с учетом времени реакции человека требует применения вполне определенного, достаточно своеобразного оборудования для обслуживания каждого рабочего места пользователя (терминалы и средства коммуникации). При построении систем с разделением времени могут быть широко использованы существующие телефонные сети; это требует существенной переориентации внутренних средств системного управления своими собственными ресурсами, т. е. создания операционных систем, отличающихся от обычных операционных систем пакетной обработки.

## **10.2. ТОЧКА ЗРЕНИЯ ПОЛЬЗОВАТЕЛЯ И НЕКОТОРЫЕ ЕЕ СЛЕДСТВИЯ**

Во введении были отмечены некоторые основные преимущества, получаемые пользователем в случае применения систем с разделением времени. Попытаемся найти подтверждение этих сформулированных выше концепций. Анализ подтверждений такого рода представляет собой довольно трудную задачу, так как при этом следует опираться на практические навыки конкретных людей, а на основе такого материала весьма трудно сделать заключения общего характера. Одно из исследований было проведено в Массачусетском технологическом институте с участием 66 студентов класса менеджеров. В этом исследовании использовалась имитационная модель строительного производства, реализованная на языке имитационного моделирования DYNAMO. Каждому студенту была предоставлена информация о состоянии отрасли и рынка, и он должен был использовать реализованную на ЭВМ модель оптимизации прибыли. Класс был разделен на две группы. Обе группы использовали одну модель, но одной из групп ЭВМ была предоставлена в распоряжение как система с разделением времени, а другой — как система пакетной обработки. Данные о производительности студентов и их впечатлениях о проделанной работе были получены из анкет путем анализа преподавателем результатов из обычного отчета ЭВМ о ресурсах, которые были применены каждым пользователем. По результатам данного эксперимента оказалось возможным сделать следующие основные выводы:

1. Общие затраты ресурсов системы и машинного времени пользователей для обеих групп были примерно равны, но их распределения оказались весьма различными. В группе, работавшей с разделением времени, относительно меньшие затраты пришлись на действия человека и более высокие — на использование ЭВМ.

2. Пользователи системы с разделением времени получили значительно лучшие решения поставленной задачи.

3. В группе, использовавшей пакетную обработку, вдвое большее число студентов, чем в группе, работавшей с разделением времени, вообще не получили приемлемого решения задачи.

4. Работы с разделением времени понравились большему числу участников эксперимента.

Результаты эксперимента свидетельствуют о важности учета выигрыша, достигаемого за счет получения лучшего решения задачи и отношения пользователей к системе, наряду с легко поддающимися подсчету непосредственными финансовыми затратами. Дальнейшие исследования результатов эксперимента позволяют сделать еще одно важное заключение: **расхождение в производительности пользователей внутри каждой группы значительно превышает различия, имеющие место между группами.** Это наводит на мысль, что, возможно, различия в способностях людей в большей степени сказываются на эффективности работы, чем используемый тип вычислительной системы.

С точки зрения отдельного пользователя немаловажными являются стоимость непосредственно применяемого оборудования и его обслуживание. Указанные на рис. 10.1 категории затрат представляют больший интерес, чем конкретные значения этих затрат, которые, несомненно, будут меняться со временем. Стоимость аренды терминала, стоимость сопряжения с телефонными линиями и стоимость аренды самих телефонных линий не нуждаются в дальнейшей детализации. Стоимость аренды ЦП относится только к времени, в течение которого система обслуживает конкретного пользователя. Стоимость подключения определяется количеством часов, в течение которых пользователь подключен к системе независимо от того, использует он ресурсы системы или нет; это мотивируется необходимостью отведения некоторых основных ресурсов ЭВМ, таких, как области основной памяти для терминальной буферизации и подканалы для управления линиями связи. На рис. 10.1 приведены тарифы затрат по различным категориям; при этом величина затрат в ряде случаев зависит от времени. В типичных ситуациях большая часть затрат времени приходится на обеспечение связи между частями системы, а не

Категория затрат	Типовые нормы оплаты	Количество единиц	Стоимость долл./месяц
Аренда терминала (IBM2741)	100 долл./мес.	1	100
Устройство сопряжения с телефонной линией	28 долл./мес.	1	28
Аренда телефонной линии <sup>а)</sup>	3 долл./миля/мес.	50 миль	150
Аренда ЦП <sup>а)</sup>	5 долл./мин	20	100
Стоимость подключения <sup>а)</sup>	12 долл./ч	40	480
Плата за файлы	1/7 долл./(Кбайт/мес.)	70 Кбайт	10

<sup>а)</sup> Расходы на связь обычно включают стоимость 1 мин аренды ЦП в расчете на час связи.

Рис. 10.1. Типовые нормы оплаты (по состоянию на 1971 г.).

на работу ЦП. По этой причине широкое применение находят устройства, называемые **концентраторами телефонных линий**, которые объединяют сигналы, относящиеся к несколько территориально близким друг к другу терминалам; при этом относительно высокая арендная плата выделенных линий связи с центральной вычислительной системой раскладывается на пользователей этих терминалов. Этот способ связи пользователей с системой особенно эффективен при значительном удалении группы терминалов от ЭВМ (т. е. при возрастании стоимости аренды каналов связи).

Чтобы разобраться в факторах, определяющих производительность систем с разделением времени, попробуем представить себе, какую форму общения с системой пользователь счел бы достаточно удобной. В связи с тем что большинство показателей системы являются характерными для любых вычислительных систем общего назначения, нетрудно составить длинный список желаемых качеств, но это лишь в малой степени позволит уяснить интересы пользователей именно систем с разделением времени. Целесообразнее поставить вопрос: какие системные свойства являются специфичными или по крайней мере более важными для пользователя систем с разделением времени? Ответ можно начать с повторения соображений, приведенных во введении к данной главе: основным назначением системы с разделением времени является обеспечение скорости, удобства и экономичности. Основная цель данного раздела состоит в максимально возможном раскрытии смысла этого утверждения и иллюстрации его с помощью нескольких примеров.



1. *Программы и информационные объекты*
  - (а) Список текущих имен и их описание
  - (б) Назначение и уничтожение имен
  - (в) Объекты, адресуемые по имени
2. *Редактирование программ (операторы и целые программы) построчное и контекстное*
  - (а) Отобразить (Display)
  - (б) Модифицировать (Modify)
  - (в) Уничтожить (Delete)
  - (г) Вставить (Insert)

(Замечание: вместо оператора «Модифицировать» может быть использован оператор «Уничтожить», а затем «Вставить».)
3. *Редактирование данных*
  - а) Вводимых в программу
  - (б) Вычисленных данных
4. *Композиция*

Формирование из именованных объектов (программ, файлов) объектов с новыми именами
5. *Управление последовательностью действий*
  - (а) Начать сеанс и Кончить сеанс
  - (б) Начать выполнение программы
  - (в) Прервать программу
  - (г) Сведения о прерванной программе
6. *Обнаружение ошибок пользователя*
  - (а) Построчное при вводе
  - (б) После ввода всей программы
  - (в) Во время выполнения

Рис. 10.2. Список функциональных возможностей пользователя.

Рассмотрим сначала ситуацию, когда только один пользователь взаимодействует с «личной» ЭВМ через свой терминал. При этом сразу же отпадают вопросы о производительности системы, порождаемые наличием многих пользователей, совместно использующих ее ресурсы.

Большинство систем разделения времени часто называют «разговорными» по той причине, что в них взаимодействие человека и машины может осуществляться в форме «вежливой» беседы, в ходе которой стороны поочередно посылают сообщения друг другу. Эта очередность обычно устанавливается системой путем блокировки клавиатуры с момента получения переданной пользователем строки и до тех пор, пока она не отреагирует выдачей результатов или передачей собственного сообщения, а иногда и просто разблокировкой клавиатуры.

На рис. 10.2 указаны основные функциональные возможности системы, необходимые пользователю. Их выполнение обеспечивается системными программами, интерпретирующими команды. Посредством этих программ пользователь может выдавать запросы на обслуживание различного типа. Для более четкого понимания этих функций целесообразно сравнить их с потребностями пользователя пакетной системы. Более того, первоначально будет предполагаться, что транслятор языка,

который преобразует программы пользователя в программы, готовые к выполнению на ЭВМ, является программой компилирующего типа, какие часто включаются в системы пакетной обработки.

Для реализации большинства функциональных возможностей, указанных на рис. 10.2, требуется, чтобы программы пользователя и данные сохранялись в системе от сеанса к сеансу. Функции типов 1 и 4 обычно выполняются по отношению к именованным объектам с помощью схемы присвоения имен, которая реализуется в виде набора указателей. Во избежание путаницы между одинаково именованными объектами обработки разных пользователей в большинстве систем соблюдается иерархия указателей. В результате имена всех информационных единиц данного пользователя неявным образом автоматически включают его имя (или идентификатор) как составную часть.

Редактирование программ и данных обычно обеспечивается программами, которые предоставляются в распоряжение пользователя в виде набора «команд редактирования», соответствующих операциям, которые перечислены в п. 2. Редактирование исходных данных, предназначенных для ввода под управлением программ пользователя, может выполняться в основном теми же средствами, что и редактирование программ.

Наибольшие трудности с точки зрения реализации рассматриваемых функциональных возможностей представляют сведенные в п. 5 средства прерывания программ. (Напомним, что речь идет здесь лишь о возможности прерывания программ одним пользователем и не имеются в виду прерывания, инициируемые системой для выполнения свопинга.) Можно начать рассмотрение этой проблемы с вопроса, насколько необходима возможность реализации прерываний. А если необходима, то какими минимальными средствами можно обойтись для осуществления обработки прерываний? Возможность прерывания программ действительно существенна, так как из теории вычислений известно, что нет способов узнать заранее, конечна ли выполняемая программа. Таким образом, если программа выполняется достаточно долго, то она может находиться как в состоянии успешного выполнения, так и в состоянии заикливания. По этой причине системы пакетной обработки всегда снабжаются средствами снятия программы с решения (выбрасывания), позволяющими исключить неправомерную монополизацию ресурсов системы. Обычно выбрасывание осуществляет оператор машины, получив от операционной системы сообщение, что время выполнения программы превысило ориентировочную величину, которая была указана пользователем в карте задания. Такой способ можно было бы использовать и в систе-

мах с разделением времени. Однако ограничиваться лишь им было бы нецелесообразно, так как пользователь, работающий в непосредственном контакте с системой, часто (хотя и не всегда) может установить, что программа работает неправильно. К этому же заключению он может прийти, наблюдая за выдачей промежуточных результатов работы программы, которые он не видит при выполнении программы в пакетном режиме.

Хотя пользователь может прервать программу, например, используя кнопку «внимание» на своем терминале, не вполне ясно, какие средства необходимы для ответа на это действие. В простейшем случае он должен инициировать повторный запуск транслятора и выполнение программы, как правило, предварительно изменив исходную программу или исходные данные. Для этого компилятор не требуется снабжать какими-то особыми дополнительными возможностями. Поэтому в системе с разделением времени может быть использован компилятор, заимствованный из систем пакетной обработки, с некоторыми изменениями (способный, по крайней мере, реагировать на запросы прерывания), поскольку после каждого прерывания компиляция производится заново. Чтобы уменьшить время компиляции (часто за счет скорости выполнения объектной программы), желательно использовать быстрый компилятор, как, например, весьма удачный WATFOR для Фортрана IV, разработанный в Университете Ватерлоо. Если система должна предоставить пользователю возможность последующего возобновления прерванной программы без повторной трансляции, в ней должны быть предусмотрены дополнительные средства. Чтобы выяснить, что именно необходимо, следует проанализировать действия, желательные для пользователя после выдачи им запроса на прерывание. Во-первых, он, вероятно, захочет «увидеть» значения некоторых данных, например переменной цикла. Конечно, для этого ему нужно указать лишь их имена, содержащиеся в его программе. Для преобразования имени в адрес (местоположение) данных в памяти, требуется таблица символов и, вероятно, другие таблицы, такие, как, например, таблица смещений. Подобные таблицы после компиляции обычно уничтожаются. Таким образом, для реализации данной потребности пользователя система должна сохранять таблицу символов, построенную компилятором. Кроме того, требуемые данные должны быть представлены пользователю в формате исходного языка, для чего необходимы некоторые из обычных средств компилятора. Таким образом, для отображения или модификации вычисленных значений в ответ на оперативные запросы пользователя необходимы средства, которые не явля-

ются обязательными и по этой причине обычно отсутствуют в компиляторах пакетных систем.

Обеспечение реакции на синтаксические ошибки пользователя (п. 6 на рис. 10.2) является существенным для любой программной системы независимо от того, работает она с разделением времени или в пакетном режиме. Трудно однозначно ответить на вопрос: присущи ли системам с разделением времени какие-либо специфические особенности в решении задачи синтаксической проверки. Некоторые убеждены (и весьма искренне), что каждая строка программы после ввода должна быть подвергнута синтаксическому контролю и о каждой ошибке сразу же должно быть сделано сообщение пользователю. Другие уверены (не менее искренне), что вполне достаточно и даже предпочтительно ограничиваться, как это обычно делается, приведением краткого перечня ошибок после текста программы. Возможно, этот вопрос не будет никогда решен окончательно, ибо многое зависит от профессиональных навыков конкретных людей. Созданные к моменту написания данной работы (1975 г.) некоторые из наиболее удачных систем с разделением времени, такие, как APL и CMS фирмы IBM, не обеспечивают построчного контроля синтаксиса, тогда как другие выполняют эту проверку. При написании компилятора с самого начала не очень трудно предусмотреть средства синтаксического контроля, но требуются нетривиальные изменения, чтобы модифицировать компилятор, в котором не было предусмотрено этой возможности.

В качестве итога рассмотрения данного вопроса отметим, что вполне приемлемое для многих случаев взаимодействие человека и ЭВМ обеспечивают быстрые компиляторы, даже разработанные для систем пакетной обработки, если использовать их совместно с качественными подсистемами редактирования и обработки файлов, соответствующими требованиям инженерной психологии. Однако такая система не будет иметь некоторых, если и не главных, то весьма желательных качеств, необходимых для выполнения отладки, а в особенности для модификации вычисляемых данных, ссылки на которые осуществляются в терминах исходного языка, и для возобновления прерванной программы.

Все желательные функции, как уже описанные, так и некоторые другие, могут быть реализованы наиболее рационально и для пользователя, и для разработчика транслятора, если транслятор представляет собой не **компилятор**, а **интерпретатор**. Интерпретатор транслирует каждый оператор исходной программы каждый раз, когда этот оператор выполняется, а компилятор преобразует всю программу пользователя в машинную (объектную) программу за время специальной фазы

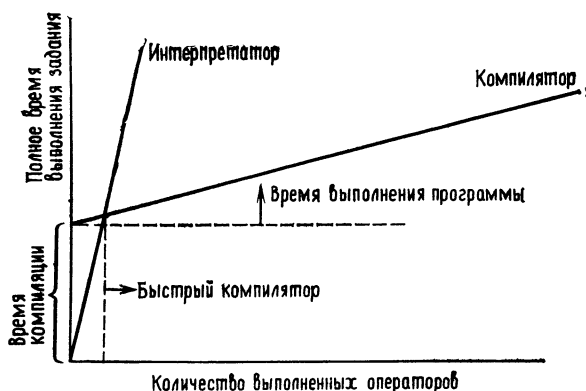


Рис. 10.3. Сравнительные характеристики компиляторов и интерпретаторов.

трансляции, и только после этого выполняет сформированную объектную программу. Интерпретаторы обычно тратят больше времени на повторную трансляцию операторов, чем на их выполнение. Характеристики относительной производительности компиляторов по сравнению с интерпретаторами показаны на рис. 10.3. Из рисунка видно, что вследствие значительных затрат времени на работу компилятора интерпретатор при выполнении очень коротких программ (особенно, если эти программы не содержат циклов) работает быстрее, но при выполнении почти любой программы, содержащей циклы, более эффективным оказывается использование компилятора. В то же время, поскольку интерпретатор транслирует операторы по мере их выполнения, при работе с ним легче внести изменения в программу, так как отсутствует необходимость повторной компиляции программы после внесения изменений. Кроме того, применение интерпретатора позволяет в какой-то степени расширить возможности используемого языка программирования благодаря тому, что при внесении изменений в программу достаточно в этом случае располагать информацией о текущем распределении памяти и использовании ресурсов, тогда как компилятор должен получить соответствующие сведения в виде сообщений пользователя или по установленным правилам до начала выполнения программы в процессе ее трансляции. Другими словами, интерпретатор требует, чтобы до самого конца работы программы поддерживалась связь имен и параметров с машинными ресурсами в форме, удобной для пользователя. В действительности компиляторы и интерпретаторы различаются не столь сильно, как это может показаться из только что приведенных соображений, имеющих целью подчеркнуть разницу между этими двумя типами трансляторов.

Однако чем в большей степени используются методы интерпретации при реализации компиляторов, тем удобнее работать пользователю, но тем большим оказывается время выполнения.

Невозможно переоценить те обстоятельства, что основные средства человеко-машинного взаимодействия, к числу которых относятся возможности редактирования программ и упорядочения данных, весьма часто реализуются на практике и, как правило, с весьма небольшими затратами вычислительного времени работы системы. Благодаря этим особенностям данные средства получили название **тривиальных трактов**. Это название в некотором смысле неудачно, так как на самом деле именно им присуща значимость, определяющая первоочередность их обслуживания. Другими словами, быстрая реакция на тривиальные запросы не только возможна, но и необходима.

Требование быстрой реакции на тривиальные запросы может рассматриваться как одно из конкретных проявлений более общего принципа психологии взаимодействия человека с машиной:

Принцип разработки программ ответа на запросы

*Систему следует разрабатывать с таким расчетом, чтобы она наиболее быстро отвечала на те запросы, которые по мнению пользователя должны быть обработаны в кратчайший срок при условии, что каждый такой запрос в монопольном режиме может быть обработан за минимальное время.*

Этот принцип может показаться с первого взгляда достаточно очевидным, но, если его принять, он окажется весьма полезным правилом при разработке временного распределения ресурсов. Таким образом, наивысшие приоритеты присваиваются не только тривиальным трактам программ редактирования и вывода на дисплей, и другие относительно быстро выполняемые тракты должны получать приоритет более высокий, чем программы с большим временем выполнения. Поскольку время выполнения часто заранее не известно, планировщик, осуществляющий циклическое обслуживание запросов, может быстро закончить обслуживание коротких трактов, даже если он не может распознать тип тракта. Следует, кроме того, отметить, что пользователь не может рассчитывать на то, что время, требуемое для обслуживания тракта, окажется значительно меньше, чем отводимое в системе для обработки информации этого тракта. В этом случае реакция (ответ) системы не может быть удовлетворительной. Однако приведенный принцип имеет и другой смысл — там, где отмеченная конфликтная ситуация отсутствует, система должна ответить именно таким образом, как от нее ожидает человек.

Наличие сведений об ожидаемой «длительности выполнения», а в некоторых системах и о требуемом объеме памяти, может значительно повысить эффективность работы планировщика. Хотя при работе циклического планировщика эти сведения не используются, действия планировщика такого типа в его простой форме могут служить причиной ощутимых задержек при свопинге, достаточных, чтобы свести на нет некоторые преимущества его применения. Вопрос об априорном знании времени обработки трактов является одним из кардинальных. В специализированных системах, где используется один или небольшое количество специфичных языков и процессоры редактирования файлов, разработчики и планировщик могут легко получить информацию о ресурсах, которые требуются для обслуживания трактов, принадлежащих к важным классам. Например, разработчики системы RAX фирмы IBM, в составе которой используется один компилятор с языка Фортран, снизили расходы времени, связанные со свопингом, спроектировав систему таким образом, что в процессе работы компилятора свопинг не производится, в результате чего в большинстве случаев полное время компиляции было оценено как достаточно короткое, чтобы не оказывать значительного влияния на обслуживание тривиальных запросов. Однако система RAX осуществляет разделение во времени и свопинг при выполнении объектных программ, поскольку длительность их выполнения не предсказуема.

Чем более универсальной является система с точки зрения разнообразных входящих в ее состав трансляторов, средств редактирования файлов и т. д., тем труднее оказывается применять априорные сведения об использовании ресурсов. Поэтому при функционировании более универсальных систем должны в большей степени учитываться сведения, получаемые в процессе этого функционирования, и выполняемые при этом замеры используемых ресурсов, чтобы распределять эти ресурсы наиболее удачным образом.

### 10.3. ВЫБОР КВАНТА ВРЕМЕНИ

Основной целью квантования времени является обращение к операционной системе, чтобы заставить ее пересмотреть распределение ресурсов. Это очень важно для обеспечения приемлемых значений большинства параметров, характеризующих производительность обслуживания запросов, так как переменные планирования и, следовательно, приоритеты часто меняют свои значения во времени. Вследствие того что вычисление этих переменных и их относительных рангов должно выполняться на той же ЭВМ, которая обслуживает и пользователей,

приходится время от времени (с интервалом квантования) прерывать работу ЦП для выполнения этих вычислений и для принятия решений о распределении ресурсов на основе результатов этих вычислений.

Если пренебречь затратами на обновление значений переменных планирования и переключение заданий, то интервал квантования следовало бы делать очень коротким, чтобы обеспечить быструю реакцию на изменения значений планируемых переменных. Однако реальное переключение ЦП с одного тракта на другой складывается из двух переключений: 1) от текущей программы пользователя к операционной системе (планировщик) и 2) от операционной системы к программе пользователя. Во многих системах первый шаг осуществляется достаточно быстро. Второй шаг тоже может быть реализован быстро, если происходит возврат в ту же программу, которая фигурировала на шаге 1. Если же на втором шаге в работу включается новая программа, причем она должна быть перемещена в основную память из вспомогательной, а этому должно предшествовать удаление из основной памяти программы, находившейся там до этого, то переключение заданий может потребовать значительного времени. При этом затраты времени на переключение заданий могут сказаться на нижнем пределе допустимых значений кванта времени. Однако в системе, как правило, присутствуют два класса заданий, на выполнение которых система может переключаться: задания, которые в данное время находятся в ОП (резидентные задания), и задания, которые в данное время не являются резидентными. То обстоятельство, что затраты на переключение для заданий первого класса значительно меньше, чем для второго, наводит на мысль использовать **два кванта времени**. Некоторые системы имеют изменяемые кванты времени, но в таких случаях, когда максимальный квант времени имеет достаточно большую длительность (например, несколько секунд), работа ЦП может прерываться для обслуживания других новых трактов, поскольку они могут оказаться принадлежащими к критической категории тривиальных запросов.

Подводя итоги, отметим, что слишком большая длительность кванта времени определяет медленную реакцию системы на запросы, а слишком малое значение этого времени увеличивает накладные расходы времени работы ЦП на переключение задач. Для того чтобы уменьшить объем свопинга, являющегося еще одним препятствием к повышению эффективности системы, и одновременно попытаться сохранить как можно большую чувствительность к запросам, часто прибегают к использованию различных квантов времени для периодов времени, в течение которых программа пребывает в основной



памяти, до тех пор пока она не будет удалена оттуда свопингом, и для времени работы ЦП, предоставленного одному тракту до переключения на другой тракт, программа обслуживания которого находится в основной памяти. Размер кванта (квантов) времени может быть установлен на этапе проектирования системы, задан как параметр, определяемый системным программистом или оператором, а также автоматически рассчитан операционной системой на основе ее собственных наблюдений за деятельностью заданий и системы.

#### 10.4. СИСТЕМА MIT CTSS

CTSS (Compatible Time-Sharing System) — совместимая система с разделением времени, разработанная в Массачусетском технологическом институте в начале 60-х годов, является одной из первых систем, подробно описанных в литературе. Как и предполагает ее название, эта система организована таким образом, что любая программа, выполняемая на обычной ЭВМ IBM 7090, может быть выполнена без каких бы то ни было изменений в системе с разделением времени. Эта особенность (которая желательна, но не обязательна для систем с разделением времени) дает пользователям возможность разрабатывать, отлаживать и модифицировать программы в системе с разделением времени, будучи уверенными при этом, что их программы будут выполняться как задания в системе пакетной обработки, где более эффективно используется машинное время. Реализация принципа совместимости, кроме того, означает, что все ресурсы, в частности вся основная память, которые доступны пользователю при работе в пакетном режиме обработки, могут быть использованы и при работе с разделением времени.

На рис. 10.4 показана конфигурация аппаратных средств системы, которая содержит один ЦП IBM 7094 и два блока основной памяти на ферритовых сердечниках по 32К слова с быстродействием 2 мкс. Один из этих блоков использовался для одновременной работы не более одной программы и, следовательно, удовлетворял требованию совместимости, поскольку объем основной памяти стандартной системы 7094 составляет всего 32К слов. Другой блок памяти емкостью 32К слов использовался для системных программ CTSS. Как и в большинстве других систем IBM 7090/7094, в данной системе для мультиплексирования и управления также применена специализированная ЭВМ IBM 7750. Обмен информацией между блоками основной памяти и внешней памятью на барабане или диске осуществляется с помощью быстродействующего канала.

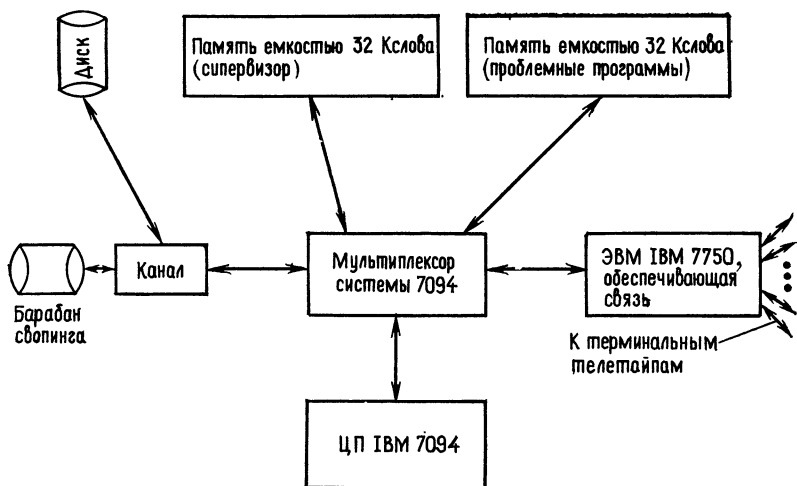


Рис. 10.4. Структура системы MIT CTSS.

При работе в режиме с разделением времени барабан является главным устройством, обеспечивающим свопинг.

Свойство совместимости обеспечивает пользователю наибольшим доступным пространством памяти и устраняет многие проблемы планирования и управления, возникающие в системах, где управление ресурсами осуществляется более гибко. В связи с тем что при использовании CTSS в основной памяти может находиться одновременно только одна программа пользователя, в этой системе отсутствует разделение памяти и необходимость в динамическом перемещении программ. ЭВМ IBM 7090, на которой первоначально была применена система CTSS, обладала этими специально предусмотренными особенностями, но использованы они не были. Еще одним следствием пребывания в основной памяти в любой момент времени только одной программы является невозможность совмещения свопинга одного пользовательского задания с работой ЦП над другим заданием. Как и во многих системах, для сокращения времени свопинга система CTSS осуществляла его максимальными по размеру блоками памяти емкостью 32К слов.

Хотя система CTSS не решала задачи планирования, характерные для более поздних систем с разделением времени, она выполняла свою главную функцию планирования, заключающуюся в принятии решения, какой тракт должен быть обслужен следующим и в течение какого времени, причем делала это изящно и оригинально. На рис. 10.5 представлена схема алгоритма «экспоненциального» планирования CTSS, который

1. *Девять уровней приоритета; на каждом уровне может быть несколько заданий*

Высший приоритет	0
	1
	2 Новые задания: память $< 4095$ слов
	3 Новые задания: память $\geq 4095$ слов.
	4
	5
	6
	7
Низший приоритет	8

2. *Алгоритм планирования задания*

(а) Для следующего обслуживания центральным процессором выбирается тракт из непустой очереди уровня наивысшего приоритета.

(б) Тракт, не выбранный в течение 60 с, перемещается в следующую очередь с более высоким приоритетом.

(в) Один из трактов на уровне  $q$ , выбранный для обслуживания, выполняется до тех пор, пока не появится первый запрос на ввод-вывод, завершится задание или пройдет  $0,5 \times 2^q$  с. Самый большой интервал времени 128 с (для уровня 8).

(г) Если один из трактов не завершается к концу предоставленного ему интервала времени, он перемещается в очередь с приоритетом, меньшим на единицу.

(д) Тракт, выполняемый в течение выделенного ему интервала, при свопинге его из ОП получает преимущество в новом уровне  $q$ , если текущая работа выполнялась по крайней мере  $0,5 \cdot 2^q$  с. Это задание возвращается в очередь, имеющую приоритет на 1 ниже очереди, в которой оно находилось.

Рис. 10.5. Экспоненциальный планировщик CTSS.

кратко может быть описан следующим образом. Планировщик присваивает **новому** поступающему запросу высокий приоритет, поскольку он может оказаться коротким интерактивным запросом, при обслуживании которого наиболее важна быстрая реакция. В этом случае тракт будет обслужен быстро. Если выясняется, что для обслуживания тракта требуется все больше и больше времени, его приоритет делается все ниже и ниже. Тем не менее, когда приоритет тракта понижается, что делает менее вероятным его выбор для обслуживания, задаваемый при этом период его обслуживания (квант времени) делается более длинным, чем для трактов с более высоким приоритетом. Этот прием способствует сокращению объема свопинга, во время выполнения которого ЦП не функционирует, для заданий, требующих для обслуживания большее время работы ЦП; тем самым повышается производительность системы. Подводя итоги, отметим, что действия CTSS опирались на предположение, что новое задание является коротким и обслуживать его надо первым. Задание, задержавшееся в основной памяти и тем самым потребовавшее более длительного времени обработки, пересылается в фоновый раздел, в котором обра-

ботка в большей степени напоминает пакетный режим. Одно из усовершенствований рассмотренного выше принципа работы системы заключалось в попытке грубого ранжирования заданий по длительности выполнения при их поступлении в предположении, что имеет место корреляция между необходимым для задания объемом памяти и (неизвестным) временем его выполнения. При этом если для размещения задания требуется больше 4095 слов памяти, первоначально присваиваемый ему приоритет оказывается ниже приоритета заданий, требующих меньшей памяти.

## 10.5. СИСТЕМА APL

Система APL (A Programming Language) получила свое название в книге К. Иверсона, в которой этот мощный и элегантный язык программирования был описан и проиллюстрирован на широком классе разнообразных алгоритмов. Этот язык по крайней мере за шесть лет до появления первой машинной реализации транслятора интенсивно использовался на «бумаге» небольшой, но весьма целеустремленной группой специалистов. В отличие от большинства других языков программирования APL прошел долгий период тщательного концептуального развития до того, как техническая реализация с присущей ей требовательностью к рациональности и совместимости положила предел его совершенствованию.

Хотя язык Иверсона не разрабатывался специально для систем с разделением времени, он идеально соответствует проблематике человеко-машинного взаимодействия благодаря идеологии его построения, которая не только делает возможным, но и способствует стремлению пользователя задавать большую по объему обработку посредством минимального количества символов и, следовательно, наименьшего числа нажатий на клавиши. Каким образом это получается, станет яснее при рассмотрении следующих особенностей языка.

1. Имя может обозначать массив, т. е. вектор, матрицу или многомерную структуру, размеры которых разрешается уменьшать или увеличивать без специальных действий пользователя.

2. Существует большое число операторов, обозначенных при помощи собственных специальных символов или комбинаций простых символов, которые применимы как ко всему массиву, так и к обычным отдельным значениям или индексированным переменным (элементам массива). К их числу относятся весьма обобщенные операторы матричной алгебры и многие другие операторы.

3. Язык имеет три типа данных (число, бит и символ); предусмотрены средства автоматического взаимного преобра-

зования друг в друга битов и чисел, а также целых чисел и чисел с плавающей точкой без явного вмешательства пользователей.

Значимость многих этих возможностей для обеспечения человеко-машинного взаимодействия была продемонстрирована в 1964 г. на примере «пилотной» системы, в которой использовались многие идеи языка APL. За годы, предшествующие реализации, этот язык постоянно развивался и подвергся дальнейшему усовершенствованию даже в процессе реализации, когда, например, было окончательно сформулировано правило прямого просмотра операторов языка справа налево. Реализация языка APL была выполнена под полным контролем К. Иверсона и А. Фалькоффа из исследовательского центра Ватсона фирмы IBM в Йорктауне.

Транслятор IBM/360 является интерпретатором; он был разработан менее чем за год в основном двумя программистами Л. Бридом и Р. Лафвеллом. Эта пара молодых «волшебников» с помощью Р. Мура, кроме того, разработала математическое обеспечение для терминальной обработки и свопинга, а также язык простейших команд, обеспечивающий пользователю доступ к ограниченному объему дискового пространства. Более ранняя система APL выполнялась в середине 60-х годов под управлением системы DOS, основной операционной системы IBM, используемой в малых и средних моделях IBM/360. Позже (около 1968 г.) APL выполнялся под управлением более мощной операционной системы ОС/360. Хотя многие ЭВМ моделей 40 и 50 с небольшой основной памятью емкостью около 250 Кбайт обслуживали до 50 пользователей APL, система больше подошла к модели 50 (или модели с большим быстродействием ЦП) с емкостью основной памяти по меньшей мере 500 Кбайт. Появилась возможность выполнять в ОС программы с пакетной загрузкой, поскольку при 50 и более терминалах APL время реакции на тривиальные запросы будет находиться в диапазоне 3 с, удобном для пользователя.

APL можно рассмотреть в двух аспектах. Первый аспект — это язык и его транслятор. Они могут быть проанализированы без учета разделения времени. Второй аспект связан с задачами распределения ресурсов между пользователями, такими, как задача свопинга.

Для того чтобы помочь пользователю совместить по мере возможности изящность языка с удобством его использования, было решено применить транслятор интерпретирующего типа. Почти при любом использовании языкового компилятора требуются описания и другие объекты языка, доставляющие пользователю неприятности. Однако время работы интерпретатора довольно велико, поскольку он должен транслировать

каждый оператор перед выполнением (вместо того, чтобы транслировать его только один раз в программе, как это делает компилятор). Этот существенный недостаток интерпретатора в данном случае ослаблен (по сравнению с другими языками), поскольку APL благодаря мощным операторам обработки массивов имеет тенденцию роста времени выполнения каждого оператора. Тем не менее циклы заметно ухудшают характеристики программ. Эффективность использования памяти при применении интерпретирующего метода трансляции для APL оказывается выше, чем при применении компилятора. APL использует одну резидентную копию интерпретатора, разделяемую всеми пользователями. Это требует пространственной совместимости памяти, что может быть использовано при проектировании резидентного компилятора с быстрой реакцией на запрос. Однако реальная экономия памяти обеспечивается благодаря тому, что этот язык позволяет разрабатывать короткие программы для задач, которым на других языках соответствуют более длинные программы. Для работы интерпретатора в каждый момент времени требуется только небольшая сверхоперативная память, достаточная для размещения одного оператора, тогда как при использовании компилятора необходимо хранить всю объектную программу.

Представляет интерес взаимосвязь между временем и памятью в языке APL. Многие операции, которые позволяют наиболее эффективно организовать работу с массивами, обходясь очень компактными операторами, требующими для размещения очень мало памяти и быстро транслирующимися, к сожалению, требуют отведения слишком большой памяти в объектном формате. В качестве крайне простого примера приведем оператор суммирования первых  $N$  целых чисел; на языке APL этот оператор имеет вид:  $+/iN$ . Но этот оператор требует, чтобы транслятор APL хранил  $N$  чисел, в результате чего при больших  $N$  необходимый объем памяти значительно превышает объем, который потребовался бы, если бы та же задача была запрограммирована с использованием цикла. Однако в APL цикл менее удобен для программирования и вычисления выполнялись бы намного медленнее.

Эти соображения могут быть с успехом использованы для анализа всевозможных особенностей транслятора APL как полезных, так и неприятных, определяющих его производительность. Было бы разумно утверждать, что при использовании транслятора интерпретирующего типа пользователь расплачивается значительными затратами времени за удобство работы с языком, а уменьшить эти затраты он может за счет интенсивного использования памяти, которое достигается при широком применении операторов обработки массивов. Конечно, экономия

X	Определение $iX$
19	Общее время работы с разблокированной клавиатурой <sup>1)</sup> во время сеанса
20	Время дня
21	Продолжительность использования ЦП во время сеанса
22	Число байтов, оставшихся в рабочей области
23	Число терминалов, подключенных в текущий момент времени
24	Время начала сеанса
25	Дата <sup>2)</sup>
26	Первый элемент $i27$
27	Номера операторов в векторе состояния

Примечания: 1. Все значения времени в пределах от 1 до 60 с.

2. Дата представлена шестизначным целым числом, по две цифры на месяц, день и год.

Рис. 10.6. Некоторые  $i$ -функции APL.

памяти важна только тогда, когда отведенный объем оказывается исчерпанным. Пока памяти достаточно, и это особенно справедливо для небольших задач, кратчайшие программы, содержащие минимально возможное количество циклов, являются наиболее удобными и вполне приемлемы с точки зрения быстродействия.

Одной из интересных и важных принадлежностей APL является группа операторов « $i$ -функций». Некоторые из этих функций, которые могут быть использованы при работе на любом терминале и включены в любую программу, приведены на рис. 10.6. Определенные  $i$ -функции могут иметь большое значение для отдельного программиста с точки зрения оценки и улучшения возможностей его программ. К ним относятся  $i22$ , которая указывает объем памяти, остающийся незанятым, в отведенной пользователю рабочей области, и  $i21$ , посредством которой производится считывание показаний его часов из ЦП. Пользуясь этими часами, можно определить время, затрачиваемое ЦП на выполнение любой части программы, как переменную системы APL. Заметим, что время, затрачиваемое ЦП на программу, определяется только характеристиками тактируемой программы и не связано с совместным использованием системы многими пользователями. Другие  $i$ -функции с номерами 14 и меньше представляют собой ссылки на векторы распределений частот различных типов работы, выполняемых системой. Эти  $i$ -функции могут быть использованы только при работе на «привилегированных» терминалах, чтобы экономнее расходовалась ценная основная память. Таким образом,  $i$ -функции служат для каждого пользователя средством реализации важных для его работы аспектов применения и характеристик системы.

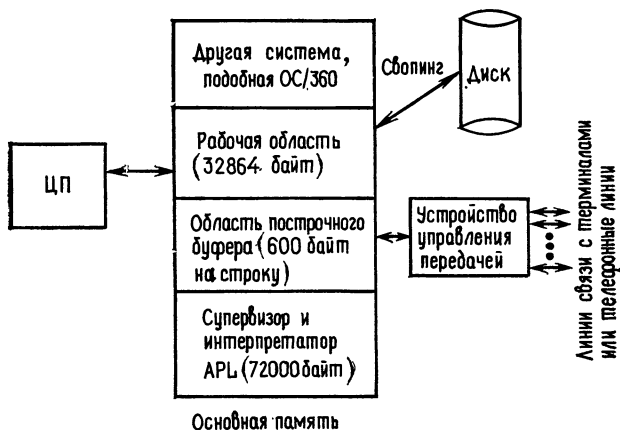


Рис. 10.7. Типичные ресурсы системы APL.

Рассмотрим теперь APL в аспекте, связанном с обслуживанием множества пользователей. На рис. 10.7 показана структура основной памяти. Небольшая область в основной памяти, имеющая объем около 600 байт, зарезервирована для буферизации сообщений, поступающих от терминала **каждого** пользователя. Поскольку обмен информацией между этой памятью и терминалом осуществляется с помощью мультиплексного канала машины, который может работать независимо от ЦП, возможно выполнение нескольких таких обменов параллельно с обработкой данных ЦП. Это значит, что пользователь может вводить программу или данные или выводить их по крайней мере в течение ограниченного времени даже тогда, когда его рабочая информация уже выведена из основной памяти во внешнюю. В системе предусмотрена возможность совмещения во времени работы ЦП и терминального обмена даже для одной и той же программы (а также и между программами). Таким образом, программа, которая повторяет последовательность «вычислить строку/вывести строку», выводит на терминал результат вычислений со скоростью печатающего устройства при условии, что объем вычислений небольшой и количество запросов на обслуживание к ЦП тоже не слишком велико.

Одна копия транслятора APL в основной памяти совместно используется всеми пользователями. В описании конфигурации системы может быть указано, что в основной памяти могут быть одновременно две или больше рабочих области по 32 Кбайт каждая. Свопинг осуществляется между этими областями и дисковой памятью IBM 2311 или 2314/3330. В связи



с тем что в основной памяти может иметь место одновременно более одной рабочей области, возможно совмещение ввода и вывода с участием внешней памяти с вычислениями (для разных заданий APL). Одно из ограничений систем APL, которые поставлялись фирмой IBM в период около 1972 г., состояло в сужении рабочих областей пользователей до фиксированных размеров (обычно до 32 Кбайт). Быть «активным», т. е. иметь доступ к своей программе, может только один пользователь. Для обращения к неактивной рабочей области необходимо ручное исполнение команды. Это серьезное практическое ограничение снято по крайней мере в одной из систем APL, а именно в системе SV (Shared Variable — коллективная переменная), выпущенной в 1973 г. для ЭВМ IBM/370, которая, кроме того, также обеспечивает возможность доступа к файлам. Стоит отметить, что APL выполняется на любой стандартной ЭВМ типа IBM/360 или 370, имеющей соответствующую конфигурацию. Для установки APL не требуется никакого специального, нестандартного оборудования<sup>1)</sup>.

Исключительно высокая надежность системы проявилась уже на первых этапах применения APL. Отчасти она является результатом того, что для разработки и отладки системы разработчики и проектировщики пользовались ею. Система APL обычно эксплуатируется в «автономном» режиме, т. е. без участия оператора машинного зала, когда в этом возникает необходимость в нерабочие дни.

## 10.6 ИЗМЕРЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

В этой главе внимание будет сосредоточено на методах измерения, применимых для анализа режима разделения времени, т. е. вычислительной системы, обслуживающей множество терминалов, за которыми работают люди. Такая система, разумеется, может выполнять программы и в пакетном режиме, как часто и имеет место на практике. Существуют два основных класса методов измерений — внутренние и внешние, применение каждого из которых определяет необходимость постановки совершенно разных измерительных экспериментов, предъявляя весьма различные требования к знаниям и опыту лиц, проводящих измерения.

**Внутренний** метод измерений предполагает возможность не только доступа к операционной системе, но и внесения в нее изменений, хотя многие средства измерения параметров

---

<sup>1)</sup> На самом деле требуются нестандартные терминалы с так называемой APL-клавиатурой. Это вызвано необычностью алфавита языка, что, с одной стороны, сделало программы очень компактными, а с другой — очень немобильными. — *Прим. перев.*

включены сегодня в состав операционных систем. Системы с разделением времени, так же как и пакетные системы, часто включают программы интерпретации прерываний, которые перехватывают все сигналы прерывания и сохраняют о каждом из них информацию, содержащую отсчет времени и описание события, вызвавшего прерывание для последующего возобновления обработки тех же данных. Кроме того, система с разделением времени должна отслеживать действия каждого из пользователей терминалов. Так как объем этих данных обычно гораздо больше, чем для пакетных систем, необходимо больше внимания уделять тому, чтобы средства и методы измерения существенно не снижали производительность исследуемой системы. В описаниях почти всех известных систем с разделением времени содержится описание заложенных в нее средств измерения. Впрочем, можно заметить, что почти всегда **любое** внутреннее программное средство измерения характерно именно для **конкретной** операционной системы. Знание внутренней структуры системы, представляющее собой важный, а зачастую и обязательный элемент успешной ее разработки, а также являющееся полезным для персонала, занимающегося вводом ее в эксплуатацию, обычно необходимо и для пользователей внутренних измерительных средств. Необходимость знания этой структуры не вытекает из сущности самих внутренних методов, и в принципе данные о производительности, полученные этим способом, могут быть представлены в форме, не требующей знания внутренней структуры системы. Но по крайней мере в настоящее время это утверждение относится к области скорее желательного, нежели действительного.

При измерении **внешними** методами система рассматривается как «черный ящик», исследование которого осуществляется только посредством пользовательских терминалов. Внешние методы в отличие от внутренних существенно зависят от особенностей рабочей нагрузки и от способа представления системы пользователю, но не требуют ни подробного знания внутреннего поведения системы, ни доступа к внутренним элементам операционной системы.

Производительность любой системы, в том числе и системы с разделением времени, определяется факторами трех типов: 1) рабочей нагрузкой; 2) структурой системы (ее программными и аппаратными средствами); 3) параметрами, используемыми в качестве показателей производительности.

Поскольку рассматривается проблема измерения производительности системы, необходимо, конечно, познакомиться с самой системой. Таким образом, будем считать, что структура системы определена полностью и нам останется лишь тщательно описать ее в отчете о проведенных измерениях. Формули-

ровка третьего пункта — показатели производительности — также не должна представлять значительных трудностей. Так, например, достаточно информативным является следующий набор критериев производительности:

1. Время ответа, описываемое:

а) средним значением, медианой, среднеквадратичным отклонением;

б) распределением вероятностей (в диапазоне времен выполнения).

2. Использование ресурсов центральной системы, т. е. вектор, компоненты которого описывают загрузку во времени (в процентах) каждой составной части системы — ЦП, каналов и т. д. — в период выполнения рабочей нагрузки.

Наиболее трудной из перечисленных характеристик для измерения является **рабочая нагрузка**, поэтому наиболее значительная часть данного раздела посвящается именно ей. Источник этой трудности достаточно ясен. В связи с тем что рабочая нагрузка в системе с разделением времени задается персоналом, работающим в непосредственном взаимодействии с системой, поведение этого персонала за терминалами представляет собой неотъемлемую часть характеристики рабочей нагрузки. Одно из прямых следствий этого обстоятельства, представляющее собой неприятную неожиданность для инженера, но известное специалистам по теории поведения, заключается в том, что получить воспроизводимые результаты при наличии человека-пользователя работающей системы весьма затруднительно. Однако статистически воспроизводимые результаты все же получить можно. Если, например, измерения производить на одной системе в одно и то же время в течение ряда дней, то, хотя деятельность каждого пользователя в деталях будет почти наверняка различаться, в распределении совместной деятельности всех пользователей, возможно, будет наблюдаться закономерность. Конечно, это не абсолютная истина, а, скорее предположение, которое необходимо проверить путем проведения измерений. Наблюдение за работой реальной системы достаточно просто осуществить внутренними измерительными средствами. Гораздо труднее использовать для этого внешние методы измерения, хотя и с ними можно получить некоторую информацию о производительности, пользуясь **терминалом администратора системы**. Терминал (терминалы) администратора системы представляет собой обычный терминал пользователя, за которым работает лицо, производящее измерения, по запросам которого осуществляются работы известного и измеримого объема, такие, как последовательность интерактивных взаимодействий или компиляция и выполнение определенных заданий. Другие пользователи в это время работают, как обыч-

но. Администратор системы измеряет реакцию системы на свои запросы и регистрирует, как она изменяется, например, в течение дня или в зависимости от характера запросов. Возможно, он также пожелает сравнить эти временные характеристики с соответствующими характеристиками других систем (желательно, с той же рабочей нагрузкой терминала администратора).

Характеристика системы с разделением времени, опирающаяся на исследование ее поведения при рабочей нагрузке, базируется на понятии **сценария**. Сценарий состоит из записей, представляющих собой описания взаимодействия реальных или воображаемых пользователей с системой, с регистрацией времени изменения каждого элемента каждой записи. Чтобы получить представление о сценарии отдельного пользователя, рассмотрим сеанс работы за терминалом с печатающим устройством, в ходе которого каждая строка снабжается отметкой момента времени, когда она появляется на терминале при вводе или выводе. Сценарий измерительного эксперимента представляет собой набор таких отдельных сценариев каждого пользователя. После того как все отдельные сценарии скомпонованы и обеспечена их одновременная реализация, можно воспроизвести ответ системы на такой объединенный сценарий. Этот же сценарий может быть использован как описание рабочей загрузки для исследования изменений в конфигурации или в операционной системе. После этого сценарий становится исходной информацией для имитатора системы.

Задача построения сценария аналогична задаче выбора какой-либо контрольной задачи. Обсуждение этих вопросов в наши намерения не входит; мы просто будем считать, что сценарий имеется. Хотя данные сценария в принципе достаточно просты, даже в том случае, когда имеется приемлемый сценарий, средства его представления и применения могут оказаться довольно громоздкими. Так как объем содержащейся в сценарии информации может быть большим и при его реализации должна быть обязательно выдержана временная последовательность, ручной ввод возможен только в случае простейших сценариев. Несколько ниже мы рассмотрим машинное представление сценария. Однако здесь целесообразно сделать небольшое отступление и предположить, что ручной ввод в систему очень простых сценариев является вполне реальным и экономически оправданным путем следующих действий.

#### **1. Предтестовая подготовка**

- а) Выбрать и отладить ряд тестовых программ, записать их в несколько тестовых файлов пользователя. Типичный сценарий содержит описание действий лиц, вызывающих тестовую программу (см. ниже).

б) Договориться о времени использования системы для выполнения исследований.

в) Подготовить группу людей, которые будут непосредственно заниматься вводом сценариев в период тестирования. Они (как пользователи) должны быть в какой-то степени знакомы с исследуемой системой.

## 2. Некоторые типичные тесты измерения

а) Тест ответа на тривиальный запрос: имеется  $j$  пользователей, которые одновременно выдают тривиальные запросы со скоростью, на которую они способны; с помощью секундомеров измеряется время обработки запросов. Испытания проводятся для  $j = 1, 2, 3$  и т. д.

б) Тест ответа на запросы программ: имеется  $j$  пользователей, которые одновременно вызывают одни и те же программы и замеряется время обработки запросов. Тест повторяется для  $j = 1, 2, 3$  и т. д.

в) Смесь тестов ответов на тривиальные и нетривиальные запросы.

Хотя приведенные тесты достаточно просты, их практическое выполнение требует достаточно сложной координации действий людей. Однако при их реализации нет необходимости применять специальное оборудование или вмешиваться в систему. При обоснованном выборе тестовых программ данный метод позволяет быстро выявить многие характеристики производительности и факторы, препятствующие ее повышению.

Необходимо и вполне реально представить сложные сценарии в формате, допускающем их чтение машиной, чтобы ЭВМ могла вводить их в исследуемую систему. Процедура преобразования информации сценария в формат, который обеспечивает выдачу запросов на обслуживание и запись ответов системы, можно представить в виде программы, для которой сценарий является исходными данными. Можно даже возложить выполнение этой программы на исследуемую систему; в этом случае системные ресурсы не только разделяются между сценариями тестов, но и частично используются самими средствами исследования системы. При этом возникает несколько очевидных проблем:

1. Ресурсы исследуемой системы (особенно основная память) сокращаются за счет выполнения тестовых программ.

2. На процедуры тестирования расходуется время работы исследуемой системы.

3. Терминалы, на которые передаются выходные данные сценария, могут оказаться занятыми.

4. Возникает необходимость в программных интерфейсах, обеспечивающих разделение ресурсов ЭВМ между программой

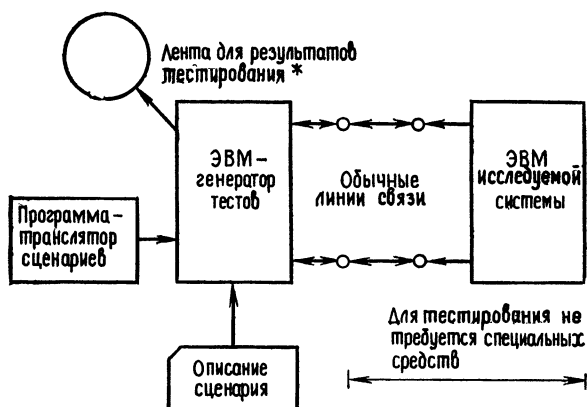


Рис. 10.8. Использование одной ЭВМ для тестирования другой ЭВМ.

\* Лента с результатами тестирования обрабатывается позже (так что ЭВМ-генератор тестов во время работы сценария может работать достаточно быстро).

генерации теста и фиксации результатов тестирования, с одной стороны, и исследуемой системой — с другой.

Первые два и последний пункты в комментариях не нуждаются. Третий заслуживает некоторых пояснений. Запросы пользователей в работающей системе обычно поступают по телефонным линиям на мультиплексное устройство управления передачей, которое связано с ЭВМ. Большинство систем построено так, что одна и та же ЭВМ не может генерировать входные данные для этих терминалов. С точки зрения тестирования системы и измерения ее производительности такая особенность является весьма полезной, но многие системы ею не обладают. Разумеется, измерения могут быть все-таки проведены с использованием самогенерируемых тестов, но при этом вырабатываемые запросы должны поступать в какую-то внутреннюю область системы, аналогичную буферно-терминальным областям основной памяти.

При более чисто поставленных измерительных экспериментах недостатки самопроверки устраняются за счет введения дополнительных аппаратных средств в виде отдельной ЭВМ, служащей для подачи тестов и регистрации результатов. Обобщенная структура такой системы представлена на рис. 10.8. Сначала в ЭВМ, предназначенную для генерации тестов, загружается программа, служащая для преобразования сценариев в машинный формат, а также для получения и регистрации ответов на запросы на обслуживание. Затем в ЭВМ-генератор тестов пересылаются сценарии (на картах или лентах). После этого исследуемая система начинает свою обычную работу, причем следует позаботиться, чтобы ни один

реальный пользователь не мог к ней подключиться. Так как реализация сценария всегда начинается с подключения к исследуемой системе, то как только система-генератор теста выдаст свои первые сообщения, исследуемая система начинает на них реагировать. Заметим, что гипотетический наблюдатель поведения исследуемой системы не имеет возможности обнаружить, что на другом конце телефонной линии работают не реальные люди, а ЭВМ-генератор тестов.

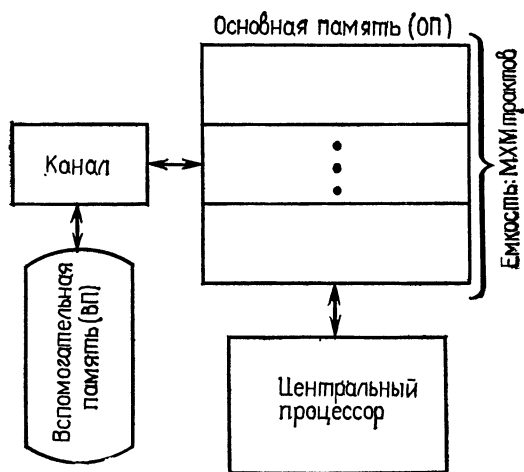
### 10.7. ИМИТАТОР СИСТЕМЫ С РАЗДЕЛЕНИЕМ ВРЕМЕНИ

Теперь рассмотрим простую имитационную модель системы с разделением времени и свопингом рабочих областей памяти и ее применение в качестве конкретного средства исследования определенных характеристик производительности. В сущности эта модель является одним из вариантов обобщенной модели обработки заданий и обладает рядом характерных особенностей систем с разделением времени, таких, например, как свопинг.

На рис. 10.9 показаны ресурсы, моделируемые имитатором, и приведен перечень параметров, которыми пользователь должен снабдить модель. Видно, что в основной памяти (ОП) может одновременно находиться до  $MXM$  трактатов. Односторонний свопинг рабочей области между основной памятью и вспомогательной памятью (ВП) происходит за  $ST$  секунд, а затраты времени ЦП на инициирование свопинга составляют  $PSV$  секунд. В зависимости от того, совмещается свопинг или нет с вычислительными операциями, параметр  $V$  имеет значение 0 или 1. Для идентификации каждого из шести планировщиков служит код  $SC$ , который может осуществлять планирование с квантованием по времени или без квантования в зависимости от значения (1 или 0) переменной  $STS$ . Квант времени обозначается через  $H$ .

При рабочей загрузке прямого класса типа  $I$  каждый тракт описывается временем его возникновения и выполнения. На рис. 10.10 приведена диаграмма состояний, на которой указаны состояния и переходы между ними. Рис. 10.11 поясняет принцип организации структур данных, которые используются в имитационной модели, и вместе с рис. 10.10 позволяет получить некоторое представление о логической структуре имитатора.

В конце каждого кванта времени или в конце обработки тракта планировщик принимает решение о предоставлении тракту следующего кванта времени или свопинге рабочей области в ОП или из нее. Процесс планирования складывается из двух частей с простыми двусторонними связями между



#### Параметры ресурсов

$MXM$  — максимальное количество резидентных трактов, одновременно находящихся в ОП

$ST$  — время одностороннего свопинга

$H$  — квант времени

$PSV$  — затраты времени ЦП на инициирование свопинга

$V=0$  — нет совмещения свопинга с вычислениями

1 — есть совмещение

$STS=0$  — без квантования

1 — с квантованием

$SC$  — код планировщика

$RB$  — величина смещения для заданий, находящихся вне ОП

#### Параметры рабочей загрузки

Для каждого тракта ( $I$ )

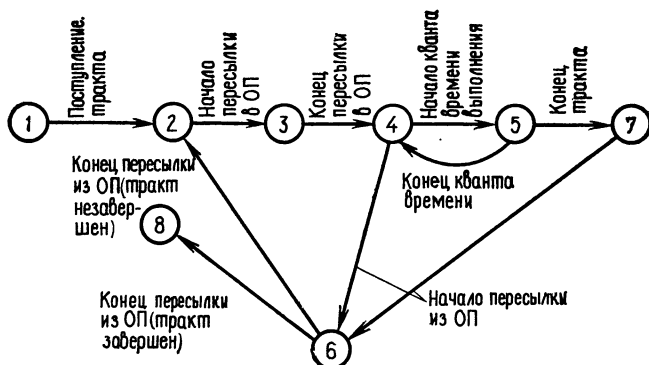
$A[I]$  — время на запрос тракта

$X[I]$  — время выполнения

Рис. 10.9. Имитационная модель.

ними. Первая часть, называемая **правилом назначения обслуживания**, принимает решение о том, какой тракт должен получить следующий квант времени. Для этого отыскивается подходящий для обслуживания тракт с наименьшим значением переменной планирования (предполагается также наличие некоторого разумного **правила выбора**). После выбора тракта для обслуживания может возникнуть одна из двух ситуаций: 1) тракт находится в ОП, 2) тракт не находится в ОП и, следовательно, находится во вспомогательной памяти. В первом случае тракту отводится очередной квант времени. Во втором случае, если в основной памяти имеется свободная рабочая область, то выбранный тракт пересылается из вспомогательной памяти в эту рабочую область. Если свопинг





Состояние тракта:

1. Не поступил
2. Поступил в ВП
3. В канале: пересылка (свопинг) из ВП в ОП
4. Поступил в ОП (ЦП не обрабатывает)
5. Обработка ЦП (в ОП)
6. В канале: пересылка (свопинг) из ОП в ВП
7. Завершение тракта: в ОП
8. Завершение тракта: в ВП

Рис. 10.10. Определение состояний и диаграмма переходов тракта.

совмещается с вычислительными операциями, то квант времени, предназначенный для этого свопинга, планировщик отводит на обслуживание одного из трактов, уже находящихся в ОП, выбранного по признаку наилучшего соответствия его длительности периоду кванта.

В том случае, когда свободной рабочей области в ОП нет, планировщик сначала отыскивает в ОП обслуженный тракт, подлежащий удалению из ОП. Если такой тракт найден, он удаляется из ОП и квант времени, предназначенный для свопинга, опять может быть отведен для обслуживания какого-либо тракта, находящегося в ОП. Если в ОП нет выполненного тракта, то система прибегает к **дополнительному правилу перемещения** для определения тракта, который можно было бы удалить из ОП в ВП и освободить место для тракта, выбранного для обслуживания. Это правило состоит в следующем: к трактам, находящимся в ОП, применяется правило назначения на обслуживание и тракт с **самым низким рангом** выбирается для удаления из ОП. Во время удаления из ОП другой тракт может получить квант времени для обслуживания. После этого свопинга в ОП образуется свободная рабочая область и в действие вступает правило назначения на обслуживание, описанное выше. Дополнительное правило перемещения — это не просто отдельный алгоритм планирования, а

		Тракт →				
		1	2	3	4	и т.д.
Время поступления	A					
Время выполнения	X					
Вектор состояний	S					
Полное время выполнения	P					
Последний квант времени	L					
Время окончания	Q					

(а) Переменные трактов

		АУ Канал	
		1	2
Время окончания	CV		
Обслуживаемые тракты	ID		
Время использования	U		
Требуемое время выполнения	XT		

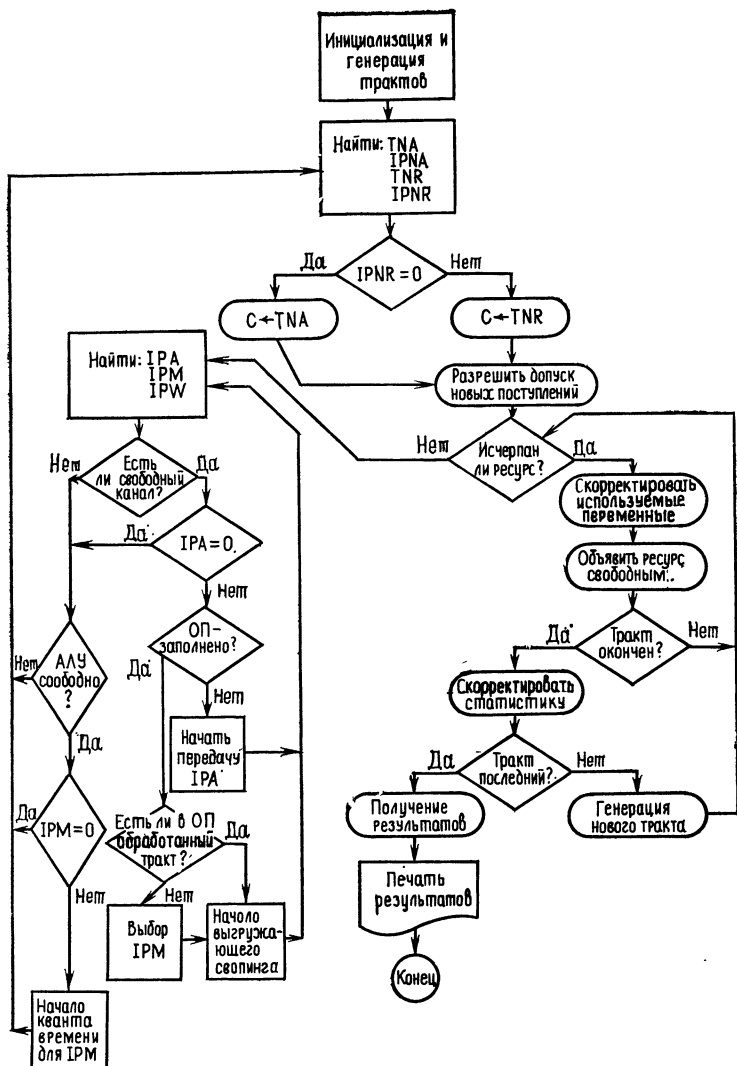
(б) Переменные ресурсов

Рис. 10.11. Некоторые внутренние переменные имитатора.

скорее оператор, регламентирующий выполнение правил перемещения в процессе планирования. Говоря точнее, оно указывает, каким образом на основании данного правила назначения на обслуживание формулируется правило перемещения.

На рис. 10.12 приведена схема алгоритма, иллюстрирующая логику работы имитационной модели. Описанный имитатор является удачным средством для изучения взаимосвязи различных параметров системы. Описание рабочей загрузки можно представить в виде последовательности пар чисел, каждая из которых содержит время поступления и длительность выполнения тракта. Объединяя несколько таких потоков с небольшим числом (скажем, около десяти) трактов в каждом, пользователь может получить наглядное представление об особенностях различных алгоритмов планирования, о последствиях одновременного пребывания в ОП более чем одной рабочей области, о времени свопинга и т. д.

На этом мы завершаем описание логики работы имитатора, с помощью которого посредством варьирования параметров можно имитировать целый класс систем с разделением времени. Как можно использовать этот имитатор? Рассмотрим этот важный вопрос, выделяя два вида работ, на которых должно быть сосредоточено внимание разработчиков любой имитационной модели.



Обозначения:

- TNA* — время следующего поступления
- TNR* — время, когда следующий ресурс будет исчерпан
- IPNR* — ресурс, который в ближайшее время будет исчерпан
- IPNA* — следующий поступающий тракт
- IPA* — лучший (с наивысшим приоритетом) тракт вне ОП
- IPM* — лучший тракт в ОП
- IMW* — тракт с наименьшим приоритетом в ОП
- C* — часы текущего времени

Примечание. Нулевое значение идентификатора тракта означает отсутствие тракта, который удовлетворяет данному условию.

Рис. 10.12. Схема основной логики имитатора.

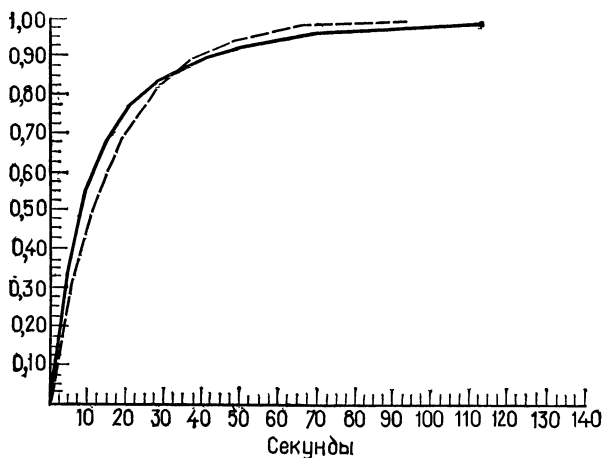


Рис. 10.13. Распределение относительной частоты суммарного времени обдумывания.

— измеренное распределение для системы APL (среднее = 16,73);  
 --- экспоненциальное распределение (среднее = 16,73).

1. Получение данных о рабочей нагрузке для управления имитатором.

2. Оценка достоверности того, что имитатор дает правильные и полезные результаты.

В связи с тем что описанный выше имитатор за счет выбора параметров может обеспечить приближенное описание логики управления ресурсами в системе APL, используем эту систему в рассматриваемом ниже примере.

Любое практическое средство оценки, в том числе и имитатор, должно быть основано на некоторых реальных представлениях о рабочей нагрузке системы, для исследования которой оно будет применяться. Идеальным источником таких данных является программный монитор. Для работы рассматриваемого имитатора требуется указание времени поступления и времени выполнения. Система APL содержит программный монитор, который вырабатывает гистограммы, т. е. распределения частот появления различных событий. Одно из таких распределений — распределение частот суммарного времени обдумывания, т. е. временных интервалов между двумя запросами пользователей (трактами). На рис. 10.13 приведены измеренное и экспоненциальное распределения с одинаковыми средними значениями. Как видно, оба распределения достаточно близки друг к другу.

На рис. 10.14 показана гистограмма, полученная в результате измерения значений времени **выполнения**, совместно с графиком экспоненциального распределения, но здесь соответ-

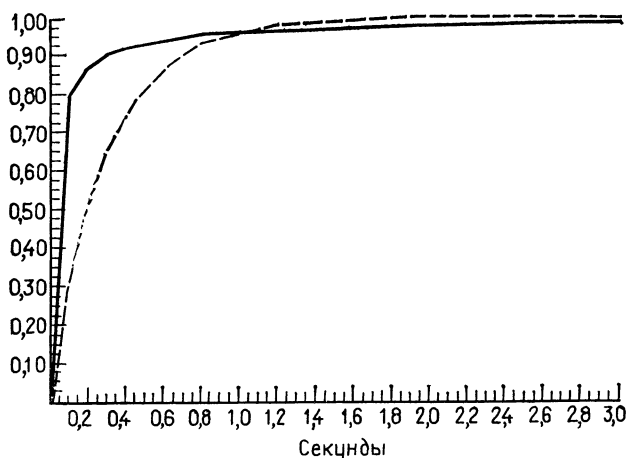


Рис. 10.14. Распределение относительной частоты суммарного времени выполнения.

— — — экспоненциальное распределение (среднее = 0,3);  
 ————— измеренное распределение (среднее = 0,3) для системы APL.

ствие между измеренным и аппроксимирующим распределениями меньше. Это является результатом наличия «длинного хвоста» на графике измеренного распределения времени выполнения, который свидетельствует о том, что значения времени выполнения, в несколько раз превышающие среднее, более типичны для измеренных данных, чем для данных, предсказанных экспоненциальным законом. Так, например, экспоненциальный закон предсказывает, что за время, большее чем 1,8 с (в 6 раз больше среднего значения), выполняется менее 1 % заданий. Результаты измерений показывают, что такое время выполнения имеют около 3 % заданий. Хотя на «хвост» графика распределения приходится лишь малая часть общего количества трактов, он все же часто представляет заметную часть времени выполнения, требуемого на обслуживание всех трактов. Поэтому учет «хвоста» распределения в использовании ресурсов оказывается существенным. Хотя, как показано на рис. 10.14, измеренное распределение не совпадает с экспоненциальным распределением с тем же средним, установлено, что оно хорошо аппроксимируется гиперэкспоненциальной функцией примерно с четырьмя членами.

Функции распределения времени обдумывания и выполнения, приведенные на рис. 10.13 и 10.14, могут быть использованы совместно с программируемым генератором случайных чисел для получения значений времени поступления и выполнения каждого имитируемого тракта.

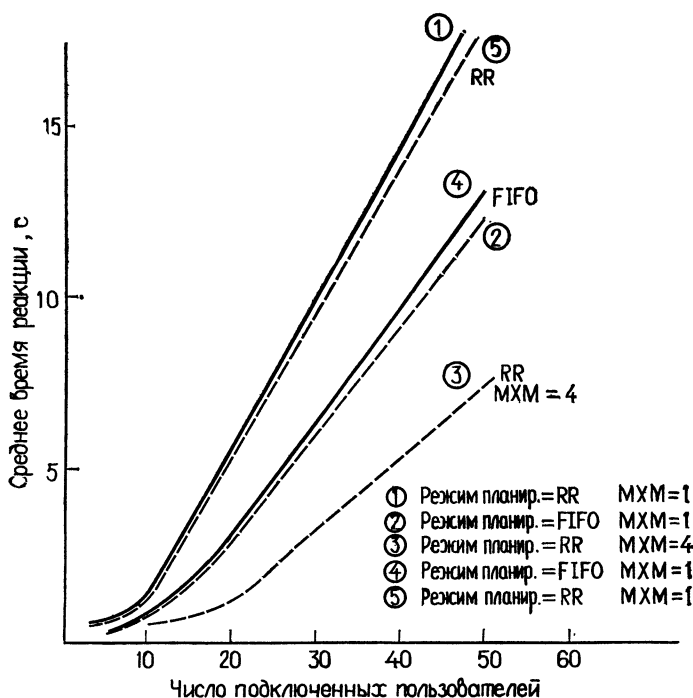


Рис. 10.15. Сравнение результатов моделирования с аналитическими результатами.

Условия: время свопинга = 0,12 с; среднее время выполнения = 0,1 с; квант времени = 0,1 с.

— аналитические результаты;  
— — результаты моделирования.

Располагая данными для управления имитатором, перейдем ко второму вопросу, указанному выше, т. е. оценке достоверности результатов работы имитатора. Один из способов оценки заключается в проверке того, что те же результаты могут быть получены и аналитико-математическими методами. Подобная проверка, конечно, является необходимой, но не достаточной для оценки логики построения имитатора, так как она обычно опирается на применение вырожденной модели, т. е. весьма упрощенного варианта модели, для которого существует аналитическое решение. На рис. 10.15 показаны результаты исследования имитатора этим способом. Были использованы два планировщика RR и FIFO<sup>1)</sup> с экспоненциальным распре-

<sup>1)</sup> RR (Roline Round) — дисциплина циклического обслуживания; FIFO (Firts-In-First-Out) — дисциплина обслуживания; первым пришел — первым обслужен. — *Прим. ред.*

делением значений времени поступления и выполнения. Видно, что аналитические данные и результаты моделирования весьма близки.

Другой более важный способ оценки достоверности имитатора заключается в сравнении результатов его работы с результатами измерений. Один из первых шагов, который должен быть здесь сделан, заключается в выборе сравниваемых переменных. Например, в качестве характеристики системы можно выбрать переменную, которая достаточно **нечувствительна** (или, как говорят статистики, **робастна**) по отношению ко многим параметрам системы, определяющим ее производительность. В этом случае сравнительно несложно добиться хорошего соответствия результатов измерений и моделирования. В то же время можно для сравнения подобрать переменные, значения которых весьма чувствительны ко многим параметрам системы. При этом трудно обеспечить высокий уровень соответствия экспериментальных и теоретических результатов из-за того, что незначительные различия между моделью и системой будут приводить к существенным расхождениям в значениях переменных, выбранных для сравнения. В последующем изложении будут использованы переменные, которые являются 1) наиболее важными показателями производительности (с учетом особенностей использования систем с разделением времени) и 2) измерение значений которых может быть выполнено системой APL. Эти переменные довольно чувствительны к изменениям параметров системы, в связи с чем можно ожидать значительных расхождений между результатами измерений и моделирования, и эти расхождения действительно наблюдаются. Возможно, сходство сравниваемых функций и описываемых ими тенденций более существенно, чем совпадение отдельных результатов. Как мы сейчас увидим, это сходство оказалось по результатам проведенных экспериментов вполне приемлемым. Измерения проводились на ЭВМ IBM/360 модели 50, работающей только с системой APL при 55 подключенных пользователях.

На рис. 10.16 показаны распределения экспериментальных и моделированных значений времени реакции. **Время реакции** определяется как время между появлением тракта и отведением ему первого кванта времени. На рис. 10.17 показано распределение отношения времени выполнения к затраченному времени, т. е. величины, обратной «коэффициенту удлинения». Так, например, точка на сплошной кривой (10 %, 0,7) означает, что для 70 % трактов, для которых были выполнены измерения, отношение времени выполнения к затраченному времени было меньше или равно 0,1. Другими словами, 70 % трактов имели коэффициент удлинения 10 или более.

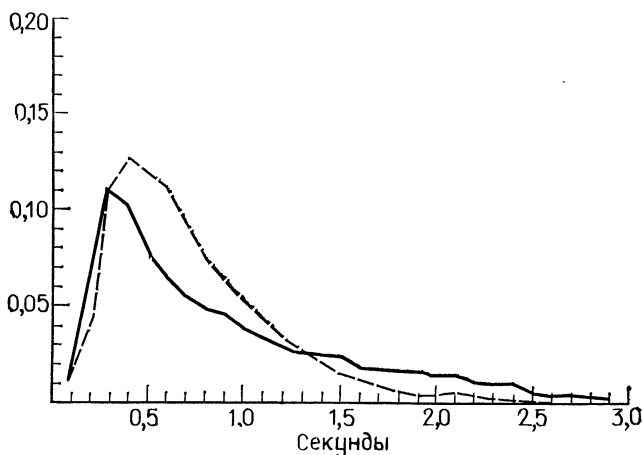


Рис. 10.16. Распределение относительной частоты времени реакции.

Условия: количество пользователей = 55;  $MXM = 4$ ; рабочая нагрузка: см. рис. 9.6.5 и 9.6.6.

— экспериментальное распределение;  
 --- моделированное распределение.

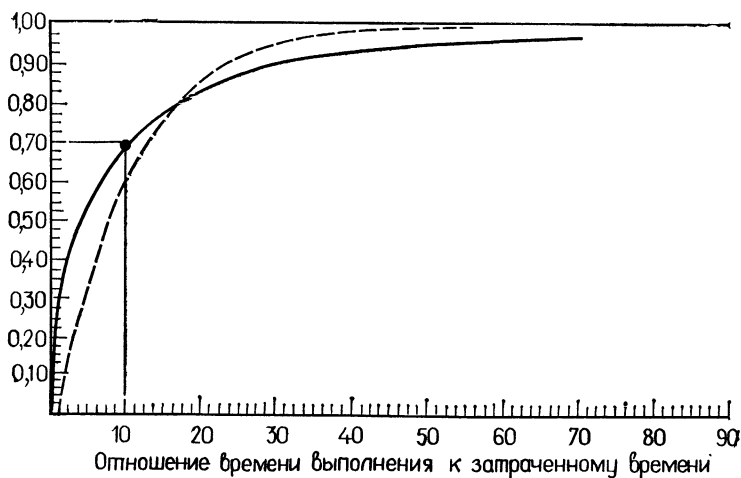


Рис. 10.17. Распределение относительной частоты отношения времени выполнения к затраченному времени.

Условия: количество пользователей = 55;  $MXM = 4$ ; рабочая нагрузка: см. рис. 9.6.5 и 9.6.6.

— экспериментальное распределение (среднее = 9,45);  
 --- моделированное распределение (среднее = 11,38).

Как рис. 10.16, так и рис. 10.17 отражают совместные характеристики производительности всех трактов. Как уже отмечалось, качество работы системы с разделением времени



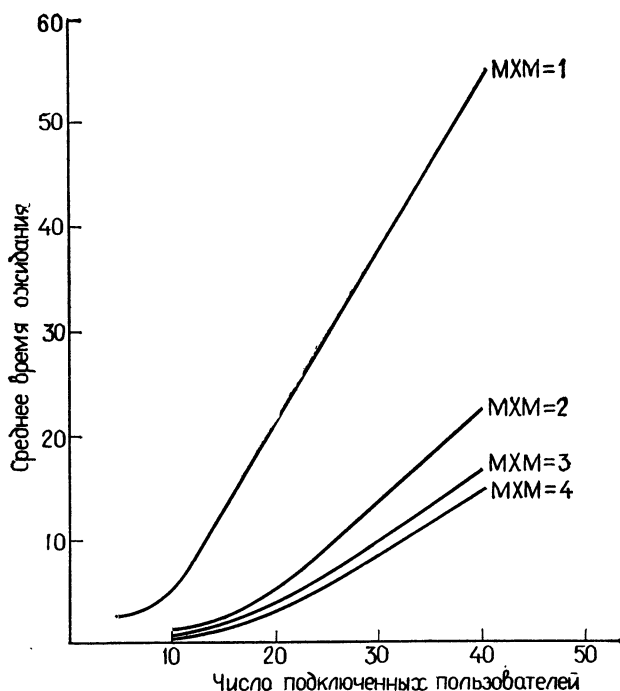


Рис. 10.18. Моделированные характеристики системы APL с различным числом рабочих областей

Распределения для системы APL. Время свопинга от 0,085 до 0,18 (равномерно), МХМ — число рабочих областей в ОП.

в наибольшей степени характеризуется ее способностью быстро обслуживать тривиальные и другие короткие тракты. Насколько удачно справляется с этим APL? Из рис. 10.14 видно, что около 80 % трактов относятся к категории тривиальных, время выполнения которых укладывается в один квант времени (в этой системе его длительность 0,1 с) или даже меньше. Из рис. 10.16 видно, что большинство запросов получают свои первые кванты времени (а для тривиального тракта только один квант) менее чем через 1 с. Этот факт явно свидетельствует о высокой чувствительности APL к тривиальным запросам. Можно показать, что эта чувствительность крайне медленно уменьшается с ростом числа подключенных пользователей в отличие от случая длинных нетривиальных трактов. Все эти наблюдения подтверждены опытом, накопленным пользователями APL.

На рис. 10.18 показаны результаты исследования имитатора с целью определения последствий одновременного исполь-

Число подключенных пользователей	30	30	50	50	30	30	50	50
Показатель быстродействия ЦП	4	4	4	4	1	1	1	1
МХМ (число рабочих областей в ОП)	2	3	2	3	2	3	2	3
Среднее время обдумывания	16,75	16,75	16,75	16,75	16,75	16,75	16,75	16,75
Среднее время выполнения	0,08	0,08	0,07	0,08	0,28	0,29	0,21	0,25
Среднее время ожидания	0,44	0,34	1,30	0,73	1,37	0,80	5,69	2,44
Средняя длина очереди	0,85	0,66	0,82	2,2	2,86	1,82	15,35	7,48
Средняя длина новой очереди	0,5	0,5	2,0	1,9	0,7	0,6	1,3	1,7
Показатель качества обслуживания	0,41	0,43	0,18	0,21	0,37	0,42	0,17	0,20
Пропускная способность	2,99	3,28	2,96	3,40	1,95	2,28	2,05	2,58
Использование времени ЦП, %	17	17	27	28	52	55	58	79
Использование времени каналов, %	50	46	84	77	63	54	88	89
Степень совмещения, %	9	9	17	20	33	34	45	69

Условия для всех работ, указанных в данной таблице

Число трактов/число включений=2000

Рабочая загрузка=распределениям APL (см. рис. 9.6.5 и рис. 9.6.6)

Квант времени=0,1 с

Продолжительность свопинга от 0,025 до 0,25 с

Дополнительное время, затрачиваемое ЦП на каждый свопинг = 0,01 с.

Взаимное влияние каналов=0,15

Рис. 10.19. Результаты моделирования.

зования нескольких рабочих областей в основной памяти (при наличии двух каналов свопинга). Из рисунка видно, что переход от одной резидентной рабочей области к двум при одном и том же числе подключенных пользователей значительно повышает производительность системы; однако дальнейшее увеличение числа рабочих областей (например, от трех к четырем) не влечет за собой существенных изменений производительности.

На рис. 10.19 приведена сводная таблица, которая отражает влияние варьирования числа подключенных пользователей, быстродействия ЦП и числа резидентных рабочих областей на среднее время ожидания, длину очереди, длину новой очереди (образующейся при поступлении новых запросов), показатель качества обслуживания коротких заданий, пропускную способ-

ность, степень использования ресурсов и совмещения различных действий. Эта таблица позволяет получить общее представление об основных параметрах системы.

### 10.8. РЕЖИМ РАЗДЕЛЕНИЯ ВРЕМЕНИ IBM

#### (TSO — Timesharing Option) ДЛЯ СИСТЕМЫ 360/370

Основным назначением этой системы, выпущенной в 1971 г., является предоставление пользователям широких возможностей операционной системы IBM ОС/360 общего назначения в режиме разделения времени. Кроме того, TSO дает абонентам, работающим в интерактивном взаимодействии с системой, возможность применять некоторые языковые средства, разработанные специально для режима разделения времени и не предусмотренные в системе пакетной обработки. Однако пользователь при желании может разрабатывать программы, полностью удовлетворяющие требованиям средств пакетной обработки, входящих в операционную систему. Работа в пакетном режиме может быть инициирована с терминала пользователя, и система способна выполнять обработку пакета одновременно с обслуживанием заданий переднего плана в режиме разделения времени<sup>1)</sup>.

Теперь кратко опишем два аспекта системы TSO. Первый касается средств, предоставляемых в распоряжение каждого отдельного пользователя, а второй охватывает вопросы управления ресурсами системы и планирования работы с разделением времени.

#### Средства отдельного пользователя

Каждый пользователь общается с TSO посредством **командного языка** этой системы, который выполняет те же функции, что и язык управления заданиями системы пакетной обработки, но имеет более широкие возможности, снабжен средствами, позволяющими в большей степени учитывать особенности деятельности пользователя в режиме разделения времени. Командный язык включает следующие средства.

1. Команды редактирования для удобного ввода, хранения, отображения и изменения информации, хранимой в виде так называемых массивов данных. Редактирование включает как контекстный поиск, так и ссылку по номеру строки.

---

<sup>1)</sup> Задания, поступающие от пользователя в результате интерактивного взаимодействия с системой, называются заданиями переднего плана. Задания, обрабатываемые в режиме пакетной обработки, называются фоновыми заданиями, или заданиями заднего плана. Заданиям переднего плана присваиваются более высокие приоритеты, чем фоновым заданиям. — *Прим. ред.*

2. Команды, которые именам массивов ставят в соответствие имена, используемые в программах (имена файлов), для последующей увязки названий хранящихся в памяти информационных единиц и используемых устройств с программными именами.

3. Выполнение программ.

Важной особенностью системы является возможность хранения последовательностей **команд** в виде **командных процедур**, т. е. подпрограмм на командном языке. Эти подпрограммы являются копиями каталогизированных процедур языка управления заданиями операционной системы. Каждой такой последовательности присваивается имя и она может снабжаться параметрами, почти так же, как при обычном связывании подпрограмм и использовании макровывозов. С практической точки зрения синтаксис командного языка TSO весьма напоминает язык макроассемблера ОС. Командные процедуры могут выполняться как с позиционными, так и ключевыми (самоопределяющимися) параметрами. В последнем случае можно задать «стандартные» значения параметров (значения по умолчанию). Рассмотрим, например, **заголовок** хранящейся командной процедуры, носящей, скажем, имя PLICLGO (в самой процедуре это имя не указано):

PROC 1, &X, SYS1 (\*), SYS2(\*).

В этой записи «1» означает, что имеется один позиционный параметр с именем X (первое имя после этого числа). SYS1 и SYS2 — имена ключевых параметров; их значения по умолчанию равны \*, что является именем терминала как системного устройства. Таким образом, пользователь может вызвать эту процедуру (названную PLICLGO) с помощью операторов вида

PLICLGO STAT SYS1 (DATA)

или

PLICLGO STAT.

При использовании первого варианта оператора вызова данной процедуры будет произведена замена именем STAT параметра X во всех его вхождениях в описание процедуры, а имя DATA будет подставлено везде, где появилось в описании ключевое слово SYS1. Отметим, что, поскольку имя SYS2 при вызове процедуры не использовалось, при каждом появлении этого имени в процедуре будет использовано его значение по умолчанию (служащее для обозначения терминала). Таким образом, применение указанного первого варианта оператора определяет вызов командной процедуры, носящей имя PLICLGO и имеющей в заголовке PROC, что приводит, например, к компиляции и последующему выполнению программы, хранящейся в массиве с именем STAT. Эта программа полу-

чает входные данные из массива с именем DATA и выводит результаты на терминал. Второй вариант оператора вызова похож на первый, но из-за того, что в данном случае опущен также и параметр SYS1 как для ввода, так и для вывода данных, по умолчанию предполагается использовать терминальное устройство.

Идея применения хранимых командных процедур очень важна. Она не только исключает необходимость утомительного повторения ввода часто используемых последовательностей, но и дает возможность проблемно-ориентированным пользователям, которые мало интересуются внутренним механизмом работы системы и мало знают о нем, создавать и использовать языковые подсистемы. Программы подобных подсистем могут быть разработаны фирмой IBM или, что делается чаще, системными программистами по месту эксплуатации систем. Даже обычные пользователи могут при желании реализовать командные процедуры собственной разработки.

Опыт применения языка TSO, накопленный к моменту написания данной работы (1975 г.), позволил выявить его большие потенциальные возможности, которые пока в определенной степени сводятся на нет недостатками его архитектуры. TSO предусматривает использование по выбору трех языков, предназначенных для интерактивного взаимодействия пользователя с системой: ITF/BASIC, ITF/ПЛ/1 и диалоговый Фортран. Имеются средства строчного синтаксического контроля при вводе и ряд других отладочных средств.

Возможно также применение стандартных компиляторов ОС/360 для Фортрана, Кобола, ПЛ/1, Алгола и Ассемблера. Для работы с терминалами могут быть использованы методы последовательного доступа (BSAM и QSAM)<sup>1)</sup>, также являющиеся принадлежностью ОС. Для работы с разделением времени предлагается ряд процессоров в несколько модифицированных вариантах. Команда TEST обеспечивает возможность работы в старт-стопном режиме и трассировку выполнения программы. К сожалению, эта команда не позволяет вести отладку программы на исходном языке программирования. Поскольку выводимые данные представлены на машинном языке, пользователь должен в последующем позаботиться о преобразовании выходных данных команды в символьный формат с помощью таблицы символов.

### Управление ресурсами в TSO

Чтобы получить начальное представление о том, каким именно образом производится в TSO управление ресурсами,

<sup>1)</sup> BSAM — базисный последовательный метод доступа, QSAM — последовательный метод доступа с очередями. — *Прим. ред.*

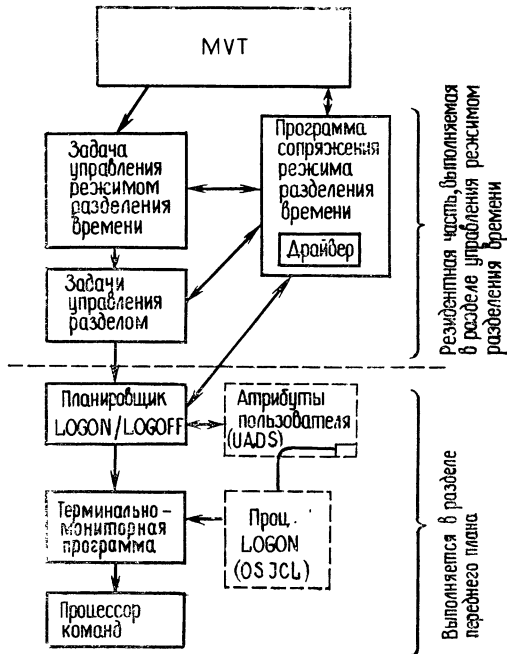


Рис. 10.20 Логическая иерархия управления в TSO.

обратимся к рис. 10.20, на котором показаны основные объекты управления, расположенные в соответствии с логической иерархией системы. Центральный планировщик ресурсов называется **драйвером**. Он взаимодействует со всеми программами, реализующими другие основные функции, через свою программу **связи** в режиме **разделения времени**. Драйвер получает сообщения о состоянии как задачи, так и ресурсов и с учетом некоторых параметров, зависящих от конкретных условий эксплуатации системы, принимает решения о том, какое задание следует загрузить в память, а какое выгрузить из нее и какое время работы ЦП должно быть предоставлено каждому заданию (подробнее эти аспекты будут рассмотрены ниже). Драйвер является также основным средством управления пакетной (фоновой) частью рабочей загрузки.

Последовательность управления системой с основных чертах представлена на рис. 10.20. Рассмотрим состояние системы сразу же после запуска MVT, но до запуска TSO. До начала работы в режиме разделения времени запускается MVT — мультипрограммный режим с переменным числом задач. Запуск TSO осуществляется с помощью команды, которая инициирует

```

//NFSPROC  EXEC  PGM=IKJEFTO1,ROLL=(NO,NO)
//SYSUDUMP  DD    SYSOUT=R
//SYSHELP   DD    DSN=SYS1.HELP,DISP=SHR
//SYSEDIT   DD    DSN=&EDIT,UNIT=SYSDA,SPACE=(1688,(50,20))
//SYSPROC   DD    DSN=SYS1.CMDPROC,DISP=SHR
//DD1       DD    DYNAM
//DD2       DD    DYNAM
//DD3       DD    DYNAM
//DD4       DD    DYNAM
//DD5       DD    DYNAM
//DD6       DD    DYNAM
//DD7       DD    DYNAM
//DD8       DD    DYNAM
//DD9       DD    DYNAM
//DD10      DD    DYNAM

```

Рис 10.21. Типичная процедура LOGON.

системную программу, называемую задачей управления режимом разделения времени (TSC — Time Sharing Control Task). Эта задача в последующем обеспечит связь оператора с системой для ввода параметров и может инициировать выполнение задач управления разделом (RCT — Region Control Task) для каждого раздела переднего плана. В ответ на команду LOGON, вводимую с терминала, каждая задача RCT может в свою очередь вызвать копию планировщика LOGON. Планировщик LOGON принимает информацию, содержащуюся в сообщениях пользователя, представленных в формате LOGON, а также имеет доступ к массиву данных, называемому **массивом атрибутов пользователя** (UADS — User Attribute Data Set), который содержит предварительно записанную информацию о каждом зарегистрированном пользователе, допущенном к работе с системой (учетный номер, пароли LOGON и т. д.). В UADS включено имя каталогизированной процедуры языка управления заданиями операционной системы, которая называется процедурой LOGON. Процедура LOGON определяет выполнение одного шага задания ОС, и поэтому каждый период работы TSO от вызова LOGON до вызова LOGOFF на каждом отдельном терминале рассматривается как один шаг задания ОС. Оператор EXEC<sup>1)</sup> указывает программу, которая должна быть выполнена. В случае процедуры LOGON (рис. 10.21) вызванная с ее помощью программа предназначена для интерпретации всех команд, которые пользователь вводит с помощью терминала во время сеанса. Хотя эта программа может

<sup>1)</sup> EXEC — управляющий оператор языка управления заданиями ОС, предназначенный для спецификации программы или каталогизированной процедуры. — *Прим. ред.*

Объект	Сокращенное обозначение	Тип	Число	Активная область пребывания	Основные функции
Драйвер	—	Системная программа	Одна на систему	Раздел управления временем	Главный распределитель ресурсов для TSO и пакетного режима
Задача управления временем	TSC	Системная программа	Одна на систему	Раздел управления временем	1. Запуск RCT 2. Связь с оператором системы TSO 3. Управление свопингом
Задача управления временем	RCT	Системная программа	Одна на раздел переднего плана	Раздел управления временем	1. Управление замораживанием 2. Запуск процедуры LOGON/LOGOFF
Программа управления терминалом	TMP	Системная или пользовательская программа	Одна на каждого подключенного пользователя	Раздел переднего плана	Интерпретация команд, поступающих непосредственно с терминала, вызов процессора команд
Массив атрибутов пользователя	UADS	Разделение массива	Одна на каждого пользователя, имеющего разрешение	Диск	1. Наименование процедур LOGON 2. Учетный номер, пароли LOGON 3. Установление категории пользователей
Процедура LOGON	—	Каталогизируемая процедура JCL	Одна или несколько на каждого пользователя	Диск	Запуск TMP, определение массивов данных, необходимых на время сеанса на терминале (включая процедуры)
Командная процедура	—	Разделение массива	Любое	Диск	Хранимая командная подпрограмма, которая: 1. Вызывает системные или пользовательские программы для выполнения 2. Резервирует пространство 3. Сопоставляет массивы (имена объектов физического пространства) с файлами (именами программ)

Рис. 10.22. Важные системные объекты TSO.



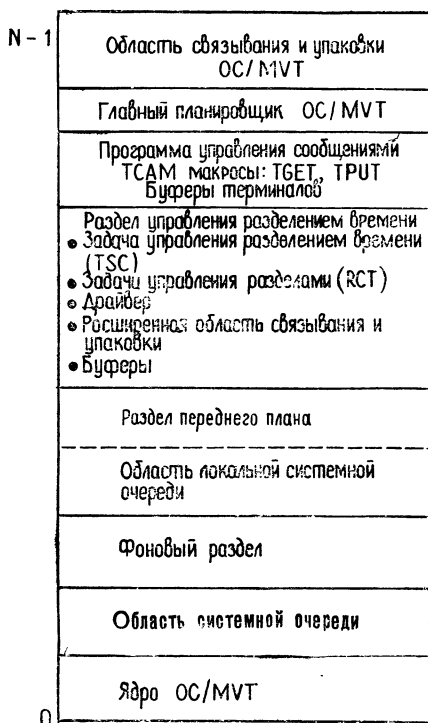


Рис. 10.23. Распределение ядра ОП для типичной системы TSO.

быть написана пользователем с учетом специфики его задач, мы будем предполагать, что (как это обычно и бывает) программа, вызываемая в данном случае, это терминально-мониторная программа (TMP — Terminal Monitor Program), поставляемая фирмой IBM. TMP может рассматриваться как «локальный супервизор», поскольку она интерпретирует команды пользователя (он может вызвать интерпретатор команд для синтаксического анализа) и инициирует процесс вызова всех программ, необходимых для реализации последовательности команд. Помимо оператора EXEC, эта процедура, входящая в каталог LOGON, определяет несколько операторов DD<sup>1)</sup>, которые будут необходимы пользователю во время его сеанса. Эти операторы предназначены для резервирования пространства на диске и установления соответ-

ствия между массивами и именами. Каждый пользователь может иметь в своем каталоге несколько процедур LOGON. Их имена хранятся в массиве UADS. Процедуру, которая необходима пользователю, он определяет в своем сообщении LOGON (или она предоставляется ему по принципу умолчания). Все, что связано с LOGON, можно представить себе как набор средств, дающих пользователю возможность описать и запустить «собственную» машину со своей логической структурой.

На рис. 10.22 приведены краткие сведения о каждом из логических объектов системы TSO. На рис. 10.23 показана схема распределения основной памяти в типичной системе TSO с одним разделом переднего плана и одним фоновым разделом. Заметим, что большинство областей памяти, показанных на рис. 10.23,— это области, необходимые для работы операцион-

<sup>1)</sup> DD — управляющий оператор определения данных языка управления заданиями ОС. — *Прим. ред.*

ной системы MVT. Кроме них в памяти имеется раздел, предназначенный для управления разделением времени, в котором постоянно находится большинство резидентных программ системы TSO и, разумеется, раздел переднего плана. Отметим, что раздел переднего плана подразделяется на две части, одна из которых загружается, сверху вниз, т. е. начиная с младших адресов, а другая — снизу вверх, т. е. со старших адресов. Вторая часть называется **областью локальной системной очереди** и зарезервирована для хранения управляющей информации, связанной с текущей работой, выполняемой в соответствующем разделе. Эта информация выводится из основной памяти (а впоследствии снова загружается в нее) вместе с другой существенной информацией раздела. (Пересылкам подлечит только та часть раздела, которая содержит необходимую информацию.)

Пользуясь рис. 10.23, можно рассмотреть и обсудить один из основных аспектов производительности системы. В **расширенной области связывания и упаковки** в разделе управления работой с разделением времени находятся копии системных программ, наиболее часто применяемых пользователями TSO. Как и при работе любой традиционной операционной системы (не использующей виртуальной памяти), в каждой установке должен быть указан необходимый для этих целей объем памяти, а также указаны, какие именно программы и данные должны быть сделаны в ней резидентными. Так как любая нерезидентная подпрограмма при каждом использовании должна быть считана с диска или барабана, отбор наиболее часто используемых модулей имеет весьма существенное значение для повышения производительности системы. Фирма-изготовитель снабжает пользователей некоторыми рекомендациями, призванными облегчить начальный запуск системы. Однако, чтобы обеспечить высокую производительность, необходимо применять программный монитор для определения набора модулей, которые наиболее целесообразно использовать в качестве резидентных. Можно утверждать, что такой подход является одной из основных черт систем типа ОС, обладающих высокой модульностью и характеризующихся сложностью структур данных и программ. Если выбор резидентных модулей осуществлен недостаточно разумно, процессы ввода-вывода в управлении данными ОС, на которые приходится значительная часть работы TSO, могут в большей степени, чем свопинг, определять остроту конфликтов, возникающих из-за обладания ресурсами.

Обратимся теперь к другой важной проблеме производительности: планированию размещения заданий, поступающих от терминалов, в основной памяти, которое называется основным

квантованием времени или обслуживанием, а также в распределении времени работы центрального процессора между заданиями, уже находящимися в основной памяти (в том числе фоновыми заданиями). Управляющие программы TSO (драйвер и TSC) планируют перемещение информации между разделом (разделами) переднего плана и внешними устройствами, участвующими в свопинге. Фактически пересылке подлежит только та часть раздела, называемая **свопинговой загрузкой** раздела, которая действительно будет использована. Выгружающий свопинг начинается только после того, как завершатся все нетерминальные действия по вводу-выводу, в которых участвует данный раздел. Этот прием называется **замораживанием**. Для каждого терминала в основной памяти предусмотрен резидентный буфер, который находится там постоянно (как это сделано в APL). Поэтому передача терминальных сообщений может происходить даже в тот период, когда раздел пользователя подвергается выгружающему свопингу. Таким образом, нет необходимости замораживать операции ввода-вывода с участием терминала.

Планирование использования важнейших ресурсов, т. е. ЦП и разделов основной памяти, базируется на использовании двух временных интервалов планирования. Первый — **основной квант времени задания** — определяется как время, отведенное заданию для нахождения в основной памяти, т. е. интервал между загрузкой его в ОП и выгрузкой из ОП. Второй — **вторичный квант времени задания** — представляет собой максимальный непрерывный интервал времени работы ЦП, выделенный для обслуживания резидентного в настоящее время задания. Кванты времени обоих типов зависят от ряда параметров, часть которых определяется системой (системными программами), а другие измеряются самой системой. Совокупность конкретных значений параметров называется **логической очередью**. Совокупность заданий пользователя с общей логической очередью составляет очередь заданий, подлежащих обслуживанию. В TSO понятие «обслуживание задания» означает лишь загрузку его в раздел основной памяти. Поэтому логическая очередь является совокупностью параметров планирования, которые определяют основной квант времени для каждого задания, входящего в очередь. Отметим, что к одному разделу основной памяти может быть несколько очередей. Задания, ожидающие обслуживания в каждой очереди, обслуживаются в циклическом порядке. Однако TSO может организовать несколько очередей с таким расчетом, что задания, ожидающие обслуживания, относятся к различным классам. Рассмотрим несколько подробнее параметры планирования, которые определяют логическую очередь, а также то, каким образом зада-

ние ставится в очередь и как оно может быть перемещено из одной очереди в другую. Предварительно отметим, что при отведении вторичных квантов (т. е. квантов времени работы ЦП) тем заданиям в основной памяти, которые готовы к выполнению, также учитываются определенные параметры планирования.

Как уже упоминалось, каждый раздел основной памяти может иметь несколько очередей заданий. Управление каждой очередью осуществляется путем ссылок на фиксированные значения, которые присваиваются при определении системы следующим параметрам и могут быть изменены лишь вручную:

$A_{R,Q}$  — среднее время обслуживания заданий в очереди,

$M_{R,Q}$  — минимальный основной квант времени,

$SL_{R,Q}$  — предельный объем загружаемой при свопинге части раздела,

$IL_{R,Q}$  — предельное время интерактивного взаимодействия,

$SC_{R,Q}$  — число циклов обслуживания данной очереди перед переходом к следующей очереди.

Время от времени система определяет дополнительный параметр  $N_{R,Q}$ . Он означает число заданий в очереди, готовых к выполнению.  $A_{R,Q}$  — интервал времени между двумя следующими друг за другом загрузочными свопингами любого задания в очереди. Значение  $M_{R,Q}$  должно служить гарантией того, что задания, стоящие в очереди, остаются в основной памяти до тех пор, пока над ними не будет выполнена некоторая продуктивная работа.  $SL_{R,Q}$  служит мерой максимального объема основной памяти, который может быть затребован заданием из данной очереди.  $IL_{R,Q}$  — максимальное разрешенное общее время нахождения в основной памяти с момента последнего запроса пользователя на обслуживание. Любой конкретный набор этих четырех значений описывает характер планирования работ по выполнению заданий данного класса. Исходя из перечисленных параметров очереди заданий, планировщик рассчитывает величину основного кванта времени для каждого задания очереди по следующей формуле:

$$T_{R,Q} = \text{Max} \left( M_{R,Q} \frac{A_{R,Q}}{N_{R,Q}} \right).$$

Каждое задание удерживается в основной памяти в течение времени  $T_{R,Q}$ ; все задания очереди обслуживаются при этом в циклическом порядке. После выполнения  $SC_{R,Q}$ -циклов обслуживания данной очереди начинается обслуживание следующей очереди.

Если значения параметров задания превышают предельные значения  $SL_R$  и  $IL_{RQ}$ , то система отыскивает для его обслуживания новую очередь, в которой эти значения являются допу-

стимыми. Новая очередь должна относиться к тому же разделу, так как отсутствие динамического перераспределения памяти в Системе 360 не позволяет перемещать программу в новый раздел после того, как началось ее выполнение. В общем случае новая очередь будет характеризоваться другими значениями параметров. Например, рассмотрим следующие две очереди:

	Q1	Q2
A	1	10
M	0,2	1
SL	100 Кбайт	100 Кбайт
IL	0,1	5

В данном случае очередь Q1, с присущими ей небольшими значениями *IL* и *A*, явно предназначена для запросов высокой интерактивности с использованием быстрого свопинга, а очередь Q2 — для запросов, длительность обслуживания которых, как это уже выяснено, достаточно велика. Отметим, что большее значение *A* определяет относительно более длительное пребывание заданий очереди Q2 в основной памяти, в результате чего они имеют большую возможность быть объектами продуктивной работы ЦП в интервалах между загружающими свопингами.

На интервал времени работы ЦП, называемый основным квантом, из заданий, находящихся в разделах TSO основной памяти в данный момент времени и готовых к выполнению (т. е. не ожидающих срабатывания средств ввода-вывода), выбирается (назначается) путем диспетчеризации одно. TSO позволяет персоналу системного уровня отводить определенную гарантированную часть общего времени работы ЦП для пакетной обработки. Остальное время ЦП называется временем заданий переднего плана. Это время подразделяется на вторичные кванты времени.

Время работы ЦП, отведенное для выполнения заданий переднего плана, может распределяться в виде вторичных квантов времени в соответствии с одной из следующих трех схем, выбираемой системными программистами.

1. *Простая диспетчеризация.* Длительность вторичного кванта устанавливается равной всему времени обслуживания заданий переднего плана; это время предоставляется заданию наивысшего приоритета, т. е. заданию, которое загружено в основную память раньше всех. Простая диспетчеризация используется в тех случаях, когда в системе имеется только один раздел переднего плана.

2. *Равномерная диспетчеризация.* Вторичный квант определяется как частное от деления времени обслуживания заданий переднего плана на число таких заданий, находящихся в основной памяти и готовых к выполнению.

3. *Диспетчеризация со взвешиванием.* Система количественно оценивает долю времени, затрачиваемую каждым заданием, на ожидание, основываясь на анализе времени ожидания им ввода-вывода (это не относится к вводу-выводу от терминала, при котором не требуется загрузочного свопинга). Чем больше полученное время ожидания ввода-вывода, тем больше рассчитанная длительность вторичного кванта. Тем самым компенсируются потери части вторичного кванта, которые задания, критичные с точки зрения ввода-вывода, несут из-за ожидания ввода-вывода по сравнению с заданиями, критичными лишь с точки зрения работы ЦП.

## 10.9. СЕТЬ ИНФОРМАЦИОННОГО ОБСЛУЖИВАНИЯ GE

До сих пор речь шла об однопроцессорной системе с разделением времени, к которой пользователи могут быть подключены посредством телефонных линий. Такие системы обычно применяются для обслуживания потребностей отдельной организации, где одно и то же оборудование часто используется как для работы с разделением времени, так и в режиме пакетной обработки. Несколько по-иному строится работа систем, когда фирма предоставляет коллективное обслуживание с разделением времени на абонентской основе всем, кто имеет доступ к терминалам, подключенным к системе.

Сеть информационного обслуживания фирмы General Electric — это служба абонентского обслуживания, представляющая собой комплекс аппаратных и программных средств, способный обслуживать множество удаленных пользователей. Группы этих пользователей могут принадлежать разным или одной организации и пользоваться или не пользоваться общими информационными ресурсами, такими, например, как файлы пользователей. Более того, системные ресурсы, как и пользователи, могут быть географически разнесены на значительные расстояния и изменяться со временем с минимальным ущербом для абонентов. Такую систему иногда называют системой информационных коммунальных услуг, так как она во многом напоминает другие коммунальные службы, например обеспечивающие снабжение электроэнергией, телефонную связь и т. д. Каждый пользователь, считающийся абонентом системы, оплачивает только те средства, которые он использует (включая затрачиваемую на его обслуживание долю времени работы центральных устройств), но он освобожден при этом от забот

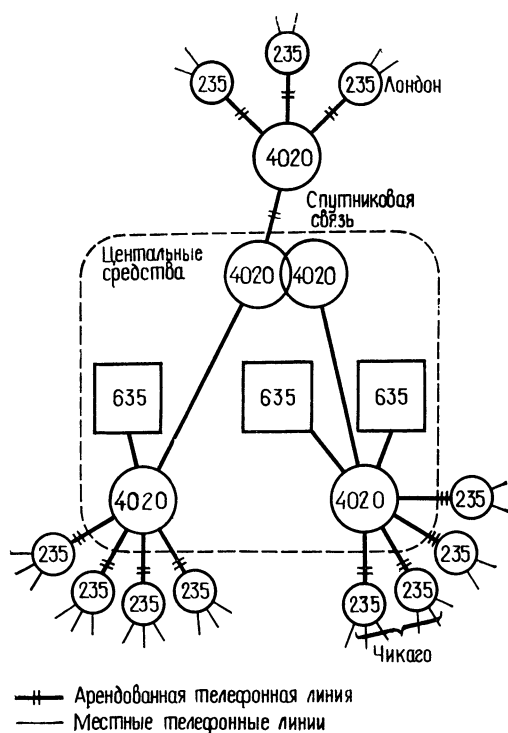


Рис. 10.24. Фрагмент сети информационного обслуживания GE (© 1972).

Обозначения.

235 — ЭВМ GE-235, используемая в качестве концентратора линий  
 4020 — ЭВМ GE-4020, используемая в качестве деконцентратора линий или переключателя сообщений  
 635 — ЭВМ GE-635

и не несет ответственности за капиталовложения, развитие ресурсов систем и управление ими. Рентабельность работы организации, предоставляющей эти возможности, обеспечивается за счет оплаты пользователями информационных услуг.

На рис. 10.24 показан фрагмент сети, поясняющий некоторые принципы построения конфигурации связей. Если рассматривать схему со стороны пользователей, то их терминалы (на рисунке не показаны) подключаются к малой ЭВМ GE-235, расположенной в территориальной близости, набором соответствующего номера телефона. После соединения весь обмен информацией с аппаратной точки зрения происходит непосредственно с GE-235, которая может обслуживать несколько терминалов, расположенных в определенном регионе (например, в Чикаго, шт. Иллинойс). Локальная ЭВМ выполняет роль концентратора сообщений. Она собирает (буферизирует) вхо-

дящие и выходящие сообщения с нескольких региональных линий и постепенно передает их по единственной арендованной междугородной телефонной линии, к месту размещения центральной ЭВМ, например в Кливленде, шт. Огайо. В этом регионе другая ЭВМ разделяет сообщения, обеспечивает контроль ошибок (а если они обнаружены, то запрашивает повторную передачу сообщения) и далее передает сообщения в GE635 — центральную ЭВМ системы. Этот процесс, описанный в терминах поступления сообщений от терминалов в ЭВМ, происходит аналогичным образом и в обратном направлении. Во введении к этой главе было отмечено, что применение концентратора линий может в значительной степени уменьшить затраты на телефонную связь за счет рационального совместного использования единственной арендованной линии. Каждый пользователь непосредственно оплачивает только локальные телефонные расходы (его доля оплаты арендованной линии включается в абонентную плату).

Отметим еще некоторые особенности сети, которые можно увидеть на рис. 10.24. Абоненты одного города могут подключаться к любой из нескольких ЭВМ-концентраторов, каждая из которых имеет прямой доступ к определенным центральным ресурсам; так, например, на схеме показаны два концентратора, размещенные в Чикаго. Каждая центральная ЭВМ-концентратор (типа 4020) соединена с другими таким образом, что обрабатывающие средства каждой ЭВМ могут быть связаны с аналогичными средствами любой другой ЭВМ, и, таким образом, каждый пользовательский терминал может в принципе установить связь с любой из нескольких центральных ЭВМ.

Одним из важных потенциальных достоинств сети является ее способность объединять ресурсы таким образом, что при изменении реальных требований они могут быть распределены различными способами. Это может быть сделано несколькими разными путями; мы остановимся лишь на некоторых из них, применяемых в сети GE. К моменту написания данной работы (1972 г.) номер каждого абонентского счета был поставлен в соответствие с файловым хранилищем по месту размещения конкретной ЭВМ GE235. Это соответствие автоматически устанавливается системой таким образом, что средства обслуживания абонента с данным номером не перемещаются автоматически между системами. Однако при выполнении назначений предпринимаются попытки балансировать нагрузку системы путем назначения абонентов на те ЭВМ 635, загрузка которых относительно невелика. Еще одним полезным фактором балансирования загрузки является учет разницы часовых поясов между различными географическими пунктами (например, того обстоятельства, что в Нью-Йорке время на три часа отстает



от времени Лос-Анджелеса), в результате которой рабочие дни, а значит, и часы наибольшей активности смещены в разных географических точках относительно друг друга.

До сих пор рассматривались только коммуникационные аспекты работы сети GE. Теперь коротко остановимся на средствах, предоставляемых отдельным пользователям, акцентируя внимание на тех из них, которые не встречались в описанных ранее системах APL и TSO.

Система в настоящее время предоставляет пользователю возможность работы на трех языках программирования: Бэйсик, Фортран и Ремапт (ранее предусматривалась также возможность работы на Алголе). Бэйсик — это язык, аналогичный Фортрану; первоначально он был разработан в дармутском колледже, впоследствии существенно пополнен операторами действий над файлами и обработки символьных строк, а также другими средствами. Как Бэйсик, так и Фортран используют компилирующие трансляторы. Язык Ремапт удобен для описания деталей, обрабатываемых станками, представленные на нем описания преобразуются в программы для числового управления станками.

Центральная библиотека программ доступна всем пользователям и включает программы по статистике, программы для научных расчетов, деловые игры и т. д. Создание, вывод и внесение изменений в программы и данные существенно облегчаются благодаря наличию удобного редактора текста. На дисках могут храниться обширные файлы (мерой оплаты за них являются килобайто-месяцы). Файловая система для хранения программ и данных характеризуется наличием эффективных средств их защиты и обеспечения безопасности. Например, когда пользователь уничтожает файл, его содержимое стирается, чтобы предотвратить случайное чтение ранее находившейся там информации новым пользователем, которому отведено освободившееся пространство. Разработать систему защиты файлов достаточно просто, если не требуется использовать их с разделением времени. Система GE обеспечивает и совместное использование, и защиту файлов. Во-первых, в некоторой степени защита обеспечивается традиционной схемой паролей. Пользователь может присвоить файлу пароль; при доступе к файлу необходимо использование и пароля и имени файла. Во-вторых, пользователь может разрешить (или запретить) отдельным пользователям доступ к его файлам. С помощью команды PERMIT задаются четыре типа доступа:

**ВЫПОЛНИТЬ**

**СЧИТАТЬ** (подразумевает ВЫПОЛНИТЬ)

**ЗАПИСАТЬ** (подразумевает ПРИСОЕДИНИТЬ)

**ПРИСОЕДИНИТЬ**

Содержание команды PERMIT может быть изменено только данным пользователем.

Описание сети GE не опубликовано. По имеющимся данным в настоящее время она включает двенадцать ЭВМ GE 635. Каждая из этих ЭВМ имеет время цикла памяти 1 мкс (длина слова 36 бит), время выполнения команд типа СЛОЖИТЬ/ВЫЧЕСТЬ/СПРАВНИТЬ с фиксированной точкой составляет 1,9 мкс. ЭВМ GE635 располагает системой прерывания, привилегированным режимом и одним адресным регистром для пересылок и защиты информации. К числу устройств, которые могут быть к ней подключены, относятся магнитные барабаны емкостью 4.7 Мбайт каждый, при среднем времени доступа 18 мкс и скорости передачи 370 Кбайт/с. Могут также использоваться диски с фиксированными головками и сменные диски. Некоторые из этих устройств безусловно входят в состав системы.

Сведений о конфигурации системы и ее производительности опубликовано не было. По неофициальным данным сеть обслуживает на сегодня около 1000 пользователей одновременно за вполне приемлемое время.

# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- Абак 14  
Абсолютный двоичный загрузчик 296  
Адрес 38  
— дорожки 339  
— магнитного барабана 260, 261  
— диска 339  
— поверхности 339  
— цилиндра 339  
Адресная константа 298—300  
— линия 212, 214, 215  
— шина 46  
Алгол, язык 17  
АЛУ 22, 40, 138, 139  
— двоично-десятичный сумматор 157—160  
— двоичное умножение 171—175, 177—183  
— двоичный сумматор 142, 143, 144—157  
— деление 169, 176—183  
— десятичная система счисления с обратным кодом 161—166  
— десятичное умножение 175, 176  
— конструкция 139, 140  
— логические операции 183—187  
— операции 45, 168—170  
— параллельный арифметический блок 151—156  
— — двоичный сумматор 144—146  
— представление положительных и отрицательных чисел 146—151  
— — целых чисел 140—142  
— (полный) сумматор 143, 144, 154—157  
— четырехразрядное 185, 186  
Аналитическая машина (Бэббиджа) 15  
Аналоговые ЭВМ 18—19  
Аппаратура (оборудование) 20, 21  
— устройство ввода 36  
— — ввода-вывода 320—329  
— — вывода 36  
— — долговременной памяти 329—340  
— — памяти 37—40  
Арифметика, АЛУ 138  
— двоичная система счисления 52, 53  
— с плавающей точкой 192 193  
— — фиксированной точкой 187  
Асинхронная триггерная схема 98  
Ассемблер 294  
— язык 293, 294  
Ассоциативный закон 73, 74  
  
База (основание) системы счисления 49  
Биполярная память 218—223  
Биполярные ПЗУ 237  
БИС кристалла в АЛУ 138  
— — ПЗУ 235  
Бит (разряд) 36, 52  
— знака 140  
— — в двоично-десятичной системе 160  
— — системе с плавающей точкой 190, 191  
Блок плат сердечников в памяти на сердечниках 245—248  
Бод 322  
Булева алгебра 65  
— описание логической схемы 87, 88  
— переменные 66, 72—74, 79  
— теоремы 72—78  
Булевы выражения 68—70  
— — логические вентили 89—95  
— — упрощение с помощью карт Карно 79—87  
— переменные 66  
Бумажная лента (перфоленда) 326, 328, 329  
— — перфоратор 41  
— — считывающее устройство 36, 37

Буферная память 341  
Буферные регистры 48  
— — памяти (БРП) 44, 198, 251—253  
Буферный усилитель 90

**Ввод 32**

— с перфокарт 37  
— — перфолент 37  
Ввод-вывод 24, 31, 41  
— каналы 46  
— устройства 24, 33, 36, 40, 320—329  
— шина 46  
Вектор 27  
Вентили 89—95  
Ветвление команд 22, 35  
Взвешенные коды 54, 55  
Внешние символы 299—303  
— — словарь (СВС) 300—303  
Восьмеричная система чисел 293  
Временная последовательность памяти на сердечниках 248, 249  
Временное разделение адресных линий 229  
— — система 17  
Временной цикл памяти 38  
Время доступа 195  
— жизни 16  
— ожидание (задержка вращения) 263, 341  
— поиска 263  
— реакции 382  
Вывод 33  
— устройство 33  
Выделение (логическое умножение) 183  
Выравнивание порядков 187  
Выходы триггерной схемы 96, 97  
Вычитание, двоично-десятичная система с обратным кодом 161—166  
— в параллельном арифметическом блоке 151—154  
— — системе с плавающей точкой 192, 193  
— двоичное 53

Генератор сигналов истинного времени 346

Гибкие (флоппи) диски 269—272, 339

Головки гибких магнитных дисков 270, 271

— магнитных барабанов 256—259, 342  
— — дисков 262—268, 338, 339  
— — лент 276, 331  
Голосовой ввод и вывод 329  
Графический дисплей на ЭЛТ 322  
Группа (файл) регистров 135

Данные 21, 35

Двоичная арифметика 52, 53

— — система счисления 21, 49, 50  
— — умножение и деление 171—175, 177—183

Двоично-десятичный код (ДД) 55

— сумматор 157—160

Двоично-кодированные таблицы состояний 106—108

Двоичные карты 327

— коды 21, 36, 54—57

— — обнаружения и исправления ошибок 58—64

— счетчики 129, 130

— — проектирование 122—124

Двоичный полусумматор 142—144

Двойное отрицание, закон 73

Двойственности принцип 73

Двумерная адресация кристаллов памяти 206—212

Двумерная выборка 242—245

— матрица магнитных сердечников 242—245

Деление в АЛУ 169, 176—183

— без восстановления остатка 178, 181—183

— двоичное 53

— с плавающей точкой 193

Де Моргана закон 72, 73, 76, 77

Десятичная система счисления 49

Десятичный счетчик 118—122

Дешифраторы 199, 200, 203—206

— выбор адреса на шинах 216

— двумерный системы адресации памяти 207—210

— магнитных барабанов 261

— многоступенчатый 205, 206

— пирамидальный 205

— ПЗУ 233

— типа «1 из 2<sup>n</sup>» (декодирующая матрица) 203

Диаграмма состояний 101—108

Дизъюнкция 67, 72

Динамическая память 196

Динамическое ЗУ на МОП-транзисторах 218, 228—231

Диоды

— дешифратор 204—206

- Дноды ПЗУ 233, 234  
 Дискетки (гибкие) (флоппи) 269—272  
 Диски с плавающими головками 264, 265, 267  
 — — перемещающимися головками 263  
 — — фиксированными головками 263, 264  
 Дисковая память 261—269, 336—341  
 — — используемая в спулинге 308  
 — — на гибких дисках 269—272  
 Дистанционный ввод задания (ДВЗ) 17  
 Дистрибутивный закон 72  
 Длина слова памяти 38  
 Долговременная память 329—340  
 Дополнение (цифры) 50  
 Дополнительное правило перемещения 376  
 Дополнительный код 149—151  
 Дорожки 39  
 — на гибких дисках 270—272  
 — — магнитных барабанах 256—259, 342  
 — — — дисках 262—269, 337—339  
 — — — лентах 277—279, 331  
 Драйвер запрета 247—249  
 Дуплексный режим передачи 322
- Загрузка программ 295—299  
 Загрузочный модуль 299  
 Загрузчик 295  
 — абсолютный двоичный 296  
 — настраивающий 298  
 — программ раскрутки 298  
 — связывающий 299  
 Задание 304  
 — обработка в мультипрограммном режиме 308—310  
 — список 310  
 Задержка распространения 100  
 Записи на магнитной ленте 332—334  
 Запоминающее устройство (ЗУ), используемое в АРЛ 365—367  
 — — на дисках типа Винчестер 268  
 — — — сапфино-силиконовых элементах 219  
 — — последовательного доступа 195  
 — — типа ячейки данных 340
- Идемпотентности закон 73  
 «И» операция 66, 67  
 «ИЛИ» операция 67, 72, 73  
 «И — НЕ» операция 90—92  
 Инверсия 67
- Инверторы 88  
 Индекс 27  
 Индексная переменная 27  
 Индикаторная панель 42  
 «ИЛИ-НЕ» операция 93, 94  
 «И-НЕ» операция 93, 94  
 Индукция (метод) 73, 74  
 Инкрементный графопостроитель 324, 325  
 Интегральная инжекционная логика (И<sup>2</sup>Л) 219  
 Интегральные схемы (ИС)  
 — — памяти 199—203, 207—209  
 — — памяти произвольного доступа 218, 219  
 — — ЗУ на биполярных транзисторах 219—223  
 Интерактивная обработка  
 — — интерпретатор 295  
 — — управление ОС 315—319  
 Интервал синхронизации (такт) 24  
 Интервальный таймер 316  
 Интерпретатор 14, 295  
 — АРЛ 364, 365  
 — в системе разделения времени 355, 356  
 «Исключающее ИЛИ-НЕ» операция 89—95  
 «Исключающее ИЛИ» операция 89—95  
 Исходная программа 36
- Каналы 312  
 — ввода-вывода 22, 46  
 Каноническое произведение макстермов 77, 78  
 — — минтермов 76, 77  
 Карты Карно 79—87  
 — Холлерита 326  
 Кассеты лент 279—282  
 Клавиатура 322  
 Классификация ЭВМ 14—18  
 Код Грея 56, 57  
 — рефлексивный (отраженный) 57  
 — самодополняющийся 55  
 — с избытком три 55—56  
 — — исправлением ошибки 60—64  
 — — обнаружением ошибки 60—64  
 — Хэминга 60—64  
 — циклический 56  
 Кодовый набор (слова) 54, 58, 59  
 Команда 22  
 — перехода 22  
 — последовательность команд 22, 33—35, 43, 45  
 Командные процедуры 387, 388

- Комбинационная схема 68, 88  
— — логические диаграммы 87, 88  
Коммутативности закон 72  
Компилятор 14, 35, 294, 295  
— системы разделения времени 355  
Компиляция 35  
Конденсатор 218, 228, 229  
Консоль (индикаторная панель) 42  
Контроллер ввода-вывода 22  
Контрольная сумма 226  
Контрольные разряды 61—64  
Концентратор телефонных линий  
351—399  
— в сети информационного обслужи-  
вания GE 399  
Кратковременное планирование 311
- Лента бумажная 328, 329  
— — перфорированная 41  
— — хранение программ 395, 396  
— — магнитная 272—279  
— — в устройствах долговременной  
памяти 329—336  
— — кассеты и обоймы 279—282  
Ленточный перфоратор 41  
Линейно упорядоченная программа  
28  
Линия выборки  
— — биполярной памяти 220—222  
— — данных биполярной памяти 222  
— — динамической памяти 229  
— — статической памяти на МОП-  
транзисторах 224  
— — переноса 145, 146  
— — предварительной загрузки 129  
— — разряда 220  
— — статического ЗУ на МОП-транзи-  
сторах 224—226  
— — управления направлением счета  
129  
— — Чтения/Записи (R/W) 213  
Логика обрабатывающая 21  
— — управляющая 21  
Логическая очередь 394  
Логические вентили 87—95  
— — обозначение 88, 89  
— — диаграммы 88  
— — примеры 121, 126, 128  
— — операции с АЛУ 183—187  
— — схемы 87, 88  
— — таблицы состояний 101  
Логическое сложение 67  
Логическое умножение 66, 67  
— — устройства ввода 41  
— — — долговременного хранения  
330—336  
Магнитные диски 261—269  
— — устройства долговременной па-  
мяти 336—339  
— — — промежуточного хранения  
341, 342  
Магнитный барабан 255—259  
— — ЗУ промежуточного хранения  
342  
— — параллельного и последователь-  
ного действия 259—261  
— — параллельные операции 259—  
261  
Макроассемблер 294  
Макрокоманды (макросы) 294  
Макстерм 78  
Мантисса 187  
Маскирование (логическое умноже-  
ние) 66, 67  
Массив атрибутов пользователя  
(UADS) 390, 392  
Массовая память 342—344  
Матрица магнитных сердечников  
242—245  
Машинный язык 14, 293, 294  
Межблочный промежуток 273—332  
Метка конца файла 335  
— сектора 338  
Методы цифровой записи 284, 285  
МикроЭВМ 32  
Минимальная сумма 79—84  
Минимальное произведение 84—86  
Мини-ЭВМ 32  
Минтерм 72  
Многоступенчатый дешифратор 205  
Моделирование в системе разделения  
времени 374—386  
Модем 347  
Модули (пакеты) магнитных дисков  
268  
«Монтажное ИЛИ» 203  
МОП-память 216, 217  
— — динамическая 228—231  
— — постоянная 234—236  
— — статическая 223—236  
МОП-транзисторы  
— — комплементарные 218  
— — *n*-канальные 224  
— — *p*-канальные 224, 225  
Мультипрограммирование 308—310  
— — управление ОС 310—315
- Магнитная лента 272—279  
— — кассеты и обоймы 279—282
- Набор данных (файл) 235  
Назначение задания (в TSO) 396

- Накапливающий сумматор (аккумулятор) 139, 140
- Накладные расходы
- — интерактивного режима 317, 318
  - — системы разделения времени 359, 360
- Накопитель (см. Память и ЗУ)
- на магнитной ленте 275
- Настраивающий (перемещающий) загрузчик 298
- Невзвешенные коды 55—57
- Неопределенные условия (состояние) 86, 87
- Область локальной системной очереди** 393
- Обмотка разряда 254
- считывания 240—242, 254
- Обнаружение и направление ошибок 58—64
- Обрабатывающая логика 21
- Обратный код 147—149, 161—166
- Объектная программа (модуль) 299
- Одноразрядный двоичный сумматор 143—145
- — — конструкция 154—157
- Операнды 21, 22, 26
- Оперативные терминалы 322, 323
- Операторы APL 365, 366
- Операции ЦП 43—46
- булевой алгебры 65—68
  - — — логические схемы 87, 88
  - — универсальная 91
  - условные 169
- Операционная система (ОС) 17, 303—310
- Операционные регистры 194
- Оптическое сканирующее устройство 329
- Организация памяти с линейной выборкой 199—203
- Основание системы счисления 49
- Основная (главная) память (ОП) 37—40
- Остаточная намагниченность 240
- Отрицание 64
- выражения 76
- Отрицательная логика 97, 129
- Отрицательные числа
- — в двоично-десятичной системе 160
  - — — двоичной системе с дополнительным кодом 149—151
  - — — двоичной системе с обратным кодом 147—149
  - — — десятичной системе с обратным кодом 161—166
- Ошибки в системах разделения времени 352—355
- Пакет дисков** 337
- Пакетная обработка 305—307
- Память в последовательностных схемах 96, 134—137
- вторичная 40, 321
  - гибкие магнитные диски 261—269 336—341
  - долговременного хранения 329—340
  - магнитные барабаны 255—259
  - — диски 272—283
  - — сердечники 238—248
  - на цилиндрических магнитных доменах (пузырьковая) 238—284
  - основная 38
  - полупроводниковая 38, 218—231
  - постоянная 231—238
  - произвольного доступа 38, 196—199, 218, 219
  - промежуточного хранения 340—344
- Параллельная передача 133
- Параллельные структуры 46
- Параллельный двоичный сумматор 144—146
- Параметры в TSO 384—388
- Переключательная алгебра 66
- Переменные булевой алгебры 66, 72—74, 79
- Перемещаемые головки ЗУ на дисках 263
- Перенос 145, 146
- линия 145, 146
- Переполнение в параллельном арифметическом элементе 154
- — — двоичном сумматоре 146
- Перфокарты 24, 25, 37, 326, 327
- Печатающее устройство 36, 41
- Пирамидальный дешифратор 205
- Плавающая точка, арифметические операции 192, 193
- — реализация подпрограммами 187
  - — система представления чисел 187—192
  - — система со смещением 190
- Планирование
- интерактивных вычислений 315 316

- мультипрограммирования 310, 311
- прерывания устройств ввода-вывода 314, 315
- при моделировании измерений производительности 374—377
- системы разделения времени 357 358
- циклического обслуживания 316 381
- Поглощения закон 72
- Положительная логика 97
- Положительные числа 146
- Полудуплексный режим передачи 323
- Полупроводниковая память
  - — биполярная 219—223
  - — динамическая 228—231
  - — произвольного доступа 218, 219
  - — статическая на МОП-транзисторах 223—228
- Полусумматор 142, 145
  - в одноразрядном сумматоре 143, 144
- Пользователь системы реального времени 349—358
- Пользовательские программы 21, 292
- Пользовательский режим 314
- Порядок 187
- Последовательная передача 133
  - работа магнитного барабана 259—261
- Последовательно-параллельная система 164
- Последовательностные схемы 68
  - — определение 96
  - — примеры 128—137
  - — проектирования 118—128
- Поток данных 23, 30
- Предусилитель 225—227
- Преобразование 23
  - из одной системы счисления в другую 50—52
  - логической диаграммы в таблицу состояний 114—118
  - параллельных данных в последовательные 126—133
  - последовательных данных в параллельные 133
  - таблицы состояний в логическую диаграмму 108—114
- Преобразователь (данных) 21
- Прерывание
  - программа обработки 45
  - схема 346
- Прикладные (пользовательские) программы 21
- Принцип двойственности 73
- Приоритеты в мультипрограммировании 311
  - — в системе разделения времени 362
  - прерывания устройств ввода-вывода 314, 315
- Проблемно настроенные машины 19
- Проверка кодами Хэмминга 60—64
  - на четкость 58—59
- Программа 13, 21, 31, 35, 292
  - загрузчик 295—299
  - линейно-упорядоченная 28
  - обработки прерывания 44
  - прикладная 21
  - связи режима разделения времени 289
  - системная 21, 292, 293
  - системы разделения времени 352—357
  - хранимая на перфоленте 295, 296
  - циклическая 28
- Пограммирование 26, 27
  - языки 13, 14
- Программируемый ввод-вывод 46
- Программное обеспечение 21, 292
- Программный счетчик (ПС) 22, 44
- Пространство — время, соотношение 29, 30
- Процесс 311, 312
  - управление 312, 313
- Процессор предварительной обработки (предпроцессор) 318
  - центральный (ЦП) 22, 23, 33
- Рабочая нагрузка системы разделения времени 369
  - — измерение 369—371
  - — моделирование 370—386
- Разделение времени 17, 30
- Разметка
  - магнитного диска 337—338
  - магнитной ленты 332—335
- Разрешения памяти сигнал (МЕ) 212, 215, 216
- Разряд знака 140
- Разрядные линии 220—224
- Разрядный усилитель записи 228
- Распределение сигналов во времени (тактирование) 24, 42
- Расширенная область связывания и упаковки 393
  - память на сердечниках CDS 343
  - таблица истинности 110—113
- Реверсивный счетчик 122—124
- Регистр 22, 40, 44



- Регистр адреса памяти (РАП) 44,  
198—203  
— АЛУ 139, 140, 168—170  
— буфера 48  
— буферный памяти 251—253  
— данных памяти (РДП) 44  
— команд (РК) 44, 45  
— общего назначения 44  
— операционный 194  
— переполнение 154  
— сдвиговой 124—128, 133, 134  
— — примеры 124—128
- Редактирование при разделении времени 352
- Редактор связей 299—302
- Режим супервизора 314
- Ремапт язык 400
- Ресурсы  
— распределение в пакетном режиме 305—307  
— сети информационного обслуживания 399
- Рефлексивные коды 57
- Самодополняющиеся коды 55
- Световое перо 325
- Свопинг 348—358  
— моделирование для измерения производительности систем 374—377
- Связывающий загрузчик 299
- Связь в сети информационного обслуживания GE 398—400  
— ЭВМ и терминалов 322
- Сдвиг, операция 166—168
- Сдвиговые регистры 133, 134  
— — примеры 124—128
- Сектор магнитного барабана 342  
— метка 338
- Сигнал, распределенный во времени 42  
— прерывания 45
- Символ, базовый элемент информации 334  
— внешний 299—302  
— интерактивный режим 317—319  
— коды представления 326—328
- Симплексный режим передачи 323
- Синхронизация  
— интервал 24  
— сигнал 29  
— триггерных схем 98—100
- Синхросигналы 97, 98
- Система двумерной выборки 242—245  
— разделения времени 17, 30, 345—358  
— — — ALP 363—368  
— — — выбор кванта времени 358—360  
— — — измерение производительности 368—374  
— — — имитатор 374—386  
— — счисления 49—53  
— — восьмеричная 293  
— — двоичная 49, 54  
— — — с дополнительным кодом 149—151  
— — — обратным кодом 147—149  
— — — двоично-десятичная (ДД)  
— — — с обратным кодом 161—166  
— — — десятичная 49  
— — — со смещением 190  
— — с плавающей точкой 187—192
- Системные программы (программное обеспечение) 21, 292, 293
- Словарь внешних символов (СВС) 300—303  
— переменных (СП) 300—303
- Слово 38, 139  
— ЗУ с произвольным доступом 197, 198  
— расширенной памяти на сердечниках 343
- Сложение  
— булевой алгебры 67  
— в двоичной системе с дополнительным кодом 149—151  
— — — с обратным кодом 147—149  
— — — десятичной системе с обратным кодом 161—166  
— двоичное 53  
— логическое 184  
— накапливающий регистр 140  
— по модулю два 183  
— с плавающей точкой 193
- Совершенная индукция 73, 74
- Состояние (триггерной схемы) 96, 97
- Список заданий 310
- СППЗУ (стираемое программируемое ПЗУ) 237
- Спулинг 307, 308  
— ввода 307, 308  
— вывода 308
- Статическая память 196, 218  
— — на МОП-транзисторах 223—228
- Сумма по модулю 2 193
- Сумматор (однозарядный) 143—146, 154—157
- Схема комбинационная 68, 88  
— — — примеры 128—137  
— — — проектирования 118—128

- последовательностная 68, 88
- триггерная 96—101
- синхронная 98
- Сценарий 371—374
- Счетчик 122
- десятичный 118—122
- команд (СК) 22
- программ (СП) 22, 44, 45
- реверсивный двоичный 122—124
- Считывание без разрушения 197
- область 240—242, 254
- операция 26
- Таблица**
- истинности (комбинации) 68, 69
- переходов 101—103, 109
- состояний, описываемых значениями напряжений 97
- Такт 24
- Тактирование
- интерактивные вычисления 315—316
- памяти на сердечнике 248, 249
- Телеобработка 17
- Телетайп 25, 33, 36
- Телефонные линии в системе разделения времени 347
- Теоремы булевой алгебры 72—74
- Терм 71
- Терминал
- администратора системы 370
- дистанционного ввода заданий 17
- интерактивных вычислений 315—319
- реального времени (оперативные) 322, 323
- систем разделения времени 346
- устройство ввода-вывода 32, 33
- Терминальная мониторная программа (ТМП) 390, 391
- Тракт (транзакция) 348
- Транзисторы 16
- биполярной памяти 219, 220
- динамической памяти 228, 229
- Транслятор 14, 21, 294, 295
- Трансляторы языков 21
- Триггерная схема 96—101
- АЛУ 139—141, 168
- биполярные ИС 219—223
- буферных регистров памяти 251—253
- на МОП-транзисторах 223, 224
- полупроводниковой памяти произвольного доступа 218
- элементов памяти 196, 199—201
- Трестабильный буфер 137
- Умножение**
- АЛУ 169, 170
- в булевой алгебре 66, 67
- двоичное 171—175
- десятичное 175, 176
- логическое 183
- с плавающей точкой 192, 193
- Универсальные операции булевой алгебры 91
- Управление 24
- адресными (координатными) шинами  $X$  и  $Y$  249—251
- сигнал 212
- устройство 42, 43
- Управляющая логика 21
- Управляющие провода 212
- Упрощение булевых выражений 74, 75
- Усилитель записи разрядный 228
- Усилитель считывания
- биполярных ЗУ 221
- динамических МОП-ЗУ 229
- ЗУ на магнитных сердечниках 245, 253
- статических МОП-ЗУ 227
- Условные операции 169
- Устройство
- ввода 33, 36, 37
- вывода 40—42
- графического ввода 33, 37
- отображения 323—326
- памяти 37—40
- твердой копии 321
- управления 42, 43
- — передачей сигналов 347
- Файл (набор данных) 235**
- Фиксатор (буфер данных) 134
- Фортран 16, 354, 358
- Функции арифметические АЛУ 138, 139
- булевой алгебры (см. Операции) 90—95
- Хэмминга код 60—64**
- Целочисленная двоичная система со знаком 141**
- Целые числа
- представление в АЛУ 140—142
- Центральный процессор (ЦП) 22, 23
- АЛУ 40, 138, 193
- в интерактивном режиме 316—319

— — временное квантование 359, 360  
 — — мультипрограммирование 308—315  
 — — операции 43—46  
 — — операционные системы 303—307  
 — — разделение времени 346  
 — — стоимость 351  
 — — шины 212—218  
 Циклическая программа 28  
 Циклические коды 56  
 Цифровая вычислительная машина (ЭВМ) 18, 19, 32  
 Запись, методы 284, 285  
 Цифровой дифференциальный анализатор 18

**Четвертьсумматор** 143

**Число**, представление в АЛУ 146, 147  
**Чтение** без разрушения 197

**Шаги** 304

— задания 304

**Шестнадцатеричная система** счисления 293

**Шина** 46, 47

— ввода-вывода 47

— данных 47

— кристалла памяти 212, 218

— общая 47

— памяти 46

— центрального процессора 218—221

**ЭВМ**

— история 14—18

— классификация 18—20

— ENIAC 16

— Harvard Mark 1—16

— UNIVAC I 16

— UNIVAC 1108 190

Эквивалентность операций 89, 94, 95

Элементы памяти 134—137, 194, 195

**Язык** 13, 14, 16, 17

— Алгол 17

— высокого уровня 293, 294

— интерактивных вычислений 315

— Кобол 17, 294

— машинный 14, 293, 294

— программирования 13, 14

— ПЛ/1 17, 294

— Ремант 400

— сети информационного обслуживания GE 400

— системы разделения времени 348

— — — — APL 363—368

APL 363—368

ASCII 327

BASIC 348

DYNAMO 349

EBCDIC 36, 327

# ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА	5
ТЕНДЕНЦИИ В ПРОИЗВОДСТВЕ микроЭВМ	7
ПРЕДИСЛОВИЕ	10
<b>Глава 1. ИСТОРИЯ СОЗДАНИЯ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И ОСНОВНЫЕ ПОНЯТИЯ. Г. Хеллерман</b>	13
1.1. Введение	13
1.2. Исторический обзор	14
1.3. Классификация автоматических вычислительных машин	18
1.4. Структура вычислительной системы	20
1.5. Принципы организации аппаратуры	21
1.6. Требования к запоминающим устройствам	25
1.7. Элементы программирования	26
1.8. Соотношение «пространство — время»	29
<b>Глава 2. СТРУКТУРЫ ЭВМ. В. Гамахер, З. Вранежич, С. Заки</b>	32
2.1. Введение	32
2.2. Функциональные устройства	32
2.3. Устройства ввода	36
2.4. Устройство памяти	37
2.5. Арифметико-логические устройства	40
2.6. Устройства вывода	40
2.7. Устройство управления	42
2.8. Основные операционные понятия	43
2.9. Структуры шин	46
<b>Глава 3. СИСТЕМЫ СЧИСЛЕНИЯ И КОДЫ. З. Кохави</b>	49
3.1. Системы счисления	49
3.2. Двоичные коды	54
3.3. Обнаружение и исправление ошибок	58
<b>Глава 4. БУЛЕВА АЛГЕБРА И ЛОГИЧЕСКИЕ СХЕМЫ. Д. Гивоне, Р. Рессер</b>	65
4.1. Введение	65
4.2. Булева алгебра	65
4.3. Таблицы истинности и булевы выражения	68
4.4. Теоремы булевой алгебры	72
4.5. Применение теорем булевой алгебры	74
4.6. Упрощение булевых выражений с помощью карт Карно	79
4.7. Логические схемы	87
4.8. Логические вентили других типов	89

<b>Глава 5.</b>	<b>ПОСЛЕДОВАТЕЛЬНОСТНЫЕ СХЕМЫ. Дж. Голт, Р. Пиммел</b>	<b>96</b>
5.1.	Введение	96
5.2.	Триггерный элемент	96
5.3.	Таблицы и диаграммы состояний	101
5.4.	Преобразование таблицы состояний в логическую диаграмму	108
5.5.	Преобразование логической диаграммы в таблицу состояний	114
5.6.	Примеры проектирования последовательностных схем	118
5.7.	Примеры некоторых важных последовательностных схем	128
<b>Глава 6.</b>	<b>АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО. Т. Барти</b>	<b>138</b>
6.1.	Введение	138
6.2.	Конструкция АЛУ	139
6.3.	Представление целых чисел	140
6.4.	Двоичный полусумматор	142
6.5.	Одноразрядный сумматор	143
6.6.	Параллельный двоичный сумматор	144
6.7.	Положительные и отрицательные числа	146
6.8.	Сложение в системе с обратным кодом	147
6.9.	Сложение в системе с дополнительным кодом	149
6.10.	Сложение и вычитание чисел в параллельном арифметическом блоке	151
6.11.	Конструкции сумматоров	154
6.12.	Двоично-десятичный сумматор	157
6.13.	Положительные и отрицательные числа в двоично-десятичной форме	160
6.14.	Сложение и вычитание чисел в системе с обратным кодом	161
6.15.	Операция сдвига	166
6.16.	Основные операции	168
6.17.	Двоичное умножение	171
6.18.	Десятичное умножение	175
6.19.	Деление	176
6.20.	Логические операции	183
6.21.	Системы представления чисел с плавающей точкой	187
6.22.	Выполнение арифметических операций над числами с плавающей точкой	192
<b>Глава 7.</b>	<b>ЭЛЕМЕНТ ПАМЯТИ. Т. Барти</b>	<b>194</b>
7.1.	Введение	194
7.2.	Запоминающее устройство с произвольным доступом	196
7.3.	Организация памяти с линейной выборкой	199
7.4.	Дешифраторы	203
7.5.	Системы адресации памяти	206
7.6.	Связь кристаллов памяти с шиной ЭВМ	212
7.7.	Полупроводниковые запоминающие устройства с произвольным доступом	218
7.8.	Запоминающие устройства на биполярных ИС	219
7.9.	Статические запоминающие устройства на МОП-транзисторах	223
7.10.	Динамические запоминающие устройства	228
7.11.	Постоянные запоминающие устройства	231
7.12.	Память на магнитных сердечниках	238
7.13.	Запоминание информации в двумерной матрице магнитных сердечников	242
7.14.	Блок плат сердечников в памяти на сердечниках	245

7.15. Временная последовательность	248
7.16. Управление адресными линиями $X$ и $Y$	249
7.17. Буферный регистр памяти и связанные с ним схемы	251
7.18. Организация памяти на сердечниках и прошивка сердечников	253
7.19. Память на магнитном барабане	255
7.20. Магнитные барабаны параллельного и последовательного действия	259
7.21. Запоминающие устройства на магнитных дисках	261
7.22. Запоминающие устройства на гибких дисках (флоппи-дисках)	269
7.23. Магнитная лента	272
7.24. Кассеты и обоймы	279
7.25. Память на цилиндрических магнитных доменах и память на приборах с зарядовой связью	283
7.26. Методы цифровой записи	284
7.27. Методы записи с возвращением к нулю и с возвращением к смещению	285
7.28. Методы записи без возвращения к нулю	288
<b>Глава 8. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ. В. Гамахер, З. Вранесик, С. Заки</b>	292
8.1. Введение	292
8.2. Языки и трансляторы	293
8.3. Загрузчики	295
8.4. Редакторы связей	299
8.5. Операционные системы	303
<b>Глава 9. УСТРОЙСТВА ВВОДА, ВЫВОДА И ДОПОЛНИТЕЛЬНОЙ ПАМЯТИ. К. Гизр</b>	320
9.1. Введение	320
9.2. Устройства ввода-вывода	321
9.3. Устройства долговременной памяти и промежуточного ввода-вывода	329
9.4. Запоминающие устройства промежуточного хранения	340
9.5. Сравнение скорости и емкости	344
<b>Глава 10. СИСТЕМЫ С РАЗДЕЛЕНИЕМ ВРЕМЕНИ. Г. Хеллерман, Т. Конрой</b>	345
10.1. Введение	345
10.2. Точка зрения пользователя и некоторые ее следствия	349
10.3. Выбор кванта времени	358
10.4. Система MIT CTSS	360
10.5. Система APL	363
10.6. Измерение производительности	368
10.7. Имитатор системы с разделением времени	374
10.8. Режим разделения времени IBM (TSO — Timesharing Option) для системы 360/370	386
10.9. Сеть информационного обслуживания GE	397

### УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Ваши замечания о содержании книги, ее оформлении, качестве перевода и другие просим присылать по адресу: 129820, Москва, И-110, ГСП, 1-й Рижский пер., д. 2, изд-во «Мир».

Герберт Хеллерман, В. Карл Гамахер, Звонко Вранежич,  
З. Кохави, Д. Гивоне, Р. Рессер, Дж. Голт, Р. Пиммел,  
Т. Барт, З. Вранесик, С. Заки, К. Гиэр, Т. Конрой

КОМПЬЮТЕРЫ. Т. 1

Ст. научный редактор Л. П. Якименко  
Младший научный редактор Н. И. Сивилева  
Художник С. А. Бычков  
Художественный редактор Н. М. Иванов  
Технический редактор И. М. Кренделева  
Корректор С. А. Денисова

ИБ № 5497

Сдано в набор 24.05.85. Подписано к печати 20.12.85. Формат  
60×90<sup>1</sup>/<sub>16</sub>. Бумага кн.-журн. Печать высокая. Гарнитура  
латинская. Объем 13,00 бум. л. Усл. печ. л. 26,00. Усл.  
кр.-отт. 26,00. Уч.-изд. л. 24,78. Изд. № 41/4002. Тираж 50 000 экз.  
Зак. № 638. Цена 1 р. 60 к.

ИЗДАТЕЛЬСТВО «МИР»  
129820, ГСП, Москва, И-110, 1-й Рижский пер., 2.

Ленинградская типография № 2 головное предприятие орде-  
на Трудового Красного Знамени Ленинградского объединения  
«Техническая книга» им. Евгении Соколовой Союзполиграф-  
прома при Государственном комитете СССР по делам изда-  
тельств, полиграфии и книжной торговли. 198052, г. Ленин-  
град, Л-52, Измайловский проспект, 29.



В ИЗДАТЕЛЬСТВЕ «МИР»  
В 1985 г. ВЫШЛА КНИГА

**Д. Ван Тассел. СТИЛЬ, РАЗРАБОТКА, ЭФФЕКТИВНОСТЬ, ОТЛАДКА И ИСПЫТАНИЕ ПРОГРАММ:** Пер. с англ. — 2-е изд., испр., 332 с., ил., цена 1 р. 90 к.

В книге американского автора рассмотрен широкий круг вопросов, касающихся стиля написания программ, рациональных методов их разработки и оптимизации, стратегии отладки и тестирования. Приводимые рекомендации основаны на многолетнем опыте автора.

Для специалистов с разной подготовкой — от начинающих программистов до профессионалов, а также для студентов, обучающихся программированию.

