В.И. Пономаренко, Е.Е. Лапшева

ИНФОРМАТИКА. ТЕХНИЧЕСКИЕ СРЕДСТВА

Саратов
«Научная книга»
2009

УДК 004.2, 004.3
ББК 32.97
П56

Рецензенты:

Б.П. Безручко, д-р физ.-мат. наук, профессор, зав. кафедрой динамического моделирования и биомедицинской инженерии Саратовского государственного университета им. Н.Г. Чернышевского

Е.П. Селезнев, д-р физ.-мат. наук, вед. науч. сотрудник Саратовского филиала Института радиотехники и электроники им. В.А. Котельникова РАН

Пономаренко В.И., Лапшева Е.Е.

П56 Информатика. Технические средства : учеб. пособие / В.И. Пономаренко, Е.Е. Лапшева. – Саратов : Научная книга, 2009. – 212 с. : ил.

ISBN 978-5-9758-1140-0

Учебное пособие дает целостное представление о структуре и взаимодействии основных узлов современного компьютера, о работе некоторых периферийных устройств. Предназначено для студентов направлений 210100 «Электроника и микроэлектроника» и 200300 «Биомедицинская инженерия», а также специальностей 200104 «Микроэлектроника и твердотельная электроника», 220501 «Управление качеством», 210601 «Нанотехнологии в электронике», 150601 «Материаловедение и технология новых материалов», обучающихся на факультете нано- и биомедицинских технологий Саратовского государственного университета им. Н.Г. Чернышевского и других вузов.

Может быть полезно для начинающих и опытных программистов, а также для школьников старших классов.

УДК 004.2, 004.3
ББК 32.97

ISBN 978-5-9758-1140-0

ОГЛАВЛЕНИЕ

Предисловие

Информатика – это наука, изучающая все аспекты получения, хранения, преобразования, передачи и использования информации[1]. В настоящее время информатика охватывает все виды человеческой деятельности, связанные с применением компьютеров. С информатикой часто связывают одно из следующих понятий: это либо совокупность определенных средств преобразования информации, либо фундаментальная наука, либо отрасль производства, либо прикладная дисциплина.

Теоретическая информатика является математической дисциплиной, использующей математические методы для построения и изучения моделей обработки информации, и создает теоретический фундамент, на котором построена вся информатика. Информатика как совокупность средств обработки информации включает в себя технические средства (hardware) и программные продукты (software).

Software – это программы, выполняемые вычислительной системой (как исполняемые файлы, так и символические записи программ).

Технические средства, hardware, – это материальная часть вычислительных систем. В состав технических средств входят компьютеры и связанные с ними периферийные устройства (мониторы, клавиатуры, принтеры и плоттеры, модемы и т. д.), линии связи, средства оргтехники и другие материальные ресурсы, которые обеспечивают преобразование информации. Среди этих устройств компьютер можно считать основным, играющим главную и «руководящую» роль. При этом под компьютером можно понимать не только системный блок персонального компьютера или стойку суперкомпьютера, но и любое программируемое устройство, независимо от его размеров (например, карманный компьютер или даже специальная микросхема, называемая микроконтроллером). Компьютер предназначен для решения самых различных задач по преобразованию информации. Выполнение конкретной задачи определяется программным средством, под управлением которого функционирует компьютер. К программным средствам относятся операционные системы, системы программирования и проектирования программных продуктов, различные прикладные пакеты (текстовые и графические редакторы, средства для математических расчетов и моделирования, бухгалтерские программы, издательские системы и др.).

---

[1] Информатика: Энциклопедический словарь для начинающих / сост. Д.А. Поспелов. М. : Педагогика-Пресс, 1994. 352 с. : ил.

Предлагаемое пособие посвящено описанию технических средств, предназначенных для решения задач обработки информации. Эта тема слишком обширна для того, чтобы уместить ее в одной книге, поэтому в первую очередь в этом пособии описаны принципы построения основных средств вычисления, а также некоторые способы передачи информации. Кроме того, уделяется внимание вопросам построения некоторых важных периферийных устройств. Затрагиваются вопросы развития новых методов обработки информации.

Где можно применить полученные в рамках этого курса знания? Практически везде! Если Вы, читатель, будущий программист, то эти знания жизненно необходимы. В том случае, когда программист четко представляет структуру компьютера и каким образом в нем представлены данные, то он сможет организовать свою программу более эффективно. Если Вы хотите программировать микроконтроллеры – без этих базовых знаний тем более не обойтись. Придется, правда, почитать еще несколько книжек, но если непонятна эта, то и те, другие, поставят Вас в тупик. Если Вы собираетесь просто обрабатывать данные (данные измерений или статистические данные), Вы обязательно должны прочитать эту книгу, потому что столкнетесь с различными представлениями чисел и Вам придется перекодировать их из одного вида в другой. Если Вы будете разрабатывать электронные схемы – Вам совершенно необходимо знать основы логики, базовые логические схемы и принципы цифровой обработки информации. Наверняка здесь перечислены не все направления деятельности, в которых пригодятся эти знания, и следует сказать, что если Вы в своей будущей деятельности будете иметь хоть какое-то отношение к естественным наукам, Вам пригодятся эти знания. Да и многие гуманитарии с успехом смогут применить их на практике.

Лекция 1. ИСТОРИЯ И РАЗВИТИЕ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Развитие электронной и вычислительной техники

В развитии электронной техники выделяют несколько поколений. Первое поколение аппаратуры – это техника на электронных лампах, второе – на дискретных полупроводниковых элементах, а третье – с использованием интегральных микросхем. В некоторых случаях, подчеркивая важное значение повышения степени интеграции микросхем, к аппаратуре четвертого поколения относят устройства на СБИС (сверхбольшие интегральные микросхемы). Параллельно развивалась и компьютерная техника, поэтому в компьютерной технике также выделяют четыре поколения машин, существенно различающихся своей элементной базой. В настоящее время определяющий акцент поколений все более смещается с элементной базы на другие показатели: логическая архитектура, программное обеспечение, интерфейс с пользователем, сферы приложения. В этом смысле персональные компьютеры можно отнести к новому, пятому поколению ЭВМ, поскольку они удовлетворяют качественно новым функциональным требованиям. Эти компьютеры компактны, доступны и в то же время обладают достаточно высокой вычислительной мощностью. В начале XX века речь идет о разработке единого глобального информационного пространства, доступ к которому должен быть обеспечен большей части земного населения. Современные машины должны обслуживать новейшие информационные технологии, позволяющие пользователям персональных компьютеров общаться друг с другом и получать информацию любого вида, распределенную по информационным центрам во всем мире.

Однако, несмотря на все многообразие существующей вычислительной техники, в ней заложены принципы, разработанные достаточно давно. Предшественники конструкторов современной электронной вычислительной техники разработали принципы вычислений, продумали структуру и возможности электронных вычислительных машин.

Создателями первых механических калькуляторов были Блез Паскаль, Вильгельм Шиккард и Готфрид Вильгельм Лейбниц. Ради справедливости следует также отметить, что еще Леонардо да Винчи занимался конструированием суммирующего устройства. В 1967 году в Мадриде были найдены эскизы его 13-разрядного суммирующего устройства, причем машина, построенная в соответствии с чертежами в наше время, оказалась работоспособной.

Идея использования двоичной системы счисления принадлежит Лейбницу. Дальнейшее развитие эта идея получила в трудах Джорджа Буля, который внес решающий вклад в создание алгебры логики. Принцип программного управления был предложен Чарльзом Бэббиджем в его проекте аналитической машины. Бэббидж внес значительный вклад в создание техники для автоматизации вычислений. В 1822 году им была построена действующая модель разностной машины, которая могла выполнять вычисления для составления таблиц различных функций. В 1936 году английский математик Алан Тьюринг создает умозрительную «машину Тьюринга», которая была прообразом электронных цифровых компьютеров по своему логическому устройству. Позднее Клод Шеннон показал, что алгебра логики может с успехом использоваться для анализа и синтеза электрических схем с использованием переключателей и реле. Работы Шеннона и Тьюринга считают поворотными в истории вычислительной техники.

Первой электронной вычислительной машиной принято считать ENIAC (Electronic Numerical Integrator And Computer). Его создатели – это американские ученые Джон Маучли и Джон Эккерт. Проект по разработке этого компьютера был секретным и поддерживался артиллерийским департаментом США (был одобрен правительством 9 апреля 1943 года). Компьютер был впервые запущен 14 февраля 1946 года, в день Святого Валентина. Он весил 30 тонн и содержал 18 000 электронных ламп, 6 000 переключателей, 10 000 конденсаторов и 4 000 неоновых лампочек для индикации состояния различных узлов компьютера. ENIAC мог умножать за 2.8 мсек, делить за 24 мсек. По тем временам это был огромный проект. Самым большим электронным оборудованием было электронное радарное устройство, содержавшее 200 ламп. В ENIAC была использована десятичная система счисления, самое длинное число содержало двадцать цифр, а программа вычислений не сохранялась в памяти и не могла быть изменена. Компьютер проработал до 22 октября 1955 года и использовался для вычисления таблиц стрельбы, создания водородной бомбы, а также для аэродинамических расчетов и предсказания погоды.

При работе над этим проектом Маучли и Эккерт поняли, что его можно значительно улучшить. Уже в декабре 1943 года они стали думать относительно проектирования усовершенствованного компьютера, который обеспечил бы хранение не только данных, но и команд. Новый проект назывался EDVAC (Electronic Discrete Variable Automatic Computer). В этом проекте Джон Эккерт впервые выдвинул концепцию хранимой в памяти программы. Она состояла в том, что программы вычислений хранилась в той же памяти, что и данные, а команды, составляющие программу, по форме не отличаются от чисел и позволяют проводить с ними такие же операции.

Осенью 1944 года военный представитель проекта Герман Голдстайн пригласил в качестве консультанта Джона фон Неймана, блестя-

щего математика. Познакомившись с проектом EDVAC, Нейман очень им заинтересовался и понял его высокую значимость. В июне 1945 года Нейман подготовил отчет «Предварительный доклад о машине EDVAC», в котором дал описание основных элементов компьютера и логики его работы. Голдстайн, не посоветовавшись с основными авторами проекта, Маучли и Эккертом, разослал этот доклад известным ученым Америки и Англии. Доклад произвел большое впечатление в научном мире, а поскольку имя Неймана было широко известно, то никто не усомнился в его приоритете. Архитектуру компьютера, основанную на положениях этого доклада, называют неймановской. В 1946 году была опубликована статья «Предварительное рассмотрение логической конструкции электронного вычислительного устройства», которая получила еще большее распространение.

В это послевоенное время идея создания электронного вычислительного устройства витала в воздухе. В Англии Тьюринг принимал непосредственное участие в работе Национальной физической лаборатории, где в 1945 году была организована группа по проектированию и созданию вычислительной машины ACE (Automatic Computing Engine). С 1945 по 1948 год он сделал первые наброски ACE и внес ряд предложений по ее конструированию. Отчет Тьюринга по ACE датирован более поздней датой и ссылается на отчет фон Неймана по EDVAC. Но Тьюринг пошел значительно дальше, поскольку его работа содержала множество конкретных деталей и имела полную концепцию компьютера с хранимой программой.

В нашей стране одним из создателей первых образцов отечественной вычислительной техники был Сергей Алексеевич Лебедев. В 1947 году в Институте электротехники АН УССР была организована лаборатория моделирования и вычислительной техники. До конца 1948 года в лаборатории были разработаны общие принципы построения электронной цифровой вычислительной машины, до конца 1949 года была разработана блок-схема и общая компоновка машины. Лебедев руководил этими работами. 6 ноября 1950 года был произведен пробный пуск МЭСМ (малой электронной счетной машины) и начато решение на ней тестовых и простейших практических задач.

В 1948 году в Москве создается Институт точной механики и вычислительной техники (ИТМ и ВМ АН СССР). Лебедев переходит на работу в этот институт в 1950 году. Он создает специальную лабораторию для создания БЭСМ-1 (быстродействующая электронная счетная машина). В 1948 году произошло поистине знаменательное событие – был изобретен транзистор, полупроводниковый прибор, усиливающий сигнал. Изобретатели – американские ученые Вильям Шокли, Джон Бардин и Уолтер Браттейн. Этот год считается началом эры полупроводниковой электроники. Новая элементная база не сразу, но стала активно использоваться при создании вычислительной техники. В 1960-х

годах отечественная промышленность приступила к серийному выпуску полупроводниковых приборов. Ламповые машины переводят на новую элементную базу. Сначала были построены полупроводниковые машины БЭСМ-3М и БЭСМ-4, которые повторяли архитектуру ламповой БЭСМ-1. В дальнейшем строится новая машина, максимально использующая возможности новой элементной базы. В 1965 году под руководством Лебедева запущен в опытную эксплуатацию макет машины БЭСМ-6. Ее быстродействие – миллион операций в секунду, в ее состав входят 60 000 транзисторов и 180 000 полупроводниковых диодов. Последняя разработка Лебедева – это отечественный суперкомпьютер «Эльбрус». Всю свою жизнь Лебедев боролся с поклонением Западу и был против прямого копирования и внедрения в России компьютеров известной американской фирмы IBM. Однако политики к его мнению не прислушались и посчитали продвижение иностранных марок компьютеров более выгодным. В советское время у нас был налажен выпуск ЕС ЭВМ (электронные вычислительные машины единой серии), которые были копиями американской системы IBM/360.

Величайшим конструктором компьютеров является американец Джин Амдал. Он был главным конструктором и разработчиком уже выпускавшихся серийно компьютеров IBM. В 1953 году он становится главным проектировщиком IBM 704. Эта машина отличалась высокой скоростью работы, а данные в ней представлялись в форме с плавающей запятой. Кроме того, это был первый компьютер, в котором был реализован первый язык программирования высокого уровня Fortran. Затем была следующая машина, IBM 709. После ламповых машин были выпущены их аналоги на полупроводниках – IBM 7090 и IBM 7094. Джин Амдал многое сделал в качестве архитектора компьютерного семейства третьего поколения IBM 360. Он по праву считается создателем машин общего назначения – мэйнфреймов, которые охватили в свое время большой сегмент рынка.

Электроника в это время развивалась по пути миниатюризации. Схемы усложнялись, число элементов (и особенно число межсоединений) увеличивалось. Известно, что увеличение числа межсоединений приводит к понижению надежности схем, усложнению их конструкции, увеличению стоимости. Поэтому конструкторы всегда искали пути преодоления этих проблем. Поиски решения проблемы межсоединений привели к созданию интегральных микросхем, в которых все элементы изготавливаются в едином технологическом цикле на поверхности подложки и имеют герметичный корпус. Первые интегральные микросхемы были созданы в начале 1959 года Джеком Килби и Робертом Нойсом. С этой даты началось бурное развитие интегральной электроники, или микроэлектроники. Вычислительные устройства тоже пытались заключить во все меньший объем. Сначала это были системы небольшой степени интеграции, но с развитием технологии количество активных элементов в микросхемах увеличивалось.

В 1971 году фирма Intel представила микропроцессор Intel 4004 и его архитектора Теда Хоффа. Это изобретение было настолько важным для развития вычислительной техники, что Тед Хофф был признан одним из величайших ученых XX века. К работе над микропроцессором Хофф приступил еще в 1969 году, когда разрабатывал новый набор микросхем для программируемых калькуляторов. При работе над этой задачей Хоффу удалось спроектировать единую универсальную микросхему – центральный процессор ЭВМ общего назначения. При наличии дополнительных микросхем, отвечавших за хранение программы и общение с внешними устройствами, микропроцессор можно было использовать без каких-либо переделок во многих электронных приборах. В конце 1971 года микропроцессор 4004 официально появился на рынке. Микросхема стоимостью 200 долларов выполняла 60 тыс. операций в секунду, содержала 2 300 транзисторов и обладала вычислительной мощью первого электронного компьютера – ENIAC.

С этих пор развитие микропроцессоров идет небывалыми темпами. В 1978 году появился микропроцессор с числом элементов 20 000. В процессоре Pentium Pro размещались уже 3.5 млн транзисторов и изготавливают их по субмикронной технологии (ширина дорожки составляет около 0.35 мкм). Еще более современные компьютеры изготавливаются по 0.09 мкм технологии. Процессор Intel Pentium D – это процессор с двумя ядрами на одном кристалле. Каждое ядро процессора имеет собственный кэш второго уровня L2 объемом 1 Мбайт, соответственно общий объем кэша L2 составляет 2 Мбайта. Процессор производится по 90-нанометровому технологическому процессу, при этом размер самого кристалла процессора составляет 206 мм^2, а количество транзисторов внутри процессора – 230 млн. Казалось бы, такой мощный двухъядерный процессор будет выделять чрезмерно много тепла и потребует эффективной системы охлаждения. Однако процессор поглощает всего 130 Вт, а максимальная температура поверхности кристалла не превосходит 70 градусов Цельсия. Напряжение питания ядра процессора составляет от 1.25 до 1.388 В, а максимальный ток – 125 А.

Количественно динамику роста производительности охарактеризовал еще Гордон Мур. В 1965 году, в процессе подготовки выступления, он сделал замечательное наблюдение. Представив в виде графика рост производительности микросхем памяти, он обнаружил любопытную закономерность: новые модели микросхем разрабатывались спустя более или менее одинаковые периоды (1,5–2 года) после появления их предшественников, а емкость их при этом возрастала каждый раз примерно вдвое. Если такая тенденция продолжится, заключил Мур, то мощность вычислительных устройств будет расти со временем экспоненциально. Когда статья вышла в свет, число транзисторов на среднем микрочипе было весьма скромным – всего 60.

Таблица 1.1. Некоторые процессоры фирмы Intel

Тип процессора	**Год выхода**	**Число транзисторов**
4004	1971	2250
8008	1972	2500
8080	1974	5000
8086	1978	29000
286	1982	120000
386	1985	275000
486 DX	1989	1180000
Pentium	1993	3100000
Pentium II	1997	7500000
Pentium III	1999	2,4E+07
Pentium 4	2001	4,2E+07
Pentium D	2005	2,3E+08

Наблюдение Мура, еще не возведенное в то время в ранг закона, впоследствии блестяще подтвердилось, а обнаруженная им закономерность наблюдается и в наши дни, причем с поразительной точностью, являясь основой для многочисленных прогнозов роста не только производительности, но и других характеристик вычислительной техники.

В табл. 1.1 приведена зависимость количества транзисторов на кристалле микропроцессора фирмы Intel за последние 35 лет. На рис. 1.1 эта зависимость представлена в логарифмическом масштабе. Она практически линейна, и таким образом, количество транзисторов растет по экспоненциальному закону. Из этого закона есть множество следствий. Например, закон Рока: «Стоимость оборудования и других основных фондов, используемых в производстве полупроводников, удваивается каждые четыре года».

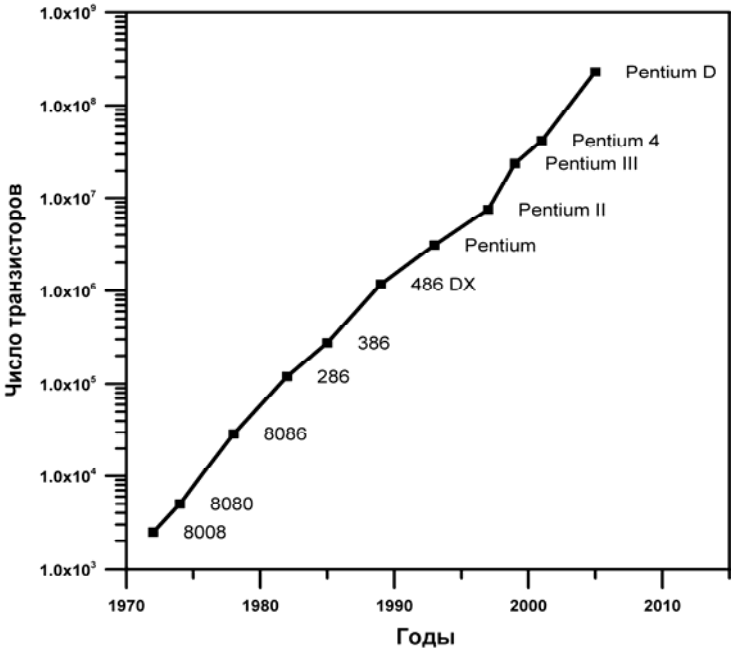


Рис. 1.1. Темпы роста степени интеграции за последние 30 лет

Основные характеристики, соответствующие различным поколениям ЭВМ, приведены в табл. 1.2.

Таблица 1.2. Характеристики поколений ЭВМ

Поколение ЭВМ	**I**	**II**	**III**	**IV**	**V**
Хронологические границы периодов	Начало 50-х – середина 50-х годов	Конец 50-х – середина 60-х годов	Конец 60-х – начало 70-х годов	Середина 70-х и далее	Начало 90-х и далее
Элементная база процессора	Вакуумные лампы	Полупроводниковые транзисторы	Интегральные схемы	Большие и сверхбольшие интегральные схемы	Сверхбольшие интегральные схемы, многоядерность
Производительность (количество операций в секунду)	10^4	10^6	10^7	10^8	более 10^8
Емкость ОЗУ (бит)	10^4	10^6	10^7	$0.5*10^8$	более 10^9
Программное обеспечение, языки программирования	Машинный язык, библиотеки стандартных программ	Добавляются: языки высокого уровня, трансляторы с этих языков	Добавляются: языки управления заданиями, операционные системы, пакеты прикладных программ	Добавляются: непроцедурные языки, генераторы программ, операционные системы реального времени	Добавляются: эффективная работа с аудио- и видеоматериалами, распознавание речи и т. д.
Режим пользования	Монопольный, прохождением задачи управляет пользователь	Монопольный, прохождением задачи управляет оператор	Пакетный, коллективный, прохождением задач управляет ОС	Коллективный. Параллельно решаются несколько задач	Персональный и коллективный. Также параллельно решаются несколько задач
Область применения	Научные расчеты	Добавляются технические расчеты	Добавляются экономические расчеты	Добавляются управление большими системами	Персональное и профессиональное использование
Типичный представитель	БЭСМ IBM-701	БЭСМ-6 IBM-7090	ЕС-1060 IBM-370/75	«Эльбрус» КРЕЙ-1	Персональные ЭВМ, персональные супер-ЭВМ, большие супер-ЭВМ

Классификация компьютеров

Характеристик компьютеров очень много, к ним относятся система адресации и система команд, особенности архитектуры, наличие конвейера и кэша и т. д. Для сравнения компьютеров между собой мы будем использовать такие характеристики, как производительность, тактовая частота и разрядность. Производительность измеряется в MIPS (Millions Instruction Per Second) и MFLOPS (Millions Floating point Operation Per Second). В общем случае MIPS есть скорость операций в единицу времени, т. е. для любой данной программы MIPS есть просто отношение количества команд в программе к времени ее выполнения. Таким образом, производительность может быть определена как обратная к времени выполнения величина, причем более быстрые машины при этом будут иметь более высокий рейтинг MIPS. Как единица измерения, MFLOPS предназначена для оценки производительности только операций с плавающей точкой и поэтому не применима вне этой ограниченной области. Например, программы компиляторов имеют рейтинг MFLOPS близкий к нулю вне зависимости от того, насколько быстра машина, поскольку компиляторы редко используют арифметику с плавающей точкой.

Производительность, измеренная в MFLOPS (в русской транскрипции Мфлопс), зависит от машины и от программы. Этот термин базируется на количестве выполняемых операций, а не на количестве выполняемых команд. По мнению многих программистов, одна и та же программа, работающая на различных компьютерах, будет выполнять различное количество команд, но одно и то же количество операций с плавающей точкой. Именно поэтому рейтинг MFLOPS предназначался для справедливого сравнения различных машин между собой. Данная величина определяется путем запуска на испытуемом компьютере тестовой программы, которая решает задачу с известным количеством операций и подсчитывает время, за которое она была решена. Наиболее популярным тестом производительности на сегодняшний день является программа LINPACK, используемая, в том числе, при составлении рейтинга суперкомпьютеров TOP500.

Тактовая частота измеряется в герцах и определяет скорость работы основных «часов» компьютера. Современные персональные компьютеры работают на частоте несколько гигагерц.

Разрядность – это параметр, отражающий количество одновременно отрабатываемых электронным устройством разрядов, длина слова, обрабатываемого процессором. Микропроцессор 8086 был 8-разрядным, в дальнейшем разрядность увеличивалась. Термин применим к составным частям счетных или измерительных устройств: индикаторам, шинам данных компьютеров, процессорам и т. д.

Развитие электроники привело к тому, что компьютеры сейчас распространены повсеместно, они стали доступны большому числу

пользователей, и в настоящее время можно выделить условно следующие большие классы компьютеров, различающиеся по своим характеристикам: суперкомпьютеры, просто большие компьютеры (мэйнфреймы), миникомпьютеры и персональные ЭВМ. Кроме того, можно отдельно рассматривать кластерные архитектуры (распределенные системы), объединяющие большое число компьютеров, что значительно повышает вычислительные мощности.

Суперкомпьютеры – это сверхдорогие устройства, созданные в одном или нескольких экземплярах. Вообще говоря, этот термин до сих пор не имеет четкого определения. Область применимости суперкомпьютеров – решение задач, требующих гигантских объемов вычислений, к которым относятся аэродинамические задачи, сейсмический анализ, задачи вычислительной химии, предсказания погоды, криптографии, сложные задачи виртуальной реальности и многие другие. Они обычно работают в крупных научных центрах, где работа связана с необходимостью больших вычислительных мощностей. Отцом суперкомпьютеров по праву считают американца Сеймура Крея.

Работая в своей компании CDC (Control Data Corp), он разработал модель компьютера CDC 1604. Эта модель была выпущена в 1958 году и содержала 100 000 диодов, 25 000 транзисторов и память на магнитных сердечниках емкостью 32 768 48-разрядных слов. Эта модель имела большой успех, поскольку ее огромным преимуществом по сравнению с машинами подобного класса была низкая цена. В 1962 году CDC объявила о выходе модели CDC 6600, который стал самым мощным компьютером того времени, причем был намного дешевле и компактнее тогдашнего фаворита фирмы IBM. В 1969-м была создана модель CDC 7600, которая многими считается первым суперкомпьютером. В 1972 году Крэй основал новую фирму, Cray Recearch, и в марте 1976 года выпустил уникальный компьютер Cray-1, который был установлен в Лос-Аламосском научном центре. В этом суперкомпьютере архитектура была подчинена идеям параллельной обработки информации. Производительность его достигала 160 MFLOPS.

Весной 1985 года был запущен первый образец компьютера Cray-2, который приобрела Ливерморская национальная лаборатория в Калифорнии. Для охлаждения плотно упакованных плат Крэй погрузил их в охлаждающую жидкость. По этой причине Cray-2 называли «компьютером в аквариуме». В последующих суперкомпьютерах – Cray-3 и Cray-4 высокая производительность обеспечивалась применением новой элементной базы на основе арсенида галлия. Cray-3 был выпущен в начале 1990-х годов и имел производительность 16 GFLOPS, объем памяти 2048 млн слов и 16 процессоров в своей конструкции. Правда, за два года Крэй не смог продать ни одного компьютера Cray-3 и заявил о своем банкротстве.

Мейнфрейм – это синоним понятия «большая универсальная ЭВМ». Мейнфреймы и до сегодняшнего дня остаются наиболее

мощными (не считая суперкомпьютеров) вычислительными системами общего назначения, обеспечивающими непрерывный круглосуточный режим эксплуатации. Они могут включать один или несколько процессоров, каждый из которых, в свою очередь, может оснащаться векторными сопроцессорами (ускорителями операций с суперкомпьютерной производительностью). В нашем сознании мейнфреймы все еще ассоциируются с большими по габаритам машинами, требующими специально оборудованных помещений с системами водяного охлаждения и кондиционирования. Однако это не совсем так. Прогресс в области элементно-конструкторской базы позволил существенно сократить габариты основных устройств. Наряду со сверхмощными мейнфреймами, требующими организации двухконтурной водяной системы охлаждения, имеются менее мощные модели, для охлаждения которых достаточно принудительной воздушной вентиляции, и модели, построенные по блочно-модульному принципу и не требующие специальных помещений и кондиционеров.

В архитектурном плане мейнфреймы представляют собой многопроцессорные системы, содержащие один или несколько центральных и периферийных процессоров с общей памятью, связанных между собой высокоскоростными магистралями передачи данных. При этом основная вычислительная нагрузка ложится на центральные процессоры, а периферийные процессоры (в терминологии IBM – селекторные, блок-мультиплексные, мультиплексные каналы и процессоры телеобработки) обеспечивают работу с широкой номенклатурой периферийных устройств.

Первоначально мейнфреймы ориентировались на централизованную модель вычислений, работали под управлением патентованных операционных систем и имели ограниченные возможности для объединения в единую систему оборудования различных фирм-поставщиков. Однако повышенный интерес потребителей к открытым системам, построенным на базе международных стандартов и позволяющим достаточно эффективно использовать все преимущества такого подхода, заставил поставщиков мейнфреймов существенно расширить возможности своих операционных систем в направлении совместимости.

Стремительный рост производительности персональных компьютеров, рабочих станций и серверов создал тенденцию перехода с мейнфреймов на компьютеры менее дорогих классов. Эта тенденция получила название «разукрупнение» (downsizing). Главным недостатком мейнфреймов в настоящее время остается относительно низкое соотношение производительность/стоимость. Тем не менее они продолжают работать в крупных объединениях, банках, системах обработки данных, аэропортах и т. д.

Миникомпьютеры предназначены для работы в качестве серверов небольших сетей, в многопользовательских системах. Их называют

также рабочими станциями. Первоначальная ориентация рабочих станций на профессиональных пользователей (в отличие от персональных компьютеров (ПК), которые вначале ориентировались на самого широкого потребителя непрофессионала) привела к тому, что рабочие станции – это хорошо сбалансированные системы, в которых высокое быстродействие сочетается с большим объемом оперативной и внешней памяти, высокопроизводительными внутренними магистралями, высококачественной и быстродействующей графической подсистемой и разнообразными устройствами ввода/вывода.

Изобретателем миникомпьютеров считают американского ученого Гордона Белла. Он работал в компании DEC (Digital Equipment Corporation), которая была создана в 1957 году, и ее целью было производство более дешевых и компактных компьютеров. Компьютеры Белла линейки PDP (PDP-4, 5, 6, 8, 10, 11) были настолько удачными, что проработали очень долгое время. Копии PDP-8 и PDP-11 производились и в Советском Союзе. Несмотря на то, что миникомпьютеры уступали по производительности большим компьютерам, их мощности было достаточно для решения многих задач, в том числе задач управления. Приемлемая цена оправдывала все недостатки миникомпьютера (например, у PDP-8 разрядность составляла 12 бит, а объем памяти – 4 кбайт). В 1975 году Белл и коллектив, который он возглавлял, закончили разработку следующей модели – VAX-11. Ее индекс говорил о том, что она является логическим продолжением PDP-11 и разработана на его основе. Система VAX стала образцом в компьютерной промышленности того времени, а модель VAX-11/780 стала самой популярной в классе миникомпьютеров. Компьютеры этой серии использовались в США даже шире, чем модели известнейшего семейства компьютеров IBM 360/370.

В настоящее время быстрый рост производительности ПК на базе новейших микропроцессоров Intel и AMD в сочетании с резким снижением цен на эти изделия и развитием технологии локальных шин, позволяющей устранить многие «узкие места» в архитектуре ПК, делают современные персональные компьютеры весьма привлекательной альтернативой миникомпьютерам (рабочим станциям).

Персональные компьютеры (ПК) появились в результате эволюции миникомпьютеров при переходе элементной базы машин с малой и средней степенью интеграции на большие и сверхбольшие интегральные схемы. ПК, благодаря своей низкой стоимости, очень быстро завоевали хорошие позиции на компьютерном рынке и создали предпосылки для разработки новых программных средств, ориентированных на конечного пользователя. Это прежде всего – «дружественные пользовательские интерфейсы», а также проблемно-ориентированные среды и инструментальные средства для автоматизации разработки прикладных программ.

По мере продолжения процесса разукрупнения (downsizing) и увеличения производительности платформы Intel наиболее мощные ПК стали использоваться в качестве серверов, постепенно заменяя миникомпьютеры. Кроме того, с ростом производительности появилась возможность поручать персональным компьютерам все более и более сложные и трудоемкие задачи. Применение ПК стало более разнообразным. Помимо обычных для этого класса систем текстовых процессоров, даже средний пользователь ПК может теперь работать сразу с несколькими прикладными пакетами, включая электронные таблицы, базы данных и высококачественную графику. ПК активно работают с мультимедийными приложениями, их производительности хватает для показа и обработки видео, работы со звуком и т. д.

Создание первых персональных компьютеров связано с именами Стива Джобса и Стива Возняка, которые в 1976 году провели презентацию компьютера Apple I. Для существовавших тогда компьютерных компаний это было большим сюрпризом, поскольку они не придавали микрокомпьютерам большого значения только потому, что они не могли делать то, что могли большие компьютеры. В апреле 1977 года появился компьютер Apple II. Он имел клавиатуру и корпус и весил 5,5 кг. Эти компьютеры были ориентированы в основном на игровое применение, а к семи слотам расширения в материнской плате могли подключаться синтезатор звука, дигитайзер, карта графического расширения, модем и другие устройства. С 1976 по 1982 год фирма Apple доминировала на рынке, но с приходом IBM на рынок персональных компьютеров в 1981 году ситуация изменилась. Корпорация IBM сконструировала персональный компьютер IBM PC специально для применения в бизнесе, школе и дома. Менее чем за один год корпорация наладила производство и сбыт своих компьютеров. Важным фактором успеха IBM PC стала так называемая «открытая архитектура», позволяющая другим фирмам приложить свои творческие и предпринимательские способности в его пополнении все новыми возможностями и программами, и тем самым способствовать утверждению этого компьютера как мирового стандарта. Появилось большое число фирм, выпускающих машины, полностью совместимые с IBM PC, а также разрабатывающих дополнительные внешние устройства. На сегодняшний день компьютеры с IBM-совместимой архитектурой составляют около 90 % всех персональных компьютеров.

Кластерные архитектуры предназначены для увеличения производительности при решении сложных задач. Такие системы называют еще распределенными.

Термин «кластеризация» на сегодня в компьютерной промышленности имеет много различных значений. Строгое определение могло бы звучать так: «Реализация объединения машин, представляющегося единым целым для операционной системы, системного программного

обеспечения, прикладных программ и пользователей». Машины, кластеризованные вместе таким способом, могут при отказе одного процессора очень быстро перераспределить работу на другие процессоры внутри кластера. Это, возможно, наиболее важная задача многих поставщиков систем высокой готовности.

Первой концепцию кластерной системы анонсировала компания DEC, определив ее как группу объединенных между собой вычислительных машин, представляющих собой единый узел обработки информации. По существу VAX-кластер представляет собой слабосвязанную многомашинную систему с общей внешней памятью, обеспечивающую единый механизм управления и администрирования.

В настоящее время широкое распространение получила также технология параллельных баз данных. Эта технология позволяет множеству процессоров разделять доступ к единственной базе данных. Распределение заданий по множеству процессорных ресурсов и параллельное их выполнение позволяет достичь более высокого уровня пропускной способности, поддерживать большее число одновременно работающих пользователей и ускорить выполнение сложных запросов.

Параллельные базы данных находят широкое применение в системах обработки обращений к единой базе данных в режиме on-line, системах поддержки принятия решений и часто используются при работе с критически важными для работы предприятий и организаций приложениями, которые эксплуатируются по 24 часа в сутки.

В табл. 1.3 приведены примерные значения производительности различных типов компьютеров.

Таблица 1.3. Сравнение производительности различных типов компьютеров

Тип компьютера	**Год**	**Производительность**
Большие компьютеры и суперкомпьютеры		
ENIAC	1946	300 флопс
IBM 709	1958	5 Мфлопс
Cray-1	1974	160 Мфлопс
Cray Y-MP	1988	2,3 Гфлопс
ASCI Red	1993	1 Тфлопс
Cray XT4/XT3	2004	101,7 Тфлопс
Blue Gene/L	2006	280,6 Тфлопс
Персональные компьютеры		
IBM PC/XT	1983	0,0069 Мфлопс
Intel 80386 40 МГЦ	1985	0,6 Мфлопс
Intel Pentium 75 МГц	1993	7,5 Мфлопс
Intel Pentium II 300 МГц	1997	50 Мфлопс
Intel Pentium III 1 ГГц	1999	320 Мфлопс
AMD Athlon 64 2,211 ГГц	2003	840 Мфлопс
Intel Core 2 Duo 2,4 ГГц	2006	1,3 Гфлопс

Кластерные архитектуры (распределенные системы) предназначены для решения задач с большим объемом вычислений, и их производительность весьма высока. На кластерах проводится поиск лекарств от рака и других заболеваний, поиски внеземных цивилизаций, климатические и погодные расчеты. Быстродействие некоторых распределенных систем представлено в табл. 1.4.

Таблица 1.4. Быстродействие распределенных систем

Название архитектуры	**Производительность**
BOINC	577 Тфлопс
SETI@home	261 Тфлопс
Climate Prediction	62 Тфлопс
Rosetta@home	50 Тфлопс
Folding@home	799 Тфлопс

Несмотря на непрерывный рост производительности компьютеров, предельные показатели достижений микроэлектроники не соответствуют набирающему силу научно-техническому прогрессу. Уже сейчас существует целый ряд задач, ждущих своего разрешения. Среди них – создание систем оперативного распознавания образов, искусственного интеллекта, синтеза конструкций и систем, разработка устройств параллельной обработки информации, устройств управления базой знаний и т. п.

Пути решения стоящих проблем

Идут интенсивные поиски методов, разрабатываются устройства, предназначенные для обработки больших массивов информации в реальном масштабе времени. Анализ схем цифровой обработки изображений показывает, например, что рост их быстродействия приближается к насыщению. При этом ряд упомянутых задач принципиально не может быть решен в рамках современных методов обработки больших информационных массивов, в частности, фон-неймановской схемы построения вычислительных систем.

Ресурсы компьютера являются не просто самоцелью, а еще и средством для решения больших вычислительных задач. Одной из таких задач, требующих суперкомпьютерной мощности, является расчет глобальных климатических изменений на несколько десятков или даже сотен лет вперед. Чтобы увеличить существующую производительность, можно воспользоваться несколькими путями.

Первый путь – это увеличение производительности компьютера, заданное в MIPS или MFLOPS.

Второй – увеличение параллельности вычислений при сохранении двоичной, или дискретной архитектуры компьютера.

Третий – создание принципиально новых способов обработки и хранения информации.

На первом пути один из способов увеличения производительности – это повышение тактовой частоты. С одной стороны, его можно добиться совершенствованием существующей кремниевой технологии. С другой стороны, возможно использование более быстродействующих полупроводниковых структур. Сеймур Крэй, конструктор арсенид-галлиевого суперкомпьютера «Cray-3» показал, что изготовление микросхем из арсенида галлия является дорогим и трудоемким удовольствием. Тем не менее поиски быстродействующих полупроводников активно ведутся. Другой способ – это создание оптической ЭВМ. В настоящее время созданы оптические элементы, позволяющие обеспечить скорость переключения 10^{12} логических операций в секунду. Можно использовать эту новую оптическую технологию для создания оптического компьютера с логической организацией, напоминающей организацию обычной ЭВМ.

Параллельность вычислений достигается при использовании большого числа компьютеров или процессоров. Для эффективного использования этой технологии нужно решить ряд вопросов, связанных с параллельной структурой решаемых задач. Допустим, что задача состоит в моделировании поведения четырех частиц. Тогда нецелесообразно пользоваться машиной со 100 параллельными процессорами. Если же мы имеем дело с задачей, которую можно легко распределить между 10 000 процессоров, то для достижения суммарной производительности 1 Тфлоп нужно 10 000 процессоров со скоростью работы 100 Мфлоп. Кроме того, нужно ответить на вопросы: какова будет связь между процессорами и памятью, структура связей между процессорами и, наконец, кто согласен заплатить за такой компьютер?

Создание принципиально новых способов обработки информации потребует углубления наших знаний в области строения мозга человека и, возможно, совершенно нового взгляда на эту проблему. Те новые способы обработки, о которых уже сейчас может идти речь, – скорее аналоговые, чем цифровые. Похоже, что здесь лучше будут работать не универсальные, а специализированные системы. В этих направлениях ведутся интенсивные исследования. В целом разработаны принципы построения нейронных компьютеров, ведутся исследования возможностей создания квантовых компьютеров.

Таким образом, встает вопрос, что является более эффективным – искать пути сохранения тенденции экспоненциального роста степени интеграции интегральных схем и тем самым расширить возможности схемотехнической микроэлектроники, или искать принципиально новый подход при создании систем обработки больших информационных массивов.

Можно ли удержать тенденцию экспоненциального роста степени интеграции и, соответственно, экспоненциальное снижение стоимости обработки информации? Вот основной вопрос перспективного развития схемотехнической микроэлектроники.

С ростом степени интеграции перед схемотехнической электроникой возникают проблемы.

1. «Тирания межсоединений». По мере роста числа элементов на кристалле становится все более острой задача организации соединений между отдельными элементами. При этом резко увеличивается площадь, занимаемая межсоединениями. При очень большом числе элементов применяют многослойный монтаж, что увеличивает еще и сложность изготовления сверхбольших интегральных микросхем.

2. Проблема пробоя. По мере уменьшения размеров активных элементов в них увеличивается электрическое поле и даже при небольших напряжениях поле может достигать таких больших величин, что увеличивается вероятность пробоя. Поэтому возникает необходимость переходить на более низкие напряжения питания.

3. Проблема отвода рассеиваемой мощности. Уменьшение геометрических размеров элементов приводит к увеличению их сопротивления и, следовательно, к увеличению тепловыделения.

Разработчики ИС активно ищут способы преодоления «тирании межсоединений», пути обхода технологических и физических барьеров. С этой целью разрабатываются вертикальные структуры, в которых стараются разместить максимум элементов в минимальном пространстве. Активные и пассивные элементы схемы размещаются в объеме, и интегральная схема становится трехмерной. Технология «кремний на диэлектрике» открывает определенные перспективы вертикальной интеграции и позволяет получать многоярусные транзисторные структуры. Предполагается, что трехмерные ИС будут иметь высокие быстродействие и плотность упаковки элементов, обладать возможностью параллельной обработки информации и станут многофункциональными. Однако придется преодолеть много препятствий, прежде чем в трехмерных ИС удастся решить проблемы взаимных помех элементов и паразитных наводок между слоями, большой потребляемой мощности и необходимости охлаждения кристалла, разработать методы проектирования схем с комплексными параметрами и сложной топологией поверхностных активных слоев и сделать их конкурентоспособными по цене. Переход в трехмерную электронику отнюдь не решит проблемы межсоединений, напротив, резко усложнит конструкции межуровневых соединений. Надежность таких схем вызывает сомнение, а доказательств обратного пока нет.

Могут ли «спасти» схемотехническую электронику метод интеграции на пластине или создание «суперкристаллов»? Проблема межсоединений в этих случаях тоже принципиально не решается, а значит,

и достижение успеха сомнительно. По этой же причине сомнительны перспективы использования в схемотехнической электронике различных эффективных и сверхминиатюрных транзисторных структур.

Можно ли уйти от проблемы «тирании межсоединений»? Видимо, да. Но для этого нужно уйти от традиционного принципа обработки информации, отказаться от схемотехнической ячейки как основного преобразователя и хранителя информации.

Заметим, что принцип последовательной обработки информации является искусственным, его придумал человек. В природе такой принцип не используется! На сегодняшний день самым совершенным устройством обработки информации можно считать человеческий мозг. Для того чтобы создать устройство, которое будет работать с использованием принципов организации мозга, необходимо ответить на огромное число вопросов, таких как: «Что есть мышление?», «Как человек мыслит?», «Где обитает сознание?» и др. Сложность человеческого мозга пока несопоставима с электронными устройствами. В его составе порядка 1011 нейронов, и он ежесекундно обрабатывает такие объемы информации, которые недоступны современным компьютерам при всем их быстродействии.

В настоящее время активно ведутся работы над созданием квантовых компьютеров, разрабатывается их теоретическая основа. Возможно также, что будут открыты новые возможности для роста производительности компьютеров, будут предложены новые способы обработки информации.

Контрольные вопросы

1. По каким признакам ЭВМ делятся на поколения?
2. Какие характеристики используются для сравнения различных ЭВМ?
3. Какие проблемы стоят перед производителями вычислительной техники и каковы пути их решения?

Список литературы к лекции 1

1. Информатика : учебник / под ред. Н.В. Макаровой. – М. : Финансы и статистика, 1997. – 768 с. : ил.
2. *Частиков А.П.* Архитекторы компьютерного мира. – СПб. : БХВ-Петербург, 2002. – 384 с. : ил.
3. *Полунов Ю.Л.* От абака до компьютера: судьбы людей и машин. Книга для чтения по истории вычислительной техники : в 2 т. – М. : Русская редакция, 2004. – Т. 1. – 480 с. : ил.; Т. 2. – 544 с. : ил.

Лекция 2. АРХИТЕКТУРА ФОН НЕЙМАНА

Архитектура вычислительной машины – это ее внутренняя организация, состав ее различных устройств, логика их работы и взаимодействия. Архитектуру можно рассматривать на различных уровнях детализации, подходящих различным категориям пользователей. Обычный пользователь оперирует минимальным набором сведений об основных технических параметрах (производительность, объем оперативной и дисковой памяти, графическая система, характеристики монитора). Специалист по программированию должен более детально представлять архитектуры машины, поскольку от этого зависит эффективность написанной им программы. С одной стороны, может показаться, что программист не должен так уж много знать о структуре ЭВМ, поскольку современные языки программирования скрывают от него многие важные черты внутреннего устройства, но без знания важных особенностей внутренней организации машины невозможно разработать программу, оптимально использующую возможности компьютера. Специалист по разработке электронной техники на основе процессоров и контроллеров должен знать еще больше, поскольку ему предстоит наладить обмен информации между различными узлами вычислительной системы и другими устройствами. Наконец, разработчик самих вычислительных систем должен детально представлять себе, что делает каждый из элементов и как он взаимодействует с другими.

В понятие архитектуры отдельно взятой вычислительной машины включают следующие сведения:

1. Основные блоки машины, их состав и взаимодействие.
2. Система команд машины и порядок работы команд.
3. Типы данных, обрабатываемые машиной.
4. Организация памяти, иерархия запоминающих устройств и их взаимодействие (регистровая память, кэш-память, оперативная память, внешняя память).
5. Организация системы прерываний.
6. Организация обмена данных с внешними устройствами.
7. Топология связей отдельных устройств и модулей.

В дальнейшем в рамках этого курса мы будем обсуждать различные вопросы архитектуры, а также принципы работы отдельных узлов и внешних (периферийных) устройств.

В настоящее время любому пользователю известно, что практически во всех компьютерах используется архитектура фон Неймана. Это название архитектуры носит имя американского ученого венгерского происхождения Джона фон Неймана (может быть, не совсем заслуженно),

который опубликовал в 1946 году работу «Preliminary discussion of the logical design of an electronic computing instrument» (Предварительное рассмотрение логической конструкции электронного вычислительного устройства).

Рассмотрим основные принципы построения электронных вычислительных машин (ЭВМ), которые обосновал в своей статье фон Нейман. В ходе эволюции компьютеров с 1946 года они практически не изменились, но появилось множество усовершенствований, повышающих эффективность работы вычислительных систем. Подавляющее большинство вычислительных машин в настоящее время являются фон-неймановскими машинами. Тем не менее современные компьютеры используют множество нововведений, неизвестных во времена фон Неймана. В этой лекции будет рассмотрена структура машины фон Неймана и некоторые нововведения, применяющиеся в современной вычислительной технике.

Машина фон Неймана

В упомянутой работе фон Нейман подробно описывает основные компоненты машины, их взаимодействие и логическое устройство. Как таковые, принципы фон Неймана, которые приписываются ему, не описаны в его работе. Они выделены позже его последователями и обычно формулируются следующим образом:

1. Основными блоками машины являются арифметико-логическое устройство, устройство управления, запоминающее устройство и устройства ввода-вывода.
2. Машина работает в двоичной системе счисления.
3. Программы и данные хранятся в одной и той же памяти.
4. Центральный процессор (устройство управления и арифметико-логическое устройство) в соответствии с программой выполняют последовательность команд из оперативной памяти.

Подбор компонентов машины и их взаимодействие определялось этими принципами. Рассмотрим каждый из этих принципов более подробно.

Основные блоки машины фон Неймана

Статья фон Неймана (в соавторстве с Артуром Барксом и Германом Голдстайном) «Предварительное рассмотрение логической конструкции электронного вычислительного устройства» начинается с обсуждения этих основных компонентов. Ниже приведены отрывки этой статьи на языке оригинала и примерный перевод. Анализ работы фон Неймана показывает, насколько глубоко обсуждались основные принципы работы вычислительной машины,

1.1. *Inasmuch as the completed device will be a general-purpose computing machine it should contain certain main organs relating to arithmetic,*

memory-storage, control and connection with the human operator. It is intended that the machine be fully automatic in character, i.e. independent of the human operator after the computation starts. A fuller discussion of the implications of this remark will be given in Sec. 3 below.

1.1. Так как законченное устройство будет универсальной вычислительной машиной, оно должно содержать несколько основных органов, таких как орган арифметики, памяти, управления и связи с оператором. Мы хотим, чтобы после начала вычислений работа машины не зависела от оператора. Более полное обсуждение этого будет дано в разделе 3 ниже.

1.2. *It is evident that the machine must be capable of storing in some manner not only the digital information needed in a given computation such as boundary values, tables of functions* (*such as the equation of state of a fluid*) *and also the intermediate results of the computation* (*which may be wanted for varying lengths of time*)*, but also the instructions which govern the actual routine to be performed on the numerical data. In a special-purpose machine these instructions are an integral part of the device and constitute a part of its design structure. For an all-purpose machine it must be possible to instruct the device to carry out any computation that can be formulated in numerical terms. Hence there must be some organ capable of storing these program orders. There must, moreover, be a unit which can understand these instructions and order their execution.*

1.2. Очевидно, что машина должна быть способна запоминать некоторым способом не только цифровую информацию, необходимую для данного вычисления, такую как граничные условия, таблицы функций (уравнение состояния жидкости) и промежуточные результаты вычислений (которые могут быть необходимы для различных времен), но также и команды, управляющие программой, которая должна производить вычисления над этими числовыми данными. В специализированной вычислительной машине эти команды являются неотъемлемой частью устройства и составляют часть его структуры. В универсальной машине должна быть возможность отдать приказ устройству произвести вообще любое вычисление, которое может быть сформулировано при помощи чисел. Следовательно, должен быть некоторый орган, способный хранить эти приказы программы. Кроме того, должно быть устройство, которое может понимать эти команды и управлять их выполнением.

1.3. *Conceptually we have discussed above two different forms of memory: storage of numbers and storage of orders. If, however, the orders to the machine are reduced to a numerical code and if the machine can in some fashion distinguish a number from an order, the memory organ can be used to store both numbers and orders. The coding of orders into numeric form is discussed in 6.3 below.*

1.3. Выше мы в принципе указали на два различных вида памяти – память чисел и память приказов. Если, однако, команды машины све-

дены к числовому коду и если машина сможет некоторым образом отличать число от команды, то орган памяти можно использовать для хранения как чисел, так и команд. Кодирование команд в числовой форме обсуждается ниже в п. 6.3.

1.4. *If the memory for orders is merely a storage organ there must exist an organ which can automatically execute the orders stored in the memory. We shall call this organ the Control.*

1.4. Если память команд является просто органом памяти, то должен существовать еще орган, который может автоматически выполнять команды, хранящиеся в памяти. Мы будем называть этот орган управлением.

1.5. *Inasmuch as the device is to be a computing machine there must be an arithmetic organ in it which can perform certain of the elementary arithmetic operations. There will be, therefore, a unit capable of adding, subtracting, multiplying and dividing. It will be seen in 6.6 below that it can also perform additional operations that occur quite frequently.*

The operations that the machine will view as elementary are clearly those which are wired into the machine. To illustrate, the operation of multiplication could be eliminated from the device as an elementary process if one were willing to view it as a properly ordered series of additions. Similar remarks apply to division. In general, the inner economy of the arithmetic unit is determined by a compromise between the desire for speed of operation-a non-elementary operation will generally take a long time to perform since it is constituted of a series of orders given by the control-and the desire for simplicity, or cheapness, of the machine.

1.5. Поскольку наше устройство должно быть вычислительной машиной, в нем должен иметься арифметический орган – устройство, способное складывать, вычитать, умножать, делить. Мы увидим также, что оно может выполнять и другие операции, которые встречаются довольно часто…

1.6. *Lastly there must exist devices, the input and output organ, whereby the human operator and the machine can communicate with each other. This organ will be seen below in 4.5, where it is discussed, to constitute a secondary form of automatic memory.*

1.6. Наконец, должен существовать орган ввода и вывода, с помощью которого осуществляется связь между оператором и машиной…

Далее в статье подробно рассмотрены память, принципы кодирования команд и управление вычислениями, логическая и электронная организация машины, арифметический орган, набор управляющих инструкций, преобразование из двоичного кода (удобного для машины) в десятичный (удобный для человека) и обратно.

На современном языке блоки компьютера называются следующим образом:

- *Арифметико-логическое устройство* (АЛУ). Это устройство, в котором производятся операции по обработке информации.

- *Устройство управления* (УУ). Это устройство, обеспечивающее организацию выполнения программы обработки информации и взаимодействие составных частей компьютера.
- *Память* – устройство, обеспечивающее хранение исходных данных, промежуточных значений, результатов обработки и самой программы обработки информации. В настоящее время разделяют ОЗУ (оперативное запоминающее устройство), ПЗУ (постоянное запоминающее устройство), ВЗУ (внешнее запоминающее устройство) и их разновидности.
- *Устройства ввода-вывода*, предназначенные для преобразования информации в форму, принятую в компьютере, и обратно, в форму, доступную для восприятия человеком.

Память машины состояла из 4096 слов (ячеек), каждое из которых содержало 40 бит. Для записи адресов использовалось 12 двоичных разрядов. В одну стандартную ячейку помещались две команды. Каждая из команд содержала адрес информации и 6-битный код команды. Оставшиеся 2 бита не использовались. Внутри АЛУ находился специальный регистр – аккумулятор. Регистром в вычислительной технике называется специализированная ячейка памяти или набор таких ячеек, предназначенных для оперативного обмена информацией. Обычно он имеет небольшой размер.

Операции проводились в целых числах. Фон Нейман считал техническую реализацию арифметики с плавающей точкой, с одной стороны, слишком сложной, а с другой – ненужной, поскольку любой сведущий математик должен быть способен держать плавающую точку в голове.

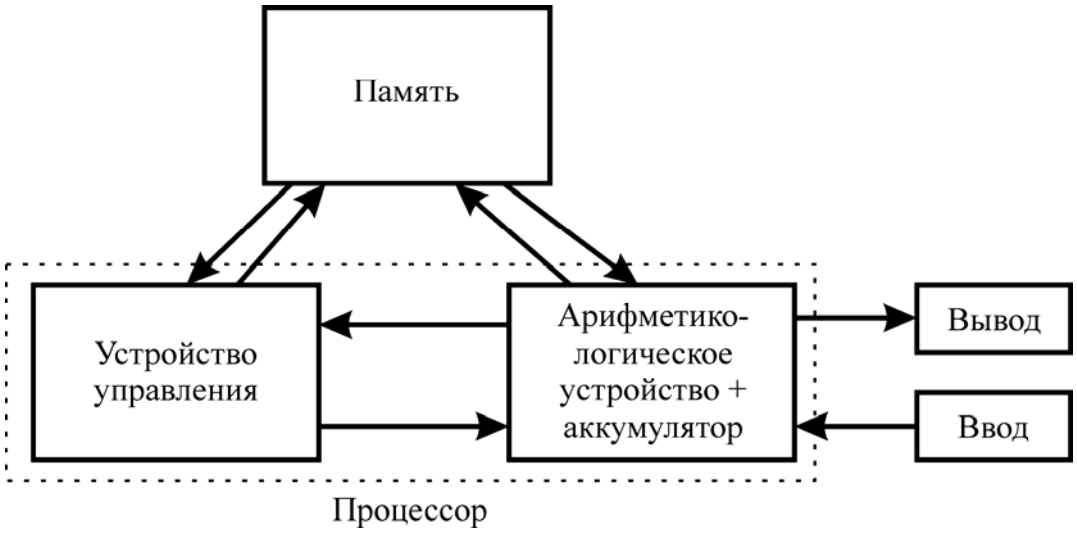


Рис. 2.1. Блок-схема машины, описанной фон Нейманом

Из рис. 2.1 видно, что центральное место в конструкции машины занимает собственно процессор – АЛУ + УУ. Через него проходят все информационные потоки, и он согласовывает работу отдельных устройств.

Система счисления

Что касается выбора системы счисления, то этот вопрос в 1946 году был совсем не тривиальным. В повседневной жизни мы используем десятичную систему счисления, и для восприятия человеком она является наиболее удобной. Однако при создании электронных устройств наиболее подходящей является двоичная система. Это связано с тем, что устройство с двумя состояниями сделать довольно просто, и главным аргументом обоснования двоичной системы фон Нейман назвал техническое удобство.

Вообще говоря, возможно создание устройств, в которых есть несколько устойчивых состояний. В частности, стандартный триггер обычно состоит из двух каскадов, но их может быть несколько. Однако с ростом количества каскадов растет и сложность, и, соответственно, цена.

Другая инженерная причина заключается в том, что для логических операций в двоичном случае уже была подробно разработана так называемая булева алгебра. Для арифметических операций также можно использовать алгебру логики, записав логические выражения для результата вычислений.

Ради справедливости, следует отметить, что двоичная система – не единственная, на которой пробовали делать ЭВМ. В математическом смысле троичная система выгоднее, там числа получаются короче. Кроме того, если используется так называемая «уравновешенная троичная система», то логические ветвления могут быть записаны проще (см. рис. 2.2).

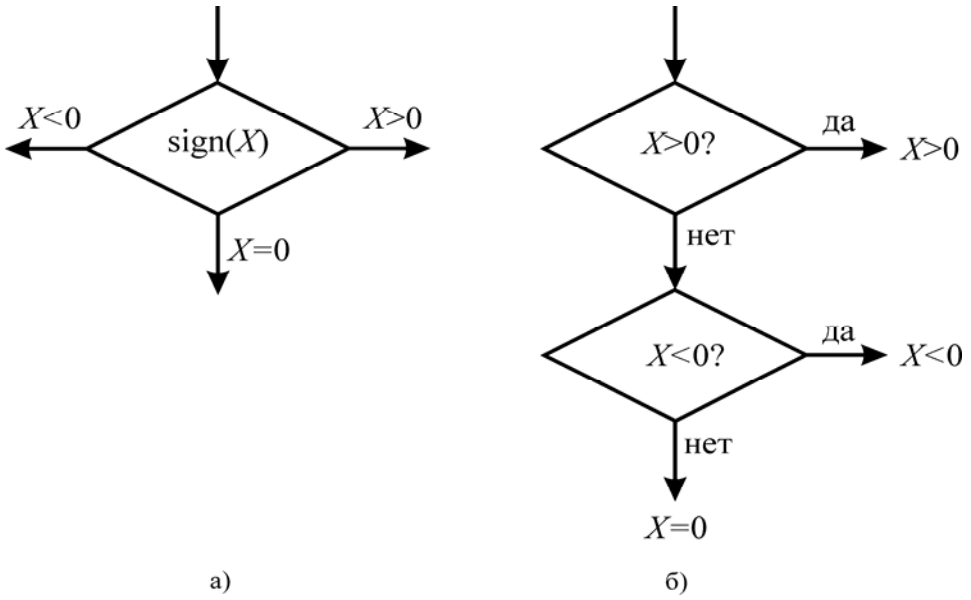


Рис. 2.2. Организация ветвления в троичной (*а*) и двоичной (*б*) системах

В Советском Союзе была построена такая ЭВМ и в 1962–1965 годах она выпускалась промышленным образом. Руководителем разра-

ботки был Николай Петрович Брусенцов. Однако, с развитием кремниевой электроники, троичная элементная база перестала использоваться.

Принцип хранимой программы

Это один из наиболее важных принципов, который до сих пор работает в большинстве компьютеров и заключается в том, что программа записывается в двоичном коде и, таким образом, может храниться в одном запоминающем устройстве вместе с данными. Этот принцип соблюдался не для всех машин. Например, в машине МЭСМ команды записывались в двоичном коде, но сама программа в памяти не хранилась, а для программирования необходимо было соединять провода на наборной панели. Это достаточно трудоемкая процедура, а принцип хранимой программы позволяет не только автоматизировать ввод программы, но и изменять программу в процессе работы.

Иногда память для команд и данных разделяют на два отдельных устройства. Такая архитектура называется гарвардской, поскольку она появилась в 40-х годах XX века в университете Гарварда. В последнее время интерес к ней увеличивается в связи с развитием производства специализированных микросхем, работающих в режиме реального времени. Здесь она обладает преимуществом в том, что наличие двух областей памяти, каждая из которых работает со своей шиной, существенно ускоряет выполнение программы. Хотя, если речь идет о микроконтроллерах, то важным является не столько выделение своей памяти для программы, сколько просто наличие двух независимых устройств ОЗУ, работающих параллельно.

Выполнение программы машиной фон Неймана

Компьютер начинает работу всегда по какой-то программе в соответствии с программным принципом управления. Команды этой программы записаны в памяти. Для организации вычислений устройство управления выполняет определенную последовательность действий, выбирая команды программы по очереди и выполняя их. Для эффективной работы в устройство управления были введены специальные регистры (это запоминающие устройства небольшой емкости, предназначенные для временного хранения команд и данных). Их названия и назначение приведены в табл. 2.1. В арифметико-логическом устройстве также есть свои регистры – аккумулятор (Ac) и арифметический регистр (AR). Аккумулятор предназначен для хранения промежуточных вычислений, ввода и вывода данных, выполнения операций сравнения. Арифметический регистр дополняет аккумулятор там, где это необходимо (например, при умножении, когда разрядность результата больше, чем разрядность каждого из сомножителей).

Таблица 2.1. Регистры устройства управления

Оригинальное название		**Современный русский эквивалент**		**Разрешение (бит)**	**Назначение**
CC	Control Counter	СК	Счетчик Команд	12	Хранит адрес следующей команды
FR	Function Table Register	РК	Регистр Команд	20(18)	Хранит исполняемую команду; 12 разрядов служат адресом при чтении из ОЗУ
CR	Control Register	ДР	Дополнительный Регистр	20(18)	Хранит вторую пару команды
SR	Selection Register	РП	Регистр памяти	40	Обеспечивает обмен данными с ОЗУ

Последовательность работы всей машины пояснена на рис. 2.3, где приведены регистры устройства управления и схема их взаимодействия.

Счетчик команд в машине фон Неймана хранит адрес очередной исполняемой инструкции. Самое первое значение заносил оператор (в современных ЭВМ роль оператора играет постоянное запоминающее устройство). В дальнейшем компьютер автоматически обновляет значение данного регистра.

Исполняемая в данный момент команда хранится в регистре команд РК. 12 младших бит играют особую роль – в них находится адрес для считывания-записи информации в ОЗУ.

Поскольку в одной ячейке упаковано две команды, для хранения второй предусмотрен дополнительный регистр ДР, который в большинстве конструкций отсутствует.

Через регистр памяти процессор принимает данные из ОЗУ и записывает обратно.

Последовательность действий при исполнении каждой команды следующая.

1. Выборка очередной команды из ОЗУ. Для этого адрес очередной команды копируется из СК в РК, младшие 12 разрядов которого одновременно служат регистром адреса данных, считываемых из ОЗУ; затем производится считывание содержимого ячейки памяти в РП. Из регистра памяти эта информация копируется в РК и ДР.
2. К содержимому СК добавляется единица, и он показывает адрес следующей ячейки ОЗУ, в которой располагается программа.
3. Происходит дешифрация и выполнение первой команды (находящейся в РК).
4. Происходит копирование второй команды из ДР в РК, ее дешифрация и выполнение.
5. Если вычисления не закончены, перейти к п. 1.

В случае остановки вычислений фон Нейман предлагал подать звуковой или световой сигнал для оповещения оператора.

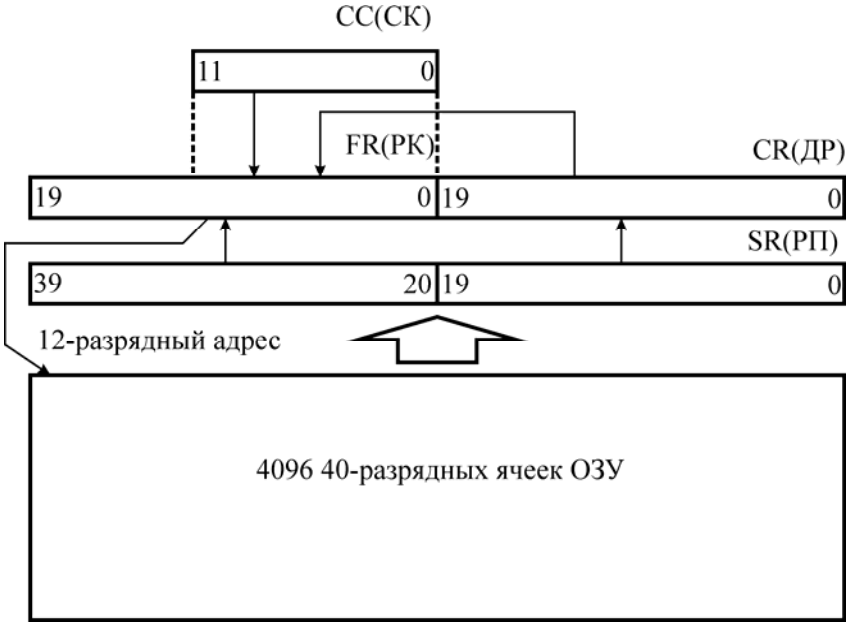


Рис. 2.3. Схема взаимодействия регистров устройства управления машины фон Неймана

При выполнении команд может оказаться необходимым прочитать данные из памяти. Это делается точно так же, как при считывании команды. Однако после считывания данные направляются в другой регистр – аккумулятор, а команды – как и в предыдущем описании, в регистр команд и дополнительный регистр.

Из приведенного описания видно, что мы можем упростить выполнение команд, если их длина будет равна длине слова в памяти. При этом не нужен дополнительный регистр и упрощается процедура выполнения программы. Такая идеология применялась в большинстве более поздних машин.

Контрольные вопросы

1. Что такое архитектура вычислительной машины?
2. Перечислите принципы фон Неймана.
3. Какие основные блоки составляют машину фон Неймана?
4. Какие основные регистры составляют устройство управления машины, описанной фон Нейманом, и как они взаимодействуют в процессе работы?

Список литературы к лекции 2

1. Информатика: Энциклопедический словарь для начинающих / сост. Д.А. Поспелов. – М. : Педагогика-Пресс, 1994. – 352 с. : ил.
2. *Еремин Е.А.* Развитие принципов фон-неймановской архитектуры // Информатика. 2004. № 24. С. 1–32.
3. *Королев Л.Н.* Архитектура электронных вычислительных машин. – М. : Научный мир, 2005. – 272 с. : ил.

Лекция 3. СИСТЕМЫ СЧИСЛЕНИЯ

Понятие системы счисления

Подумайте, сколькими разными способами можно записать число «десять». Один способ уже представлен в предыдущем предложении. Можно назвать еще множество способов написания этого числа: 10, X, ten и т. д. Очевидно, что от написания названия числа его значение – «вес» – не изменяется. Следовательно, под *числом* понимается его величина, а не символьная запись.

Понятие числа является фундаментальным понятием как математики, так и информатики, при помощи которого обозначается какое-либо определенное количество, отображается количественная характеристика предметов и явлений объективной деятельности и объектов из области абстрактных систем, ведутся счет и измерения. Символы, при помощи которых записывается число, называются *цифрами*.

Системой счисления принято называть совокупность приемов обозначения (записи) чисел. Различают позиционные и непозиционные системы счисления.

Непозиционная система счисления – система счисления, в которой для обозначения чисел вводятся специальные знаки, количественное значение которых («вес» символа) всегда одинаково и не зависит от их места в записи числа. Самым известным примером непозиционной системы счисления является римская система счисления. В римской системе счисления для записи числа в качестве цифр используются буквы латинского алфавита.

I – 1 V – 5 X – 10 L – 50 C – 100 D – 500 M – 1000

Для записи чисел в римской системе используются два правила:

1) каждый меньший знак, поставленный слева от большего, вычитается из него;

2) каждый меньший знак, поставленный справа от большего, прибавляется к нему:

III = 1+1+1 = 3 IV = –1+5 = 4 VI = 5+1 = 6

XL = –10+50 = 40 LX = 50+10 = 60 XC = –10+100 = 90

CIX = 100–1+10 = 109 MCMXCVIII = 1000–100+1000–10+100+5+1+1+1 = 1998

Позиционный принцип в системе счисления

Позиционной системой счисления называется система счисления, в которой значение каждой цифры в изображении числа зависит от ее положения в ряду других цифр, изображающих число.

Положение, занимаемой цифрой при письменном обозначении числа называется *разрядом.*

Привычная для нас десятичная система счисления является позиционной. Это значит, что в числе 1978 цифра «1» обозначает одну тысячу. Эта цифра стоит в позиции третьего разряда. Цифра «9» – девять сотен, второй разряд. Цифра «7» – семь десятков, первый разряд. А «8» – восемь единиц, нулевой разряд. Распишем вышесказанное в виде математической формулы:

$$1978 = 1000 + 900 + 70 + 8 = 1 \cdot 1000 + 9 \cdot 100 + 7 \cdot 10 + 8 = 1 \cdot 10^3 + 9 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0$$

Распишем подобным образом дробное число:

$$3019,7294 = 3 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0 + 7 \cdot 10^{-1} + 2 \cdot 10^{-2} + 9 \cdot 10^{-3} + 4 \cdot 10^{-4}.$$

Очевидно, что в десятичной системе счисления числа 10^n, где $n = (-\infty, +\infty)$ – номер разряда, играют ключевую роль в формировании записи числа. Эти числа называются *базисом десятичной системы счисления.* Число 10 для нашей десятичной системы счисления является ее *основанием*. Оно показывает, что каждые десять единиц образуют один десяток, десять десятков образуют одну сотню, десять сотен образуют одну тысячу и т. д. В общем случае, для десятичной системы счисления, каждые десять единиц любого разряда образуют одну единицу соседнего, более старшего разряда.

Базис системы счисления – это последовательность ключевых чисел, каждое из которых задает значение цифры в ее позиции или «вес» каждого разряда.

Выбирая за *основание системы счисления* любое натуральное число *k*, то есть, считая, что *k* единиц любого разряда образует одну единицу соседнего более крупного разряда, придем к так называемой *k-ичной системе счисления с натуральным основанием.*

Если $k < 10$, то цифры от *k* до 9 становятся лишними. Если $k > 10$, то для чисел от 10 до $k–1$ включительно надо придумать специальные обозначения цифр. Для шестнадцатеричной системы счисления приняты обозначения, представленные в табл. 3.1.

Таблица 3.1. Обозначения цифр шестнадцатеричной системы счисления

Число десятеричной системы счисления	**Цифра шестнадцатеричной системы счисления**
10_{10}	A_{16}
11_{10}	B_{16}
12_{10}	C_{16}
13_{10}	D_{16}
14_{10}	E_{16}
15_{10}	F_{16}

Базис двоичной системы счисления:

$$\ldots, 2^{-n}, \ldots, 2^{-2}, 2^{-1}, 1, 2, 4, 8, 16, \ldots, 2^{n}, \ldots$$

Базис восьмеричной системы счисления:

$$\ldots, 8^{-n}, \ldots, 8^{-2}, 8^{-1}, 1, 8, 64, 512, \ldots, 8^{n}, \ldots$$

Или в общем виде:

$$q_{-n} = q^{-n}, \ldots, q_{-2} = q^{-2}, q_{-1} = q^{-1}, q_0 = q^0, q_1 = q, q_2 = q^2, q_3 = q^3, \ldots, q_n = q^n, \ldots,$$

где $q \in N$ и $q \neq 1$. Число q называют *основанием* системы счисления.

Каждое число в любой из таких систем может быть записано в цифровой и многочленной форме:

Цифровая форма:

$$A_q = \left(\overline{a_n a_{n-1} a_{n-2} \ldots a_2 a_1 a_0, a_{-1} a_{-2} \ldots a_{-m}}\right)_q,$$

где a_i – цифра в диапазоне от 0 до q–1.

Многочленная форма:

$$A_q = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \ldots + a_2 \cdot q^2 + a_1 \cdot q + a_0 + a_{-1} \cdot q^{-1} + \ldots + a_{-m} \cdot q^{-m},$$

где q – основание системы счисления.

Некоторые примеры записи чисел в различных системах счисления сведены в таблицу:

Система счисления	**Цифровая форма**	**Многочленная форма**
Десятичная	4509,52	$4 \cdot 10^3 + 5 \cdot 10^2 + 0 \cdot 10^1 + 9 \cdot 10^0 + 5 \cdot 10^{-1} + 2 \cdot 10^{-2}$
Двоичная	11101,011	$\left(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 1^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}\right)_{10} =$ $= (1 \cdot 10^{100} + 1 \cdot 10^{11} + 1 \cdot 10^{10} + 0 \cdot 10^1 + 1 \cdot 10^0 + 0 \cdot 10^{-1} +$ $+ 1 \cdot 10^{-10} + 1 \cdot 10^{-11})_2$
Шестнадца-теричная	A5F,C	$\left(10 \cdot 16^2 + 5 \cdot 16^1 + 15 \cdot 16^0 + 12 \cdot 16^{-1}\right)_{10} =$ $= \left(A \cdot 10^2 + 5 \cdot 10^1 + F \cdot 10^0 + C \cdot 10^{-1}\right)_{16}$

Связь между системами счисления

Научимся переводить целые числа из десятичной системы счисления в любую другую позиционную систему счисления. Для начала обоснуем процедуру перевода числа из десятичной системы счисления в двоичную.

Пусть нам дано десятичное число A_{10}, которое необходимо перевести в двоичную систему счисления. Это означает, что надо найти такие b_i, где $b_i \in \{0,1\}$, для которых $A_{10} = (\overline{b_n b_{n-1} \ldots b_2 b_1 b_0})_2$. Запишем двоичное представление этого числа в многочленной форме:

$$A_{10} = (\overline{b_n b_{n-1} \ldots b_2 b_1 b_0})_2 = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \ldots + b_1 \cdot 2 + b_0.$$

Если отбросить b_0, то видно, что оставшаяся часть делится на 2 нацело. Таким образом, чтобы найти b_0, необходимо найти остаток от деления исходного числа A на 2. $b_0 = A_{10} \bmod 2$, где mod – операция поиска

остатка от деления. Итак, первый найденный остаток от деления на основание новой системы счисления есть цифра нулевого разряда искомого двоичного числа. Целую часть от деления A_{10} на два обозначим x_1:

$$x_1 = \left[\frac{A_{10}}{2}\right] = (\overline{b_m b_{m-1} \ldots b_2 b_1})_2 = b_m \cdot 2^{m-1} + b_{m-1} \cdot 2^{m-2} + \ldots + b_2 \cdot 2 + b_1 .$$

Чтобы найти цифру b_1, необходимо найти остаток отделения x_1 на 2.

$$b_1 = x_1 \bmod 2 .$$

Итак, остаток от второго деления исходного числа на 2 есть цифра первого разряда искомого двоичного числа. На следующем шаге поделим x_1 на два:

$$x_2 = \left[\frac{x_1}{2}\right] = (\overline{b_m b_{m-1} \ldots b_3 b_2})_2 = b_m \cdot 2^{m-2} + b_{m-1} \cdot 2^{m-3} + \ldots + b_3 \cdot 2^1 + b_2 .$$

Продолжаем наши рассуждения-вычисления до тех пор, пока не найдем последние искомые цифры.

$b_{m-1} = x_{m-1} \bmod 2$;

$x_m = \left[\frac{x_{m-1}}{2}\right] = b_m$ – m-й старший разряд искомого двоичного числа.

$b_m = x_m \bmod 2$.

При следующем делении целая часть получается равной нулю. Вычисления останавливаются.

Опробуем разработанный алгоритм на конкретном примере. Проведем число 58_{10} в двоичную систему счисления (см. табл. 3.2.).

Таблица 3.2. Пример перевода десятичного числа в двоичную систему счисления

№	A_{10}	x_i	Остаток b_i
0	58		$b_0 = 58 \bmod 2 = 0$
1		$x_1 = \left[\frac{58}{2}\right] = 29$	$b_1 = 29 \bmod 2 = 1$
2		$x_2 = \left[\frac{29}{2}\right] = 14$	$b_2 = 14 \bmod 2 = 0$
3		$x_3 = \left[\frac{14}{2}\right] = 7$	$b_3 = 7 \bmod 2 = 1$
4		$x_4 = \left[\frac{7}{2}\right] = 3$	$b_4 = 3 \bmod 2 = 1$
5		$x_5 = \left[\frac{3}{2}\right] = 1$	$b_5 = 1 \bmod 2 = 1$
6		$x_6 = \left[\frac{1}{2}\right] = 0$	

Последний полученный остаток – старшая цифра искомого двоичного числа. Итак, $58_{10} = 111010_2$.

Этот алгоритм работает при переводе целого числа из десятичной системы счисления в систему счисления с любым основанием. Сформулируем этот алгоритм.

Алгоритм перевода целого числа из десятичной системы счисления в любую другую. *Для того чтобы исходное целое десятичное число A заменить равным ему целым числом B_p, необходимо число A разделить нацело на новое основание p, выделив целую часть и остаток. Полученную целую часть вновь разделить нацело на основание p и т. д. до тех пор, пока целая часть не станет равной нулю. Цифрами искомого числа B_p являются остатки от деления, выписанные так, чтобы последний полученный остаток стал цифрой старшего разряда числа B_p.*

Для примера рассмотрим перевод десятичного числа в восьмеричную систему счисления. Остатками могут быть цифры от 0 до 7.

Воспользуемся методом «Деление столбиком». Будем отмечать полученные остатки. Такая запись аналогична примеру, приведенному в табл. 3.2.

$278_{10\to8}$

278	8	
24	34	8
38	32	4
32	2	
6		

Записываем остатки, начиная с последнего: $278_{10} = 426_8$.

При переводе числе из десятичной системы счисления в систему счисления, основание которой больше десяти, нужно очень внимательно отнестись к записи цифр, чей «вес» больше или равен десяти. Для примера рассмотрим перевод числа 574 из десятичной в шестнадцатеричную систему счисления.

$574_{10\to16}$

574	16	
48	35	16
94	32	2
80	3	
14		

Остатками здесь служат числа от 0 до 15. Это цифры шестнадцатеричной системы счисления. Старшая цифра – 2, вторая цифра – 3, а младшей является цифра, соответствующая десятичному числу 14. По табл. 3.1 определяем, что это шестнадцатеричная цифра E. Таким образом, для разобранного примера ответ будет следующим: $574_{10} = 23E_{16}$.

Один из часто используемых способов перевода целых чисел из десятичной системы счисления в двоичную – разложение исходного числа на сумму степеней двойки. В искомом двоичном числе единицы

будут стоять в позициях тех разрядов, степени двойки которых присутствуют в разложении. Например:

$$234_{10} = 128 + 64 + 32 + 8 + 2 = 2^7 + 2^6 + 2^5 + 2^3 + 2^1 = \overset{7\,6\,5\,4\,3\,2\,1\,0}{11101010_2}.$$

Теперь разберем алгоритм перевода правильных десятичных дробей в другую позиционную систему счисления. Пусть нам надо перевести число $0, A_{10}$ в двоичную систему счисления. Приведем его запись в цифровой и многочленной форме:

$$0, A_{10} = (0, \overline{b_{-1} b_{-2} b_{-3} \ldots b_{-m}})_2 = b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + \ldots + b_{-m} \cdot 2^{-m} = \frac{b_{-1}}{2} + \frac{b_{-2}}{2^2} + \ldots + \frac{b_{-m}}{2^m}.$$

Для того чтобы найти цифру b_{-1}, нужно умножить $0, A_{10}$ на 2. Целая часть произведения может быть либо 0, либо 1. Это и будет старшая цифра данного числа в двоичной системе счисления.

$$x_{-1} = 0, A_{10} \cdot 2 = \left(\frac{b_{-1}}{2} + \frac{b_{-2}}{2^2} + \ldots + \frac{b_{-m}}{2^m} \right) \cdot 2 = b_{-1} + \frac{b_{-2}}{2} + \ldots + \frac{b_{-m}}{2^{m-1}}.$$

$$b_{-1} = [x_{-1}].$$

Для того чтобы найти следующую цифру искомого двоичного числа, необходимо дробную часть полученного произведения вновь умножить на 2. Целая часть нового произведения будет следующей цифрой более младшего разряда искомого числа. Обозначим дробную часть числа фигурными скобками.

$$x_{-2} = \{x_{-1}\} \cdot 2 = \left(\frac{b_{-2}}{2} + \frac{b_{-3}}{2^2} + \ldots + \frac{b_{-m}}{2^{m-1}} \right) \cdot 2 = b_{-2} + \frac{b_{-3}}{2} + \ldots + \frac{b_{-m}}{2^{m-2}}.$$

$$b_{-2} = [x_{-2}].$$

Будем вычислять эти произведения, пока не найдем цифру последнего разряда искомого двоичного числа.

$$x_{-m+1} = \{x_{-m+2}\} \cdot 2 = \left(\frac{b_{-m+1}}{2} + \frac{b_{-m}}{2^2} \right) \cdot 2 = b_{-m+1} + \frac{b_{-m}}{2};$$

$$b_{-m+1} = [x_{-m+1}];$$

$$x_{-m} = \{x_{-m+1}\} \cdot 2 = \left(= \frac{b_{-m}}{2^2} \right) \cdot 2 = b_{-m};$$

$$b_{-m} = [x_{-m}].$$

Вычисления должны продолжаться до тех пор, пока дробная часть очередного произведения не превратится в ноль. Однако могут быть дроби, представление которых в новой системе счисления будет бесконечным.

Разберем пример применения этого алгоритма на конкретной правильной десятичной дроби $0, A_{10}$=0,125.

№	$0, A_{10}$	Дробная часть x_i	Целая часть b_i
–1	0,125	$x_{-1} = 0{,}125 \cdot 2 = 0{,}25$	$b_{-1} = [x_{-1}] = [0{,}25] = 0$
–2		$x_{-2} = \{x_{-1}\} \cdot 2 = 0{,}25 \cdot 2 = 0{,}5$	$b_{-2} = [x_{-2}] = [0{,}5] = 0$
–3		$x_{-3} = \{x_{-2}\} \cdot 2 = 0{,}5 \cdot 2 = 1$	$b_{-3} = [x_{-3}] = [1] = 1$

Итак, $0{,}125_{10} = 0{,}001_2$.

Сформулируем этот алгоритм в общем виде.

Алгоритм перевода дроби из десятичной системы счисления в любую другую. *Для того чтобы исходную правильную десятичную дробь* $0,A$ *заменить равной ей правильной дробью* $0,B_p$*, нужно* $0,A$ *умножить на новое основание p. Целую часть полученного произведения считать цифрой старшего разряда искомой дроби. Дробную часть полученного произведения вновь умножить на p, целую часть полученного результата считать следующей цифрой искомой дроби. Эти операции продолжать до тех пор, пока дробная часть не окажется равной нулю, или не будет найден период, либо не будет достигнута требуемая точность.*

Разберем примеры перевода десятичного числа 0,375 в двоичную, восьмеричную и шестнадцатеричную системы счисления (жирным шрифтом выделены цифры числа в новой системе счисления):

Двоичная система счисления	Восьмеричная система счисления	Шестнадцатеричная система счисления
$0,375\cdot 2 = \mathbf{0},75$ $0,75\cdot 2 = \mathbf{1},5$ $0,5\cdot 2 = \mathbf{1},0$ $0,375_{10} = 0,011_2$	$0,375\cdot 8 = \mathbf{3},0$ $0,375_{10} = 0,3_8$	$0,375\cdot 16 = \mathbf{6},0$ $0,375_{10} = 0,6_{16}$

В приведенном выше случае дроби, полученные в результате перевода из одной системы счисления в другую, оказываются конечными, но возможны случаи, когда конечная дробь при переводе в другую систему счисления окажется бесконечной. Один из таких примеров, полученных при переводе десятичного числа 0,3 в двоичную систему счисления, приведен в таблице ниже:

$0,3\cdot 2 = 0,6$
$0,6\cdot 2 = 1,2$
$0,2\cdot 2 = 0,4$
$0,4\cdot 2 = 0,8$
$0,8\cdot 2 = 1,6$
$0,6\cdot 2 = 1,2$
$0,2\cdot 2 = 0,4$
$0,4\cdot 2 = 0,8$
$0,8\cdot 2 = 1,6$, и т. д.

В этом случае необходимо найти повторяющиеся группы цифр и выделить период: $0,3_{10} = 0,0(1001)_2$ или ограничиться наперед заданной точностью.

При переводе смешанных чисел из десятичной системы счисления в любую другую необходимо для целой части исходного числа использовать **Алгоритм перевода целого числа из десятичной системы**

счисления в любую другую, а для дробной части – **Алгоритм перевода дроби из десятичной системы счисления в любую другую**. Затем полученные результаты сложить.

Обратный перевод – из любой системы счисления в десятичную – мы уже, по сути дела, проводили. Для этого достаточно просто перевести число в многочленную форму и вычислить этот многочлен по правилам десятичной арифметики.

$$(\overline{a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m}})_q = \left(a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + ... + a_1 \cdot q + a_0 + \frac{a_{-1}}{q} + ... + \frac{a_{-m}}{q^m} \right)_{10}$$

Например,

$$1101111,011_2 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 111,375_{10};$$

$$7310,24_8 = 7 \cdot 8^3 + 3 \cdot 8^2 + 1 \cdot 8^1 + 0 \cdot 8^0 + 2 \cdot 8^{-1} + 4 \cdot 8^{-2} = 3784,5625_{10}.$$

Заметьте, что этот способ перевода работает и для целых чисел, и для дробных. Следовательно, общий алгоритм перевода из любой позиционной системы счисления в десятичную может звучать следующим образом:

Алгоритм перевода из любой системы счисления в десятичную. *Для того чтобы исходное число A_q заменить равным ему десятичным числом B, достаточно записать исходное число A_q в многочленной форме по правилам десятичной арифметики, а затем вычислить полученный многочлен.*

Алгоритм перевода чисел из одной системы счисления в другую можно сделать более удобным для вычисления на компьютере. Для вычисления целой части можно применить схему Горнера. В вычислительной математике она используется для оптимизации вычисления полиномов. В ней используется минимальное число арифметических операций, и поэтому она считается оптимальной для программирования. В этом случае следует разделять перевод целых и дробных чисел.

$$(\overline{a_n a_{n-1}...a_1 a_0})_q = \left(a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + ... + a_1 \cdot q + a_0 \right)_{10}.$$

Вынесем за скобки q из суммы n старших слагаемых.

$$(\overline{a_n a_{n-1}...a_1 a_0})_q = \left((a_n \cdot q^{n-1} + a_{n-1} \cdot q^{n-2} + ... + a_1)q + a_0 \right)_{10}.$$

Далее – вынесем за скобки q из суммы n–1 старших слагаемых и т. д., пока не дойдем до последнего слагаемого.

$$(\overline{a_n a_{n-1}...a_1 a_0})_q = (...((a_n \cdot q + a_{n-1}) \cdot q + a_{n-1}) \cdot q + ...a_1) \cdot q + a_0.$$

Приведем пример для шестнадцатеричной системы счисления:

$$F8A_{16} = ((15 \cdot 16 + 8) \cdot 16 + 10)_{10} = 3978_{10}.$$

Для восьмеричной системы счисления:

$$4302_5 = (((4 \cdot 5 + 3) \cdot 5 + 0) \cdot 5 + 2)_{10} = 577_{10}.$$

Для двоичной системы счисления:

$$11011101_2 = (((((((1 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1)_{10} = 221_{10}.$$

Запишем этот алгоритм в общем случае.

Алгоритм перевода целого числа из любой системы счисления в десятичную. *Для того чтобы исходное целое число A_q заменить равным ему целым десятичным числом B, достаточно цифру старшего разряда числа A_q умножить по правилам десятичной арифметики на старое основание q. К полученному произведению прибавить цифру следующего разряда числа A_q. Полученную сумму вновь умножить на q, вновь к полученному произведению прибавить цифру следующего* (*более младшего*) *разряда. Так поступают до тех пор, пока не будет прибавлена младшая цифра числа A_q. Полученное число и будет искомым числом десятичным B.*

Подобным образом можно облегчить вычисление дробной части многочленной формы представления числа.

$$(\overline{0,a_{-1}...a_{-m}})_q = \left(\frac{a_{-1}}{q}+...+\frac{a_{-m}}{q^m}\right)_{10} = (...((a_{-m}:q+a_{-m+1}):q+a_{-m+2}):q+...+a_{-1}):q\ .$$

Например,

$$0{,}1101_2 = (((1:2+0):2+1):2+1):2 = 0{,}8125_{10}$$

Запишем этот алгоритм в общем виде.

Алгоритм перевода дробного числа из любой системы счисления в десятичную. *Для того чтобы исходную правильную дробь $0,A_q$ заменить равной ей правильной десятичной дробью $0,B$, нужно цифру младшего разряда дроби $0,A_q$ разделить на старое основание q по правилам десятичной арифметики. К полученному частному прибавить цифру следующего (более старшего) разряда и далее поступать так же, как и с первой цифрой. Эти операции продолжать до тех пор, пока не будет прибавлена цифра старшего разряда исходной дроби. После этого полученную сумму разделить еще раз на q.*

Перевод из восьмеричной системы счисления в десятичную:

$$0{,}45_8 = (5:8+4):8 = 0{,}578125_{10}\ .$$

Перевод из шестнадцатеричной системы счисления в десятичную:

$$0{,}F03_{16} = ((3:16+0):16+15):16 = 0{,}9382324_{10}\ .$$

Описанные алгоритмы допускают обобщение на перевод из системы счисления произвольным основанием q в систему счисления с произвольным основанием p. Для целых чисел их можно записать следующим образом.

Алгоритм 1. Для того чтобы исходное целое число A_q заменить равным ему целым числом B_p, необходимо число A_q разделить нацело по правилам q-арифметики на новое основание p, записанное в системе счисления с основанием q. Полученный результат вновь разделить нацело на основание p и т. д. до тех пор, пока целая часть не превратится в ноль. Цифрами искомого числа B_p являются остатки от деления в системе счисления с основанием p, выписанные так, чтобы последний остаток являлся бы цифрой старшего разряда числа B_p.

Алгоритм 2. Для того чтобы исходное целое число A_q заменить равным ему целым числом B_p, достаточно цифру старшего разряда числа A_q умножить по правилам p-арифметики на старое основание q. К полученному произведению прибавить цифру следующего разряда числа A_q. Полученную сумму вновь умножить на q по правилам p-арифметики, вновь к полученному произведению прибавить цифру следующего (более младшего) разряда. Так поступают до тех пор, пока не будет прибавлена младшая цифра числа A_q. Полученное число и будет искомым числом B_p.

Сложность в применении обобщенных алгоритмов состоит в том, что мы привыкли делать расчеты в десятичной системе счисления, а не в произвольной.

Перевод дробных чисел из системы с основанием p в систему с основанием q выполняется по следующим правилам:

Алгоритм 3. Для того чтобы исходную правильную дробь $0,A_q$ заменить равной ей правильной дробью $0,B_p$, нужно $0,A_q$ умножить на новое основание p по правилам q-арифметики. Целую часть полученного произведения считать цифрой старшего разряда искомой дроби. Дробную часть полученного произведения вновь умножить на p, целую часть полученного результата считать следующей цифрой искомой дроби. Эти операции продолжать до тех пор, пока дробная часть не окажется равной нулю, либо не будет достигнута требуемая точность.

Алгоритм 4. Для того чтобы исходную правильную дробь $0,A_q$ заменить равной ей правильной дробью $0,B_p$, нужно цифру младшего разряда дроби $0,A_q$ разделить на старое основание q по правилам p-арифметики. К полученному частному прибавить цифру следующего (более старшего) разряда и далее поступать так же, как и с первой цифрой. Эти операции продолжать до тех пор, пока не будет прибавлена цифра старшего разряда исходной дроби. После этого полученную сумму разделить еще раз на q.

Выбор оптимальной системы счисления

Задумается над вопросом: какая из возможных позиционных систем счисления наиболее оптимальна для ручных вычислений? А для машинных вычислений?

С устными (ручными) вычислениями вроде бы все ясно. Мы с детства изучаем десятичную систему счисления. Начало использования системы счисления с основанием 10 очевидно – счет с помощью пальцев. Так что десятичный счет – это традиция, заложенная тысячелетиями.

Но удобно ли использовать десятичную систему счисления в машинных вычислениях? Один из самых важных критериев – объем памяти, которая хранит числа представленные в той или иной системе счисления. Другой критерий – величина самого большого числа, которое может быть представлено в этом объеме памяти.

Введем понятие экономичности представления числа в данной системе счисления.

Под *экономичностью системы счисления* будем понимать то количество чисел, которое можно записать в данной системе с помощью определенного количества цифр.

Речь в данном случае идет не о количестве разрядов, а об общем количестве сочетаний цифр, которые интерпретируются как различные числа.

Например, чтобы написать 1000 чисел (от 000 до 999) в десятичной системе счисления нам нужно 30 цифр (от 0 до 9 на каждый из трех разрядов). А в двоичной системе с помощью 30 цифр мы можем составить 2^{15} различных чисел. Количество разрядов в числе $\frac{30}{2}=15$, количество цифр – две (0 и 1). $2^{15}=32768>1000$. Поэтому двоичная система счисления экономичнее десятичной.

Найдем самую экономичную систему счисления.

Обозначим n – количество цифр, с помощью которых записываются числа, а x – основание системы счисления. Тогда количество разрядов в числе, записанных в этой системе счисления, равняется $\frac{n}{x}$. Отсюда количество чисел, которые можно записать, равно $x^{\frac{n}{x}}$. Найдем, при каком x это выражение принимает максимальное значение.

Пусть $n=24$. В табл. 3.3 приведены расчеты экономичности систем счисления с различными основаниями.

Таблица 3.3

Основание системы счисления	**Количество чисел, записанных с помощью 24 цифр**
$x=2$	$x^{\frac{24}{x}}=2^{12}=4096$
$x=3$	$x^{\frac{24}{x}}=3^{8}=6561$
$x=4$	$x^{\frac{24}{x}}=4^{6}=4096$
$x=6$	$x^{\frac{24}{x}}=6^{4}=1296$
$x=8$	$x^{\frac{24}{x}}=8^{3}=512$

Можно сделать вывод, что основание самой экономичной системы счисления лежит в диапазоне от 2 до 4.

Найдем точное значение. Для этого необходимо найти максимум функции: $f(x)=x^{\frac{n}{x}}=\left(x^{\frac{1}{x}}\right)^{n}$. В математике экстремумы функций находят, определяя производную этой функции (экстремумы находятся в тех

точках, где производная равна нулю). Определим производную функции *f*.

$$f'(x) = n \cdot \left(x^{\frac{1}{x}} \right)^{n-1} \cdot \frac{dx^{\frac{1}{x}}}{dx}.$$

Представим $x^{\frac{1}{x}}$ в следующем виде:

$$x^{\frac{1}{x}} = \exp\left(\ln\left(x^{\frac{1}{x}} \right) \right) = \exp\left(\frac{1}{x} \cdot \ln x \right).$$

Отсюда:

$$\frac{dx^{\frac{1}{x}}}{dx} = \frac{d \exp\left(\frac{1}{x} \cdot \ln x \right)}{dx} = \exp\left(\frac{1}{x} \cdot \ln x \right) \cdot \frac{d\left(\frac{1}{x} \cdot \ln x \right)}{dx} = x^{\frac{1}{x}} \cdot \left(-\frac{1}{x^2} \ln x + \frac{1}{x^2} \right).$$

Итак, производная имеет вид:

$$f'(x) = n \cdot \left(x^{\frac{1}{x}} \right)^n \cdot x^{\frac{1}{x}} \cdot x^{-2} \cdot (1 - \ln x) = n \cdot x^{\frac{n}{x}-2} \cdot (1 - \ln x)$$

Приравняем полученную производную нулю.

$f'(x) = 0$, отсюда $\ln x = 1$, $x = e = 2{,}71828...$

Системы счисления с иррациональными основаниями также существуют и активно исследуются математиками. Однако более простой для реализации являются устройства, использующие для вычислений целые основания. Ближайшее целое число – три, и поэтому троичную систему счисления считают наиболее экономичной.

В 1960-х годах в нашей стране была построена вычислительная машина «Сетунь», которая работала в троичной системе счисления. Предпочтение все же отдается двоичной системе счисления, поскольку по экономичности она оказывается второй за троичной, а технически реализуется гораздо проще остальных. Все современные цифровые компьютеры используют двоичную систему счисления.

Взаимосвязь между системами счисления с основаниями «2», «8» и «16»

Интерес к двоичной системе счисления вызван тем, что именно эта система используется для представления чисел в компьютере. Однако двоичная запись оказывается громоздкой, поскольку содержит много цифр, и, кроме того, она плохо воспринимается и запоминается человеком из-за зрительной однородности (все число состоит из нулей и единиц). Поэтому в нумерации ячеек памяти компьютера, записи кодов команд, нумерации регистров и устройств и пр. используются системы счисления с основаниями 8 и 16; выбор именно этих систем счисления обусловлен тем, что переход от них к двоичной системе и обратно осуществляется, как будет показано ниже, весьма простым образом.

Алгоритм. Для записи двоичного числа в системе с основанием $q = 2^n$ достаточно данное двоичное число разбить на группы цифр от запятой по *n* цифр в каждой группе. Затем каждую группу цифр следует рассматривать как *n*-разрядное двоичное число и записать его как цифру в системе с основанием $q = 2^n$.

Рассмотрим подробнее механизм работы этого алгоритма.

Пусть нам дано число $A_2 = 10010111101{,}11011_2$, которое нужно перевести в шестнадцатеричную систему счисления. Вспомним, что $16 = 2^4$. Представим это число в многочленной форме записи числа.

$$A_2 = 10010111101{,}11011_2 =$$
$$= 1\cdot 2^{10} + 0\cdot 2^9 + 0\cdot 2^8 + 1\cdot 2^7 + 0\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 +$$
$$+ \frac{1}{2^1} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{1}{2^4} + \frac{1}{2^5}$$

Сгруппируем слагаемые данного многочлена по степеням двойки: от –5 до –8, от –1 до –4, от 0 до 3, от 4 до 7, от 8 до 11.

$$A_2 = \left(0\cdot 2^{11} + 1\cdot 2^{10} + 0\cdot 2^9 + 0\cdot 2^8\right) + \left(1\cdot 2^7 + 0\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^4\right) +$$
$$+ \left(1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right) + \left(\frac{1}{2^1} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{1}{2^4}\right) + \left(\frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{0}{2^8}\right)$$

Вынесем за скобки множители вида 2^i, где *i* кратно 4.

$$A_2 = \left(0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0\right)\cdot 2^8 + \left(1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0\right)\cdot 2^4 +$$
$$+ \left(1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)\cdot 2^0 + \left(1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)\cdot 2^{-4} +$$
$$+ \left(1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0\right)\cdot 2^{-8} =$$
$$= \left(0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0\right)\cdot 16^2 + \left(1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0\right)\cdot 16^1 +$$
$$+ \left(1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)\cdot 16^0 + \left(1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)\cdot 16^{-1} +$$
$$+ \left(1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0\right)\cdot 16^{-2} =$$
$$= b_2\cdot 16^2 + b_1\cdot 16^1 + b_0\cdot 16^0 + b_{-1}\cdot 16^{-1} + b_{-2}\cdot 16^{-2}$$

Мы получили многочленную форму представления числа в шестнадцатеричной системе счисления, где b_i – четырехзначные двоичные числа.

$$b_2 = 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 0100_2 = 4_{10} = 4_{16}$$
$$b_1 = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 1011_2 = 11_{10} = B_{16}$$
$$b_0 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 1101_2 = 13_{10} = D_{16}$$
$$b_{-1} = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 1101_2 = 13_{10} = D_{16}$$
$$b_{-2} = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 1000_2 = 8_{10} = 8_{16}$$

Отсюда получаем:

$$A_2 = \left(4\cdot 16^2 + 11\cdot 16^1 + 13\cdot 16^0 + 13\cdot 16^{-1} + 8\cdot 16^{-2}\right)_{10} =$$
$$= \left(4\cdot 10^2 + B\cdot 10^1 + D\cdot 10^0 + D\cdot 10^{-1} + 8\cdot 10^{-2}\right)_{16} = 4BD, D8_{16}$$

Итак, $10010111101{,}11011_2 = 4BD, D8_{16}$

Для того чтобы быстро переводить числа из двоичной системы счисления в системы счисления с основанием 2^n, нужно запомнить соответствие цифр. Одной цифре в системе счисления с основанием 4 будет соответствовать двузначное двоичное число, так как $4 = 2^2$. Одной цифре в системе счисления с основанием 8 будет соответствовать трехзначное двоичное число, так как $8 = 2^3$. Одной цифре в системе счисления с основанием 16 будет соответствовать четырехзначное двоичное число, так как $16 = 2^4$. Числа двоичной системы счисления и соответствующие цифры четверичной, восьмеричной и шестнадцатеричной приведены в табл. 3.4.

Таблица 3.4. Соответствие цифр систем счисления с основаниями 2^n

2 с.с.	**4 с.с.**	**2 с.с.**	**8 с.с.**	**2 с.с.**	**16 с.с.**
00	0	000	0	0000	0
01	1	001	1	0001	1
10	2	010	2	0010	2
11	3	011	3	0011	3
		100	4	0100	4
		101	5	0101	5
		110	6	0110	6
		111	7	0111	7
				1000	8
				1001	9
				1010	A
				1011	B
				1100	C
				1101	D
				1110	E
				1111	F

Итак, например:

$$11101101{,}11101_{2\to 8} = \underbrace{11}_{7}\underbrace{101}_{3}\underbrace{101}_{5},\underbrace{111}_{7}\underbrace{010}_{2} = 735{,}72_8 .$$

В этом примере мы столкнулись с ситуацией, когда после запятой у двоичного числа количество цифр не кратно 3. В этом случае необходимо дописать столько нулей, сколько не хватает. У целых чисел незначащие нули дописываются левее самого старшего разряда, у дробных – правее самого младшего разряда.

$$11101101{,}11101_{2\to 16} = \underbrace{0001}_{1}\underbrace{1101}_{D}\underbrace{1101}_{D},\underbrace{1110}_{E}\underbrace{1000}_{8} = 1DD,E8_{16} .$$

Аналогично при переводе числа из системы счисления с основанием $p = 2^n$, необходимо записать цифры двоичного числа в соответствии с табл. 3.4.

Алгоритм. Для замены числа, записанного в системе счисления с основанием $p = 2^n$, равным ему числом в двоичной системе счисления, достаточно каждую цифру данного числа заменить *n*-разрядным двоичным числом.

Приведем пример для четверичной системы счисления:

$$3021{,}322_{4\to 2} = \underset{11001001}{3}\underset{}{0}\underset{111010}{2}\underset{}{1},\underset{}{3}\underset{}{2}\underset{}{2} = 11001001{,}11101_2 .$$

Обратите внимание, что запятая, отделяющая дробную часть от целой, остается на месте.

Для шестнадцатеричной системы счисления:

$$F093, A8_{16\to 2} = \underset{1111}{F}\ \underset{0000}{0}\ \underset{1001}{9}\ \underset{0011}{3}\ ,\ \underset{1010}{A}\ \underset{1000}{8} = 1111000010\ 010011{,}10101_2 .$$

Двоичная арифметика

Двоичная система счисления является минимальной системой, в которой реализуется *принцип позиционности* в цифровой форме записи числа. В двоичной системе счисления вес каждой цифры при переходе от любого данного разряда к следующему, более старшему разряду, увеличивается вдвое.

Утверждение двоичной арифметики в качестве общепринятой основы при конструировании ЭВМ с программным управлением состоялось под влиянием работы А. Беркса, Х. Гольдстайна и Дж. фон Неймана над проектом первой ЭВМ с хранимой в памяти программой.

Арифметика двоичной системы счисления, как и всякой другой позиционной системы, основывается на использовании таблиц сложения и умножения цифр.

Таблица сложения двоичной системы счисления:

+	0	1
0	0	1
1	1	10

Таблица умножения двоичной системы счисления:

1	0	1
0	0	0
1	0	1

Приведем пример сложения двух многозначных двоичных чисел. Сложение проводится «столбиком», как и в десятичной системе счисления. Выравниваем два числа по запятой, а затем складываем соответствующие разряды этих чисел. При сложении двух единиц в соответствующем разряде суммы записываем ноль и единицу переносим в соседний старший разряд. Например:

$$\begin{array}{r} \dot{1}1\dot{1}0\dot{1}1,1101 \\ +\ \ 10001,001 \\ \hline 1001100,1111 \end{array}\quad ; \qquad \begin{array}{r} \dot{1}\dot{1}\dot{1}0\dot{1}\dot{1},\dot{0}\dot{0}1 \\ +\ \dot{1}101011,11101 \\ \hline 10100111,00001 \end{array}$$

Точками показано, что необходимо учесть единицу из предыдущего разряда.

Умножение двоичных чисел также проводится столбиком:

$$
\begin{array}{r}
\times 101110 \\
1101 \\
\hline
101110 \\
0000000 \\
10111000 \\
101110000 \\
\hline
1001010110
\end{array}
$$

При этом следует обратить внимание на то, что таблица двоичных чисел умножения весьма проста, и умножение сложных чисел проводится так: если младший разряд множителя равен 1, то множимое просто переписывается, если равен 0, то записываются нули. Затем проверяется более старший разряд. Если он равен 1, то множимое сдвигается на один разряд и записывается в столбик, а если равен 0, то записывают нули, и т. д. Затем получившиеся числа нужно просто сложить.

Таким образом, видно, что алгоритм сложения и умножения весьма прост и сводится к операциям сложения и сдвига. Это было также одним из аргументов при разработке первых компьютеров, ведь и оборудование, предназначенное для автоматических вычислений, также сделать значительно проще.

Другие системы счисления

Описанные системы счисления являются простейшими и наиболее распространенными. Тем не менее следует отметить, что основание системы счисления не всегда совпадает с количеством цифр и может быть отрицательным, иррациональным и даже мнимым.

Так называемые «симметричные системы счисления» являются дальнейшим развитием идеи позиционного представления чисел. Основная особенность таких систем состоит в использовании отрицательных и положительных цифр для представления чисел. Симметричная система счисления может иметь нечетное целое основание. Например, в симметричной пятеричной системе счисления определены цифры –2, –1, 0, 1, 2; в девятеричной – цифры –4, –3, –2, –1, 0, 1, 2, 3, 4. Наличие нуля здесь принципиально, именно поэтому если основание системы счисления четно, то на ее основе нельзя построить симметричную систему.

Простейшей из симметричных систем счисления является «троичная симметричная система счисления». Еще ее называют уравновешенной троичной системой счисления. В этой системе счисления в качестве основания используется число 3, а в качестве цифр – троичные цифры 1, 0 и –1. В этом состоит отличие уравновешенной троичной системы счисления от обычной (в обычной троичной системе счисления в качестве цифр используются 0, 1, 2).

Позиционная симметричная (уравновешенная) троичная система счисления была предложена математиком Леонардо Пизано Фибоначчи (1170–1228) для решения «задачи о гирях». В задаче шла речь о бедном торговце, который с помощью четырех камней на рычажных чашечных весах совершенно правильно взвешивал предметы массой от 1 до 40 кг. Для этого он использовал камни весом 1, 3, 9 и 27 кг.

Пусть груз, который надо взвесить, весит A кг. Это число можно представить в троичной системе:

$$A = (a_n a_{n-1} \ldots a_1 a_0)_3 = a_n \cdot 3^n + a_{n-1} \cdot 3^{n-1} + \ldots + a_1 \cdot 3 + a_0,$$

где коэффициенты $a_0, a_1, \ldots, a_n$ могут принимать значения 0, 1 или 2.

Очевидно, что $2 \cdot 3^i = (3-1) \cdot 3^i = 3 \cdot 3^i - 1 \cdot 3^i = 3^{i+1} - 3^i$.

Введем «отрицательную цифру «–1» и обозначим ее $\overline{1}$. Тогда последнее равенство можно записать: $2 \cdot 3^i = 3^{i+1} - 3^i = 3^{i+1} + \overline{1} \cdot 3^i$.

Следовательно, любое целое число можно записать в троичной уравновешенной системе счисления с помощью цифр 0, 1 и $\overline{1}$, заменив в многочленной форме представления числа цифру 2 на соответствующую разность.

$$A = (b_n b_{n-1} \ldots b_1 b_0)_3 = b_n \cdot 3^n + b_{n-1} \cdot 3^{n-1} + \ldots + b_1 \cdot 3 + b_0,$$

где $b_0, b_1, \ldots, b_n$ могут принимать значения 0, 1 или $\overline{1}$.

Например, представим число 100 в несимметричной и симметричной системе счисления:

$$100_{10} = 10201_3 = 1 \cdot 3^4 + 0 \cdot 3^3 + 2 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0.$$

Поскольку в симметричной системе счисления не может быть цифры 2, это выражение необходимо преобразовать к другому виду, в котором присутствуют только цифры 0, 1 и –1. Заметим, что выражение вида $2 \cdot 3^n$ может быть преобразовано следующим образом:

$$2 \cdot 3^n = 1 \cdot 3^{n+1} + (-1) \cdot 3^n.$$

Это соотношение определяет правило преобразования цифры 2 при переводе из обычной троичной системы счисления в уравновешенную: в текущем, n-м разряде необходимо вместо 2 поставить цифру $\overline{1}$ и добавить 1 к n+1-му разряду.

Проведем эти преобразования для числа 100_{10}.

$$100_{10} = 10201_3 = 1 \cdot 3^4 + 0 \cdot 3^3 + 2 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 1 \cdot 3^4 + 1 \cdot 3^3 + (-1) \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 =$$
$$= 1 \cdot 3^4 + 1 \cdot 3^3 - 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 1 \cdot 3^4 + 1 \cdot 3^3 + (-1) \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 11\,\overline{1}01_3.$$

Итак, чтобы уравновесить груз в $A = (b_n b_{n-1} \ldots b_1 b_0)_3$ кг на чашечных весах, нужно положить его на первую чашу весов, а гирю в 1 кг поставить:

– на вторую чашу, если $b_0 = 1$,

– или на первую чашу весов, если $b_0 = \overline{1}$,

– или не использовать эту гирю, если $b_0 = 0$.

Далее, гиря весом в 3 кг ставится:

– на вторую чашу, если $b_0 = 1$,

– или на первую чашу весов, если $b_0 = \overline{1}$,

– или не использовать эту гирю, если $b_0 = 0$.

Расставив гири по такому принципу, можно уравновесить любой груз. Если величина груза не была известна, то мы подбираем такое расположение гирь на весах, которое уравновешивает этот груз, и тем самым определяем и массу груза.

Представление отрицательных чисел в троичной уравновешенной системе счисления

Наличие положительной и отрицательной цифр позволяет непосредственно представлять как положительные, так и отрицательные числа (не нужно вводить специальный символ для знака числа). При этом нет необходимости вводить дополнительный код для выполнения арифметических операций (см. лекцию о машинном представлении числовых данных). Знак числа в троичной уравновешенной системе счисления определяется знаком старшей цифры числа: если она положительная, то и число положительно; если отрицательная, то и число отрицательно. Например:

$79_3 = 100\bar{1}1_3 = 3^4 - 3^1 + 1 = 81 - 3 + 1$ – число положительное;

$-79_3 = \bar{1}001\bar{1}_3 = -81 + 3 - 1$ – число отрицательное.

Очевидно, что для изменения знака числа надо изменить знаки всех его цифр (то есть инвертировать его код). Например:

$$10\bar{1} = 8; \quad \bar{1}01 = -8$$

Можно сделать вывод, что при вычислениях на основе троичной уравновешенной системы отпадает необходимость в операции «вычитания».

Операция сложения всякой цифры в этой системе с нулем дает в результате эту же цифру. Сложение 1 с $\bar{1}$ дает ноль. И только сумма двух единиц или двух $\bar{1}$ формируется путем переноса в следующий разряд цифры того же знака, что и слагаемые, и установки в текущем разряде цифры противоположного знака. Пример:

$$204 + 79 = 283$$

$$204_{10} = 10\bar{1}\bar{1}\bar{1}0_3$$

$$79_{10} = 100\bar{1}1_3$$

Проведем процедуру сложения столбиком в троичной уравновешенной системе счисления:

$$\begin{array}{cccccc} 1 & 0 & \bar{1} & \bar{1} & \bar{1} & 0 \\ + & 1 & 0 & 0 & \bar{1} & 1 \\ \hline 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

$$101111_3 = 3^5 + 3^3 + 3^2 + 3^1 + 3^0 = 283_{10}$$

Приведем пример сложения чисел с разными знаками: –204 + 79.

В троичной уравновешенной системе счисления эти числа выглядят следующим образом:

$$-204_{10} = \bar{1}01110$$
$$79_{10} = 100\bar{1}1$$

Сумма в десятичной системе счисления $-204 + 79 = -125$.

Сложим столбиком эти два числа в троичной уравновешенной системе счисления:

$$\begin{array}{cccccc} \bar{1} & 0 & 1 & 1 & 1 & 0 \\ + & 1 & 0 & 0 & \bar{1} & 1 \\ \hline \bar{1} & 1 & 1 & 1 & 0 & 1 \end{array}$$

Для проверки переведем получившееся число в десятичную систему счисления:

$$\bar{1}\,11101_3 = -3^5 + 3^4 + 3^3 + 3^2 + 3^0 = -125_{10}$$

Перевод целых десятичных чисел в троичную уравновешенную систему счисления

Для перевода из десятичной системы в троичную уравновешенную, можно воспользоваться следующим алгоритмом:

1. Выполняем деление исходного числа на 3 в десятичной системе счисления.
2. Если остаток от деления равен 2, записываем его как $\bar{1}$, а к результату от деления добавляем 1; если результат меньше 2 (т. е. 1 или 0), оставляем его без изменений.
3. Делим результат на 3 и повторяем действия, описанные в п. 2; деление продолжается до тех пор, пока целая часть не будет равна 0.
4. Переписываем полученные значения остатков (снизу вверх), начиная с последнего результата деления.

Пример

Переведем по предложенному правилу число 158_{10} в троичную уравновешенную систему счисления.

Результаты перевода сведены в таблицу:

Частное	Остаток	Цифра	Измененное частное
$\left[\frac{158}{3}\right] = 52$	2	$\bar{1}$	52 + 1 = 53
$\left[\frac{53}{3}\right] = 17$	2	$\bar{1}$	17 + 1 = 18
$\left[\frac{18}{3}\right] = 6$	0	0	
$\left[\frac{6}{3}\right] = 2$	0	0	
$\left[\frac{2}{3}\right] = 0$	2	$\bar{1}$	0 + 1 = 1
$\left[\frac{1}{3}\right] = 0$	1	1	

Итак, $158_{10} = 1\,\bar{1}\,00\,\bar{1}\,\bar{1}_3$

Второй способ перевода в троичную уравновешенную систему предложен известным программистом Дональдом Кнутом. Его суть заключается в следующем. Сначала записываем произвольное число $A = 208{,}3$ в обычной троичной системе счисления.

$$208{,}3_{10} = 21201{,}022002200220\ldots_3 .$$

Очевидно, что если к этому числу добавить другое число B и затем его же вычесть, то в результате получится то же самое число A. Возьмем в качестве B бесконечное число, состоящее из одних единиц: $B = \ldots 11111111{,}11111111\ldots$; в результате получим:

$$A + B = (\ldots 111210012{,}210121012101\ldots)_3 .$$

Теперь вычтем из суммы число B, причем при вычитании 1 из 2 получаем в результате 1, при вычитании 1 из 1 получаем 0, а при вычитании 1 из 0 получаем в результате $\bar{1}$.

$$208{,}3_{10} = (10\bar{1}\bar{1}01{,}10\bar{1}010\bar{1}010\bar{1}0\ldots)_3 .$$

Таким образом, уравновешенная троичная система счисления обладает многими привлекательными свойствами, в частности:

- отрицание числа осуществляется взаимной заменой 1 и $\bar{1}$;
- знак числа задается его наиболее значимым ненулевым тритом;
- операция округления до ближайшего целого идентична усечению, то есть при округлении отбрасывается все, что стоит правее разделяющей точки.
- естественное представление отрицательных и положительных чисел позволяет не использовать дополнительный код числа, как это было сделано для двоичных чисел, и, следовательно, упростить процедуру арифметических вычислений.

Эти важные свойства позволяют считать, что троичная уравновешенная система счисления является хорошим кандидатом для реализации вычислительной системы на ее основе.

Контрольные вопросы и задания

1. Дайте определение системы счисления с натуральным основанием.
2. Дайте определение базиса и основания системы счисления.
3. Назовите преимущества троичной уравновешенной системы счисления.
4. Переведите целые числа из десятичной системы счисления в двоичную, шестнадцатеричную, восьмеричную и троичную уравновешенную системы счисления:

214 278 263 257 333 307 314 299 317 353 269 218 377 443 391 490 241 365 525 188 292 159.

5. Переведите дробные числа из десятичной системы счисления в двоичную, шестнадцатеричную и восьмеричную системы счисления:

Часть 1

0,625	0,6875	0,75	0,8125	0,875	0,9375	0,0625
0,125	0,1875	0,25	0,3125	0,375	0,4375	0,5
0,5625	0,0859375	0,1953125	0,390625			

Часть 2

397,25	205,7	184,8	163,15	217,05	425,16	208,65
501,7	532,1	492,9	391,15	601,05	555,09	337,55
345,35	449,45	700,65	999,75	900,85	304,75	473,6
298,8	285,7	402,5	701,25	613,4	583,3	414,375
170,625	119,75	227,125				

6. Переведите числа из восьмеричной системы счисления в двоичную и шестнадцатеричную с использованием правил косвенного перевода:

764,26	532,47	374,062	405,73	271,502	674,55	173,21
247,17	777,1	514,04	105,25	333,33	5015,634	710,16
3504,144	250,456	3161,176	2241,002	2674,74	1042,7	1112,5
3660,25	333,5	421,002				

7. Проведите следующие арифметические вычисления «столбиком» в двоичной системе счисления.

   11001011+11001
   10101011+11011001
   10011·10101
   11011·1111

Список литературы к лекции 3

1. *Кнут Д.Э.* Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд.: Пер. с англ.: Уч. пос. - М.: Издательский дом "Вильямс", 2000. - 832 с.: ил.
2. *Андреева Е.*, *Фалина И.* Информатика: Системы счисления и компьютерная арифметика. – М. : Лаборатория Базовых Знаний, 1999. – 256 с. : ил.
3. *Фомин С.В*. Системы счисления. – 4-е изд. – М. : Наука, Гл. ред. физ.-мат. лит. – 48 с.
4. Интернет-ресурс. Режим доступа: http://misterkruger.narod.ru/math/base3rus.htm.

Лекция 4. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В КОМПЬЮТЕРЕ

Так как компьютер может различить только нулевое и единичное состояния бита, то он работает в системе счисления с основанием «2». Вся информация в компьютере – команды и целые программы, аудио- и видеоданные, текстовые файлы, числовые массивы представляются только при помощи нулей и единиц. В современном компьютере минимальной адресуемой единицей информации является байт, представляющий собой 8 битов, или 8-разрядное двоичное число. При этом невозможно различить различные типы данных, если неизвестно заранее, что записано в конкретной ячейке памяти. Биты 01000001 могут представлять как число 65, так и букву «А». Если программа определяет элемент данных для арифметических целей, то 01000001 представляет двоичное число, эквивалентное числу 65. Если программа определяет элемент данных как символ, тогда 01000001 представляет собой букву «А».

Каждый бит является логическим аргументом или логической функцией, и с ними можно проводить логические операции. Основной особенностью представления чисел в памяти компьютера является то, что, в отличие от записи числа на бумаге, компьютерные ячейки имеют ограниченный размер и, следовательно, вынуждают использовать при записи чисел и действиях с ними конечное количество разрядов. Это приводит к тому, что бесконечное множество чисел заменяется конечным множеством их представлений. Способы представления и допустимые над ними действия различны для следующих числовых множеств:

- целые положительные числа (целые числа без знака);
- целые числа со знаком;
- вещественные нормализованные числа.

Рассмотрим подробнее перечисленные группы.

Представление целых чисел

Минимальный объем памяти, выделяемый для хранения целого числа, один байт. Один байт – это восемь бит, тогда количество различных значений, которые можно уместить в этот объем, равняется $2^8 = 256$.

Давайте поразмышляем, какие числа мы поместили бы в один байт памяти компьютера? Рассмотрим следующие варианты.

От 1 до 256. Удобен ли этот диапазон? Даже если нам никогда не понадобятся отрицательные числа, то нулевое значение нам нужно будет всегда. Значит, этот вариант не подходит.

От 0 до 255. Да, такой вариант существует. Например, тип «byte» в Паскале использует этот формат хранения данных. Это представление

называется *беззнаковым*, и целые числа в памяти компьютера хранятся в явном прямом коде. Представление двоичных чисел типа «byte» в памяти компьютера иллюстрируется табл. 4.1.

Таблица 4.1. Представление беззнакового однобайтного числа

Значение	Представление
0	00000000
1	00000001
2	00000010
3	00000011
…	…
255	11111111

Операция сложения беззнаковых чисел происходит по стандартному алгоритму двоичной арифметики.

Сложим двоичные числа 65 и 42.

$$\begin{array}{r} 01000001 \\ +\ 00101010 \\ \hline 01101011 \end{array} \qquad \begin{array}{r} 65 \\ +\ 42 \\ \hline 107 \end{array}$$

Очевиден следующий вопрос: что произойдет, если сумма двух 8-разрядных беззнаковых чисел превысит значение 255? Например, 167 + 198.

$$\begin{array}{r} 10100111 \\ +\ 11000110 \\ \hline \mathbf{1}\,01101101 \end{array} \qquad \begin{array}{r} 167 \\ +\ 198 \\ \hline 365 \end{array}$$

Произошел перенос значащего разряда за разрядную сетку. Следовательно, в разрядной сетке останется число 01101101, которое интерпретируется компьютером как 109. Если в программе, в которой происходит данное действие, контролируются подобные ситуации, то появится сообщение об ошибке. Если такая проверка не производится, то результат вычислений будет неправильным.

Помимо однобайтового представления беззнаковых чисел (byte) в языке программирования Паскаль поддерживается двухбайтовый тип «Word».

Знаковое представление целых чисел. Давайте придумаем способ, с помощью которого компьютер будет отличать положительные числа от отрицательных. Количество различных значений в байте памяти, как говорилось выше, $2^8 = 256$. Получается 128 отрицательных значений и 128 положительных. Естественное предположение, что старший бит (под номером 7) отдается на обозначение знака. Пусть 1 в старшем разряде представляет отрицательное число, а 0 – число положительное. Возможный вариант представления целых чисел со знаком приведен в табл. 4.2.

Таблица 4.2. Один из возможных вариантов представления целых чисел

Значение	Представление
…	…
+2	00000010
+1	00000001
0	00000000
–1	10000001
–2	10000010
…	…

Получаем 127 положительных значений, 127 отрицательных значений и нулевое значение, то есть 255 значений. Потерялось одно значение 10000000.

Если цифровая часть положительных и отрицательных чисел всегда содержит абсолютную величину числа, то такой способ представления знаковых чисел называют *прямым кодом*. Обработка чисел, представленных в прямом коде:

- требует отдельных операций над цифровой и знаковой частями;
- требует альтернативного выполнения операций сложения и вычитания;
- приводит к появлению двух представлений нуля: +0 (00000000) и –0 (10000000).

Наиболее сложной для реализации в данном представлении является операция вычитания.

С этой проблемой столкнулся еще Блез Паскаль в XVII веке при конструировании своей суммирующей машины. При реализации идеи автоматического переноса десятков Паскаля остановила определенная трудность: изобретенный им механизм переноса десятков работал при вращении счетных колес только в одном направлении, а это не позволяло производить вычитание вращением колес в противоположную сторону. Простой и остроумный выход из этого положения, найденный Паскалем, был настолько удачен, что используется в современных ЭВМ. Паскаль *заменил вычитание сложением с дополнением вычитаемого*.

Для 8-разрядной машины Паскаля, работавшей в десятичной системе счисления, дополнением числа A будет $(100000000 - A)$, поэтому операция вычитания $B - A$ может быть заменено сложением $B + (100000000 - A) = 100000000 + (B - A)$. Получившееся число будет больше искомой разности на 100.000.000, но так как машина 8-разрядная, то единица в девятом разряде просто пропадает при переносе десятков из восьмого.

В компьютере происходит все то же, только в двоичной системе счисления.

Дополнением *k*-разрядного целого числа Z в двоичной системе счисления называется величина

$$D(Z,k) = 2^k - Z.$$

Данную формулу можно представить в ином виде:

$$D(Z,k) = ((2^k - 1) - Z) + 1.$$

Число $2^k - 1$ состоит из *k* наибольших в данной системе счисления цифр (если речь идет о 8-разрядном коде, то это 11111111). Поэтому $(2^k - 1) - Z$ можно получить путем дополнения до 1 каждой цифры числа Z и последующим прибавлением к последнему разряду 1. В двоичной системе счисления дополнением 1 является 0, а дополнением 0 является 1. Другими словами, если вычесть столбиком из числа 11111111 произвольное двоичное число *Z*, то в результате получится число, представляющее собой инверсию *Z*. Таким образом, построение $D(Z,k)$ сводится к инверсии данного числа, т. е. замене нулей единицами и единиц нулями, и прибавлением 1 к последнему разряду, и алгоритм записи дополнения двоичного числа сводится к двум этапам:

1. Строится инвертированное представление исходного числа (в каждом бите 0 заменяется на 1, а 1 заменяется на 0). Этот код числа называется обратным кодом.
2. К обратному коду прибавляется 1 по правилам двоичной арифметики.

Следует отметить, что все знаковые целые числа в компьютере представляются в дополнительном коде. Дополнительный код двоичных целых чисел строится по следующим правилам:

- для $Z \geq 0$ дополнительный код совпадает с двоичным представлением числа;
- для $Z < 0$ дополнительный код совпадает с дополнением модуля до числа 2^k, т. е. $D(|Z|,k)$. Здесь *k* – это количество двоичных разрядов в компьютере, отведенных под число данного типа.

Найдем диапазон знаковых целых чисел, занимающих один байт в памяти компьютера. Будем строить таблицу от нуля, затем представление +1 и –1, +2 и –2, и т. д. (см. табл. 4.3).

После построения такой таблицы становится очевидным, что ось симметрии диапазона проходит между 0 и –1. Следовательно, диапазон знаковых чисел, занимающих один байт в памяти компьютера – от –128 до +127. В языке программирования Паскаль этот тип данных обозначается служебным типом ShortInt (короткое целое).

Если число занимает два байта, то диапазоны следующие: –32768... +32767 (Integer – целое) и 0...65535 (Word – слово). Если число занимает четыре байта, то диапазоны следующие: –2147483648… +2147483647 (LongInt – длинное целое) и 0…4294967295 (такого типа в Паскале нет, но есть в других языках программирования).

Таблица 4.3. Представление знаковых однобайтовых чисел в машинном формате

Значение	Представление
+127	01111111
+126	01111110
…	…
+2	00000010
+1	00000001
0	00000000
–1	11111111
–2	11111110
…	…
–126	10000010
–127	10000001
–128	10000000

Анализируя полученную табл. 4.3, можно сделать вывод, что *целые отрицательные двоичные числа содержат единичный бит в старшем разряде.* Эта единица служит только признаком отрицательного числа, поскольку оставшиеся цифры не являются модулем этого отрицательного числа!

Приведем пример представления десятичного числа –65 в дополнительном 8-разрядном коде.

Число 65:	01000001	
Инвертированные биты:	10111110	
Плюс 1:	10111111	равно –65

Теперь рассмотрим, как найти модуль отрицательного числа, представленного в 8-разрядном дополнительном коде. Как было показано выше, дополнительный код отрицательного числа Z есть дополнение для модуля этого числа: $D(|Z|,8) = ((2^8 - 1) - |Z|) + 1$. Перенесем D в правую часть, а $|Z|$ – в левую, получаем: $|Z| = ((2^8 - 1) - D(|Z|,8)) + 1$.

Таким образом, для определения абсолютного значения отрицательного двоичного числа в машинном представлении просто повторяют предыдущие операции: инвертируют все биты и прибавляют единицу.

Для примера найдем абсолютное значение числа –65, дополнительный код которого мы нашли в предыдущем примере:

Двоичное дополнение:	10111111	
Инвертированные биты:	01000000	
Плюс 1:	01000001	равно +65

Проверим, даст ли сумма +65 и –65 в результате 0:

$$\begin{array}{r} 01000001 \\ +\ 10111111 \\ \hline \mathbf{1}\ 00000000 \end{array} \qquad \begin{array}{r} 65 \\ +\ -65 \\ \hline 0 \end{array}$$

Все восемь бит имеют нулевое значение. Единичный бит, вынесенный за разрядную сетку влево, потерян.

Сложение знаковых целых чисел

Рассмотрим в общем случае сложение двух знаковых целых двоичных чисел $C = A + B$. Все возможные варианты представлены в табл. 4.4. Для каждого из этих случаев приведем пример.

Таблица 4.4. Варианты сложения знаковых целых чисел

№ варианта	**Старший бит числа *A***	**Старший бит числа *B***	**Старший бит числа *C***	**Перенос единицы из разрядной сетки**	**Комментарий**
1	0	0	0	Нет	Сложение двух положительных чисел без перехода в знаковый разряд. Знак остается таким же, как у обоих операндов. Результат корректен
2	0	0	1	Нет	Перенос единицы в знаковый разряд при сложении двух положительных чисел. Произошло изменение знака результата (поскольку оба числа положительны, результат должен быть также положительным). Результат некорректен
3	1	1	1	Есть	Сложение двух отрицательных чисел. Знак совпадает со знаками обоих операндов. Несмотря на перенос единицы в 9-й разряд, результат корректен
4	1	1	0	Есть	При сложении двух отрицательных чисел произошло изменение знака результата. Результат некорректен
5	0	1	0 (1)	Есть (Нет)	Сложение чисел с разными знаками; $A > \|B\|$ ($A < \|B\|$). Знак результата равен знаку одного из слагаемых. Вне зависимости от того, произошел ли перенос за

					границы разрядной сетки, результат корректен

Очевидно, что для 8-битного представления диапазон чисел, сумма которых получается без ошибок, меньше, чем для 16-битных. Тем не менее, если понять, почему при сложении возможны неправильные результаты в 8-битном коде, легко можно распространить эти рассуждения и на код другой длины (16, 32 или 64 бит). Поэтому все рассуждения и примеры приведены для 8-битных чисел.

Вариант 1

Очевидно, что при сложении 8-битных чисел переход в знаковый бит не произойдет, пока сумма не превышает 127_{10}. В двоичном представлении $127_{10} = 1111111_2$; т. е. это максимальное число, занимающее 7 бит.

Вариант 2

Очевидно, что когда при сложении двух положительных чисел происходит переход в знаковый разряд, сумма уж точно получится неправильной. Это произойдет в том случае, когда сумма превысит 127.

Найдем результат сложения двух положительных знаковых целых чисел: 102 и 54. Для этого представим их в двоичной системе счисления: $102_{10} = 1100110_2$; $54_{10} = 110110_2$.

Сложим эти два числа столбиком:

$$\begin{array}{r} 01100110 \\ +\ 00110110 \\ \hline 10011100 \end{array} \qquad \begin{array}{r} 102 \\ +\ 54 \\ \hline 156 \end{array}$$

Произошел перенос в знаковый разряд, следовательно, полученная сумма интерпретируется компьютером как отрицательное число, и результат будет некорректным. Действительно, сумма двух положительных чисел всегда должна быть положительна!

Попробуем перевести результат в десятичную систему счисления. Так как тип данных – знаковый, то единица в старшем разряде говорит о том, что число отрицательное. Для того чтобы найти его модуль, надо инвертировать каждый бит и добавить единицу.

Запись результата	10011100	
Инвертированные биты:	01100011	
Плюс 1:	01100100	100_{10}

Следовательно, результат должен интерпретироваться как –100. Это неправильный результат.

Вариант 3

Сложим числа –96 и –23. Сначала найдем их дополнительные коды.

десятичное число	двоичное представление модуля	обратный код	дополнительный код

–96	1100000	10011111	10100000
–23	10111	11101000	11101001

И сложим столбиком эти числа:

Дополнительный 8-разрядный код	Десятичное представление
10100000	–96
+	+
11101001	–23
110001001	–119

Жирным шрифтом отмечена единица, выходящая за пределы разрядной сетки. Оставшиеся 8 двоичных цифр – это число, представленное в дополнительном коде. Найдем его модуль в десятичном представлении:

Запись результата	10001001	
Инвертированные биты:	01110110	
Плюс 1:	01110111	119_{10}

Следовательно, это дополнительный код числа –119, и результат получился правильным.

Вариант 4

Рассмотрим сложение чисел –56 и –107. Сначала найдем их дополнительные коды.

десятичное число	двоичное представление модуля	обратный код	дополнительный код
–56	111000	11000111	11001000
–107	1101011	10010100	10010101

Сложим дополнительные коды по правилам машинной арифметики.

Дополнительный 8-разрядный код	Десятичное представление
11001000	–56
+	+
10010101	–107
101011101	–163

Единица вышла за пределы разрядной сетки (выделена жирным шрифтом). Кроме того, знаковый бит теперь равен 0. Это говорит о том, что в результате сложения двух отрицательных чисел получилось положительное число! Это число $1011101_2 = 93_{10}$. Результат является некорректным.

Вариант 5

Убедимся, что результат, получаемый компьютером при сложении положительного и отрицательного числа, получается корректным.

Вычтем 42 из 65. Дополнительный код числа –42 равен 11010110. Двоичное представление числа 65 равно 01000001.

Прибавим к +65 –42 по правилам машинной арифметики.

Дополнительный 8-разрядный код	Десятичное представление
01000001	65
+	+
11010110	–42
100010111	23

Результат 23 является корректным.

Очевидно, что при любых числах разного знака результат умещается в диапазон представления чисел в целом формате, и для этого случая вычисления в компьютере всегда будут давать верный результат, вне зависимости от того, произошел перенос за пределы разрядной сетки (как в приведенном случае) или нет.

Умножение целых чисел

При умножении двух целых беззнаковых чисел размером в n бит максимальный результат, получаемый микропроцессором, может иметь длину $2n$ бит.

Простейший способ умножения целых беззнаковых чисел $X \times Y = Z$ основан на методах совместного анализа цифр множителя. Эти методы дробят процесс получения произведения на ряд шагов, связанных с формированием *частичных произведений* (*ЧП*) – произведений множимого на отдельные разряды или группы разрядов множителя – и их суммированием, то есть формированием *суммы частичных произведений* (*СЧП*).

Методы *умножения двоичных беззнаковых чисел* основаны на представлении произведения в виде полинома:

$$Z = X \cdot Y = X \cdot \left(\sum_{i=0}^{n-1} y_i \cdot 2^i \right) = \sum_{i=0}^{n-1} (X \cdot y_i) \cdot 2^i = \sum_{i=0}^{n-1} ЧП_i \cdot 2^i , \qquad (4.1)$$

где $y_i \in \{0,1\}$ – двоичные цифры множителя; $i = 0 \ldots n-1$; $ЧП_i = X \cdot y_i$ – частичные произведения ($ЧП_i = X$, если $y_i = 1$, и $ЧП_i = 0$, если $y_i = 0$).

Заметим, что умножение $ЧП_i \neq 0$ на степень 2^i эквивалентно сдвигу множимого на i разрядов влево.

Формирование произведения, согласно формуле (4.1), представляется как последовательность следующих действий:

1. Обнуление *СЧП*.
2. Умножение $ЧП_0 = X$ на y_0 и сложение с *СЧП*. При этом если $y_0 = 1$, то *СЧП* становится равным X, а если $y_0 = 0$, то и *СЧП* = 0.
3. Аанализ очередного разряда множителя: если $y_1 = 1$, $ЧП_1 = X$ и выполняется сложение $X \cdot 2^1$ с *СЧП*, а если $y_1 = 0$, то $ЧП_1 = 0$, и к *СЧП* ничего не прибавляется.

4. Анализ очередного y_2 разряда множителя, и т. д.

После анализа старшего y_{n-1} разряда множителя осуществляется последнее сложение $ЧП_{n-1} \cdot 2^{n-1}$ с *СЧП* (если $y_{n-1} = 1$), и процесс прекращается. Результирующая *СЧП* является искомой. Очевидно, что неподвижную *СЧП* и сдвигаемое влево на $n-1$ разряд множимое необходимо размещать в $2n$-разрядных форматах, причем операция сложения должна обрабатывать $2n$-разрядные данные.

Рассмотрим на примере умножение двух целых беззнаковых чисел. Найдем произведение 45 и 29, пользуясь машинным алгоритмом. В десятичной системе счисления 45·29 = 1305.

Таблица 4.5. Умножение целых положительных чисел

Шаг *i*	y_i	ЧП	СЧП
0	1	101101	101101
1	0	1011010	101101
2	1	10110100	101101 +10110100
3	1	101101000	101101 +10110100 +101101000
4	1	1011010000	101101 +10110100 +101101000 +1011010000 10100011001

Переведем множимое и множитель в двоичную систему счисления: $45_{10} = 101101_2$, $29_{10} = 11101_2$. Последовательность действий при умножении приведена в табл. 4.5.

Результат произведения $10100011001_2 = 1305_{10}$. Таким образом, можно сделать вывод, что алгоритм работает правильно.

Очевидно, что в случае, когда результат перемножения этих чисел будет помещен в один байт, как и оба операнда, результат получится неверным, поскольку он занимает 11 разрядов. В разрядной сетке результата будет 00011001, то есть 25_{10}. Можно провести машинный эксперимент – написать программу умножения двух 8-битных чисел с выводом результата также в 8-битное число. При отключенной проверке выхода результата за разрядную сетку на экран будет выведено 25. Таким образом, для корректного умножения всего диапазона 8-битных чисел необходимо под результат отвести не менее 16 бит. Команда ассемблера MUL поступает именно так: в двух 8-битных регистрах находятся операнды, а для записи результата служит еще два 8-битных регистра. При умножении на языке высокого уровня эти особенности следует учитывать.

При умножении *знаковых чисел* в компьютере используют два варианта. Первый заключается в том, что находятся модули чисел, отдельно запоминается их знак, потом модули чисел перемножаются, и затем произведению присваивается нужный знак, в зависимости от знаков сомножителей. Конечно, наличие дополнительного кода усложняет процедуру умножения, но тот выигрыш, который получается при использовании операции сложения, все равно весьма высок, поэтому от дополнительного кода в представлении чисел еще не отказались.

Второй вариант умножения – найти произведение двоичных знаковых чисел в дополнительных кодах. При этом необходимо ввести поправку на результат, которая зависит от того, какие числа участвуют в операции умножения.

Проведем эксперимент. Попробуем умножить два знаковых целых числа по правилам умножения беззнаковых целых чисел.

В двоичном представлении	Беззнаковые числа	Знаковые числа
11111111 × 11111111	255 × 255	–1 × –1
1111111000000001	65025	–511

Можно сделать вывод, что правило умножения для беззнаковых целых чисел напрямую не подходит для знаковых чисел, представленных в дополнительном коде. Следовательно, просто переносить метод умножения беззнаковых чисел на знаковые нельзя.

Рассмотрим, что можно сделать в этом случае. Для этого воспользуемся полученными знаниями о представлении чисел в дополнительном коде. Пусть целые двоичные сомножители X и Y представлены в дополнительном коде в n-разрядном формате, старший разряд которого используется для представления знака. Произведение записывается в слово, длина которого вдвое больше, чем длина сомножителей. При умножении этих чисел возможны четыре ситуации:

1. $X > 0$, $Y > 0$. Так как $D(X, n) = X$ и $D(Y, n) = Y$, то $D(X \cdot Y, n) = X \cdot Y$, т. е. результат умножения совпадает с произведением беззнаковых чисел.

2. $X < 0$, $Y > 0$. Согласно формуле для дополнения числа $D(X < 0, n) = 2^n - |X|$, и произведение, полученное методом умножения беззнаковых чисел, – *псевдопроизведение* – равно $D(X,n) \cdot Y = 2^n \cdot Y - |X| \cdot Y$. Правильный же результат в дополнительном коде равен $D(X \cdot Y < 0, n) = 2^{2n} - |X| \cdot Y$. Очевидно, что для получения правильного результата из псевдопроизведения необходимо к последнему прибавить 2^{2n} и вычесть множитель $Y \cdot 2^n$, то есть вычесть множитель, сдвинутый влево на n разрядов. Эта

величина является дополнением числа $Y \cdot 2^n$ в $2n$-разрядном коде: $D(Y \cdot 2^n, 2n) = 2^{2n} - Y \cdot 2^n$.

3. $X > 0$, $Y < 0$. Эта ситуация аналогична предыдущей. Для получения правильного произведения необходимо вычесть дополнение множимого, сдвинутого влево на n разрядов.

4. $X < 0$, $Y < 0$. Так как $D(X < 0, n) = 2^n - |X|$ и $D(Y < 0, n) = 2^n - |Y|$, то псевдопроизведение равно $D(X,n) \cdot D(Y,n) = 2^{2n} - 2^n \cdot |X| - 2^n \cdot |Y| + X \cdot Y$, а правильное произведение – просто $|X| \cdot |Y|$, поскольку оно положительно. Следовательно, для получения правильного произведения из псевдопроизведения необходимо *вычесть* число $2^{2n} - 2^n \cdot |X| - 2^n \cdot |Y|$, то есть дополнение суммы множителя и множимого, сдвинутых предварительно на n разрядов влево. Это число получается сложением дополнений чисел X и Y и сдвигом их влево на 8 двоичных разрядов. Применим при вычитании правила машинной арифметики (для этого полученное число переводим в дополнительный код) и получим, что для корректного определения произведения к псевдопроизведению необходимо добавить следующее число: $2^{2n} - (2^{2n} - 2^n \cdot |X| - 2^n \cdot |Y|)$, являющееся дополнением ранее полученного числа $2^{2n} - 2^n \cdot |X| - 2^n \cdot |Y|$.

Приведем примеры для каждого рассмотренного случая.

1. При умножении двух положительных знаковых чисел алгоритм полностью совпадает с алгоритмом умножения беззнаковых чисел (см. примеры выше).

2. Пусть $X = -5$, а $Y = 32$. Дополнительный код числа X равняется $D(X, 8) = 11111011$, $Y = 00100000$. Тогда псевдопроизведение равно:

$$\begin{array}{r} 11111011 \\ \times\ 00100000 \\ \hline 0001111101100000 \end{array} \qquad \begin{array}{r} -5 \\ \times\ 32 \\ \hline -160 \end{array}$$

Найдем реальное произведение, добавив к полученному псевдопроизведению следующую поправку:

$$2^{16} - Y \cdot 2^8 = 2^{16} - 00100000_2 \cdot 100000000_2 = 1110000000000000_2$$

$$\begin{array}{r} 0001111101100000 \\ +\ 1110000000000000 \\ \hline 1111111101100000 \end{array}$$

Число 1111111101100000 является дополнительным кодом числа –160. Следовательно, результат правильный.

3. Данный случай решается аналогично предыдущему.

4. Пусть X = –5, а Y = –32. Тогда D(X, 8) = 11111011, D(Y, 8) = 11100000. Отсюда, псевдопроизведение равно:

$$\begin{array}{rr} \times\ 11111011 & \times\ -5 \\ \hline 11100000 & -32 \\ \hline 1101101110100000 & 160 \end{array}$$

Найдем реальное произведение, добавляя к полученному псевдопроизведению поправку.

Найдем в дополнительном коде сумму

$$2^8 \cdot X + 2^8 \cdot Y = 11111011_2 \cdot 100000000_2 + 11100000_2 \cdot 100000000_2 =$$

$$= (11111011_2 + 11100000_2) \cdot 100000000_2 = 1110110110\,0000000_2$$

Старший бит отбрасывается по причине выхода за пределы разрядной сетки.

Дополнение этого числа равно 0010010100000000.

Сумма:

$$\begin{array}{r} +\ 1101101110100000 \\ \hline 0010010100000000 \\ \hline \mathbf{1}0000000010100000 \end{array}$$

В разрядной сетке осталось число +160, что и требовалось получить (старший бит, обозначенный жирным шрифтом, находится за границами разрядной сетки и не учитывается).

Над множеством целых чисел со знаком операция деления не определена, поскольку в общем случае ее результатом будет вещественное число. Однако допустимыми являются операции целочисленного деления и нахождения остатка от целочисленного деления (те, что в Паскале обозначаются div и mod). Точнее, значения обеих величин находятся одновременно в одной процедуре, которая в конечном счете сводится к последовательности вычитаний или, как это обычно делается в компьютере, сложений с дополнительным кодом делителя.

Представление вещественных чисел

Система вещественных чисел, применяемая при ручных вычислениях, предполагается бесконечной и непрерывной. Это означает, что не существует никаких ограничений на диапазон используемых чисел и точность их представления. Для любого вещественного числа имеется бесконечно много чисел, которые больше или меньше его, а между любыми двумя вещественными числами также находится бесконечно много вещественных чисел.

Реализовать такую систему в технических устройствах невозможно. Во всех компьютерах размеры ячеек памяти фиксированы, что ограничивает систему представимых чисел. Ограничения касаются как диапазона, так и точности представления чисел, т. е. система машинных чисел оказывается конечной и дискретной, образуя подмножество сис-

темы вещественных чисел. В компьютерном представлении вещественные числа часто представляются на основе нормализованного вида числа, известного в математике. Такое представление характерно разделением на мантиссу и порядок.

Рассмотрим, как нормализованные числа определяют математики.

Число A_{10} называется нормализованным, если оно представлено в виде:

$$A_{10} = \pm M_{10} \cdot 10^{\pm P_{10}},$$

где M_{10} – мантисса, десятичная правильная дробь, то есть $0{,}1 \le M_{10} < 1$; P_{10} – порядок, целое десятичное число.

При нормализации числа происходит своеобразное расчленение его на составляющие: знак числа, знак порядка, модуль мантиссы, модуль порядка.

$297_{10} = 0{,}297 \cdot 10^3$	$M_{10} = 0{,}297$	$P_{10} = 3$
$0{,}031_{10} = 0{,}31 \cdot 10^{-1}$	$M_{10} = 0{,}31$	$P_{10} = -1$
$-34{,}176 = -0{,}34176 \cdot 10^2$	$M_{10} = -0{,}34176$	$P_{10} = 2$

Пример 1. Приведите к нормализованному виду следующие числа, не переводя их в десятичную систему счисления: $-0{,}0000010111_2$; $98765{,}ABC_{16}$.

Решение:

$-0{,}0000010111_2 = -0{,}10111_2 \cdot 10_2^{-101_2}$;

$98765{,}ABC_{16} = 0{,}98765ABC_{16} \cdot 10_{16}^{+5_{16}}$.

Пример 2. Запишите в естественной форме следующие нормализованные числа: $0{,}1011_2 \cdot 10_2^{10_2}$, $0{,}12345_{10} \cdot 10_{10}^{3}$, $0{,}ABCD_{16} \cdot 10_{16}^{-1}$.

Решение:

$0{,}1011_2 \cdot 10_2^{10_2} = 0{,}1011_2 \cdot 100_2 = 10{,}11_2$;

$0{,}12345_{10} \cdot 10_{10}^{3} = 0{,}12345_{10} \cdot 1000_3 = 123{,}45_{10}$;

$0{,}ABCD_{16} \cdot 10_{16}^{-1} = 0{,}ABCD_{16} \cdot 0{,}1_{16} = 0{,}0ABCD_{16}$.

В ЭВМ арифметические устройства работают с нормализованными числами, но не с десятичными, а с двоичными. В некоторых представлениях мантиссы вещественных чисел существует так называемый неявный бит, который не записывается в памяти, но при вычислениях всегда предполагают, что он равен 1. Запись двоичного действительного числа с неявным (или явным) битом несколько отличается от записи нормализованного числа с мантиссой $0{,}1 \le M_2 < 1$, поскольку его мантисса принимает значения $1 \le M_2 < 2$ (целая часть всегда равна 1). Например: $1011{,}1_2 = 1{,}0111_2 \cdot 10_2^{11_2}$.

Здесь мантисса $1{,}0111_2$, порядок 11_2.

Рассмотрим, как в компьютере выглядит вещественное число в *формате с плавающей точкой* (или плавающей запятой). Значащие

цифры числа находятся в поле мантиссы; поле порядка показывает фактическое положение двоичной запятой в разрядах мантиссы, а бит знака определяет знак числа.

Мантисса, называемая также «дробью» *F* (*Fraction*), представлена в прямом коде. Порядок *E* (*Exponent*) дается в смещенной форме: он равен истинному порядку, увеличенному на значение смещения: E=(истинный порядок)+смещение(bias). Значение смещения для трех разных форматов равно 127 (на порядок отводится 8 бит), 1023 (11 бит) и 16383 (15 бит). Задание порядка в форме со смещением упрощает операцию сравнения чисел с плавающей точкой, превращая ее в операцию сравнения целых чисел. Смещенный порядок называется также «характеристикой», ее можно считать целым положительным и беззнаковым числом.

Значение числа равно:

$$(-1)^s \cdot 2^{E-bias} \cdot F_0, F_1 F_2 ... F_n,\ F_0 = 1,$$

где *n* для разных форматов равно 23, 52 или 63.

Структура записи действительного числа в формате с плавающей запятой имеет следующий вид:

S	*Exponent*	*Fraction*

В следующей таблице приведены характеристики различных форматов чисел с плавающей запятой:

Название формата	Кол-во битов знака	Кол-во битов порядка	Смещение	Кол-во битов мантиссы	Общее число байт	Примечание
Короткое вещественное (single)	1	8	127	23	4	первый бит – неявный
Длинное вещественное (double)	1	11	1023	52	8	первый бит – неявный
Расширенное вещественное (extended, временное вещественное)	1	15	16383	64	10	в записи присутствует явный бит

Отметим наличие в мантиссе бита единиц F_0. В коротком и длинном вещественных форматах бит F_0 при передачах чисел и хранении их в памяти не фигурирует. Это так называемый скрытый или *неявный бит*, который в нормализованных числах содержит единицу. Наличие этого неявного бита экономит один двоичный порядок, что экономит один бит в представлении мантиссы и повышает точность записи вещественного числа. Такой формат записи применяется при хранении данных в памяти.

Числа в расширенном (временном) вещественном формате имеют *явный бит* F_0. Такой формат позволяет несколько повысить скорость выполнения операций и используется в сопроцессоре, в этот формат числа переводятся непосредственно перед вычислениями. Числа во временном вещественном формате называются еще числами с расширенной точностью.

Эти типы данных широко распространены. Например, в Паскале короткое вещественное обозначается как single (4 байта), а длинное вещественное – как double (8 байт). Временное вещественное аналогично типу extended. Независимо от исходного формата при загрузке вещественного числа в сопроцессор оно автоматически преобразуется во временный формат, а при записи в память осуществляется обратное преобразование. Благодаря этому аппаратному преобразованию всех форматов данных во временный вещественный формат программист не должен следить за явным преобразованием форматов.

Рассмотрим примеры преобразования чисел из привычного формата в компьютерный.

Пример 3. Представим десятичное число –247,375 в вещественных форматах. Двоичный код его равен $-11110111{,}011_2$ и истинный порядок получается $+7_{10} = 111_2$. Смещенный порядок в трех вещественных форматах равен: $134_{10} = 10000110_2$, $1030_{10} = 10000000110_2$ и $16390_{10} = 100000000000110_2$.

	Знак	Порядок	Мантисса
Короткое вещественное (single)	1	10000110	11101110110...0
Длинное вещественное (double)	1	10000000110	11101110110...0
Расширенное вещественное (extended, временное вещественное)	1	100000000000110	111101110110...0

Пример 4. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления $A = \text{C2F20000}_{16}$. Тип переменной A – single для языка программирования Паскаль. Найти десятичное значение числа A.

Решение:

Преобразуем шестнадцатеричное представление в двоичное. Вычленим из представления знак, порядок и мантиссу:

$$A = \text{C2F20000}_{16} = 11000010111100100000000000000000_2.$$

Знак числа – старший бит – 1. Следовательно, искомое число отрицательное.

Порядок числа в памяти компьютера хранится в прямом коде со смещением. Найдем его: $P_{см} = 10000101_2$. Найдем реальный порядок, от порядка со смещением отнимем величину смещения 127_{10}:

$$P_{реал} = 10000101_2 - 01111111_2 = 110_2 = 6_{10}.$$

Следовательно, знак «двоичной» запятой в мантиссе нужно сместить вправо на 6 позиций.

Вспомним, что первый бит мантиссы – неявный, поэтому реальная мантисса имеет вид $M = 1{,}111001_2$.

Совмещаем знак, мантиссу и порядок: $A = -1111001_2 = -121{,}0_{10}$.

Вещественная арифметика

Пусть $a = \pm 0{,}m_a \cdot 2^{q_a}$, $b = \pm 0{,}m_b \cdot 2^{q_b}$ – два нормализованных двоичных числа, и $q_a \geq q_b$ (в противном случае мы можем просто поменять их местами). Результатом их *сложения* или *вычитания* будет являться следующее выражение: $c = (0{,}m_a \pm 0{,}m_b \cdot 2^{q_b - q_a}) \cdot 2^{q_a}$.

Порядок вычислений при этом таков:

1. Порядки чисел *a* и *b* выравниваются по большему из них (в нашем случае это q_a). Для этого мантисса числа *b* сдвигается на $q_a - q_b$ разрядов вправо (часть значащих цифр при этом могут оказаться утерянными), а его порядок становится равным q_a.
2. Выполняется операция сложения (вычитания) над мантиссами с округлением по значению *n*+1-й значащей цифры результата.
3. Мантисса результата должна быть нормализована (получившийся после нормализации порядок может отличаться от q_a как в меньшую, так и в большую сторону).

Все последующие упражнения будем выполнять в двоичной системе счисления без перевода в машинное представление!

Пример 5. Сложите два числа с плавающей запятой в двоичной системе счисления: $0{,}1001_2 \cdot 10_2^{101}$ и $0{,}111_2 \cdot 10_2^{-11}$.

Решение:

Перед сложением необходимо выровнять порядки – меньший порядок «приводится» к большему. В данном случае второе слагаемое приводится к виду: $0{,}00000000111_2 \cdot 10_2^{101}$. После чего выполняем сложение:

$$\begin{array}{r} 0{,}00000000111 \times 10^{101} \\ + \\ 0{,}10010000000 \times 10^{101} \\ \hline 0{,}10010000111 \times 10^{101} \end{array}$$

Если наша ячейка памяти имела бы восемь разрядов для хранения мантиссы, то последние три единичных разряда бы пришлось округлять.

При вычислении по данному алгоритму могут возникнуть следующие ошибки.

1. Потеря значащих цифр мантиссы у меньшего из чисел при выравнивании порядков.

2. Потеря крайней справа значащей цифры результата при сложении или вычитании мантисс.
3. Выход за границу допустимого диапазона значений того или иного вещественного типа при нормализации результата.

Рассмотрим теперь умножение и деление нормализованных чисел *a* и *b* в машинной арифметике (с ограниченным числом разрядов):

$$d = a \cdot b = (0,m_a \cdot 0,m_b) \cdot 2^{q_a+q_b};$$

$$f = a \div b = (0,m_a \div 0,m_b) \cdot 2^{q_a-q_b}$$

Для получения результата в данном случае производятся следующие действия, причем уже не важно, какое из двух данных чисел больше.

1. Мантиссы перемножаются (или одна делится на другую), при этом результат округляется до *n* значащих цифр (отметим, что при умножении может получиться 2*n* значащих цифры в мантиссе, а при делении – бесконечно много).
2. Порядки при умножении складываются, а при делении вычитаются.
3. При необходимости мантисса результата нормализуется.

Над мантиссами в арифметическом устройстве компьютер должен уметь выполнять все четыре операции (сложение, вычитание, умножение и деление), а также операции сдвига, тогда как над порядками производятся только действия сложения и вычитания. Для осуществления этих сложных для компьютера операций в его составе должен быть арифметический сопроцессор.

Пример 6. Выполнить умножение двух нормализованных двоичных числа, пользуясь алгоритмом машинной арифметики: $0{,}1001_2 \cdot 10_2^{101}$ и $0{,}111_2 \cdot 10_2^{-11}$.

Решение:

Складываем порядки этих чисел: 101 – 11 = 01.

Перемножаем мантиссы:

$$\begin{array}{r} 0{,}1001 \\ \times\ 0{,}111 \\ \hline 0{,}0111111 \end{array}$$

В результате получили число: $0{,}0111111_2 \cdot 10_2^{10}$. Приведем мантиссу к нормализованному виду: $0{,}111111_2 \cdot 10_2^{1}$.

Если бы в этом примере были большие значения порядков (например, 32 и 76), то в машинном представлении результата биты, выделенные под хранение порядка, не смогли бы вместить истинный порядок числа. Следовательно, операция умножения не будет произведена по причине *переполнения порядка*.

Пример 7. Выполнить деление двух нормализованных двоичных числа, пользуясь алгоритмом машинной арифметики: $0{,}1001_2 \cdot 10_2^{101}$ разделить на $0{,}111_2 \cdot 10_2^{-11}$.

Решение:

Найдем порядок результата: 101 – (–11) = 101 + 11 = 1000.

Делим мантиссы (столбиком): $0{,}1001 \div 0{,}111 = 0{,}10(1001)$. Мантисса получилась нормализованной, но (!) периодической. Что же делать?

Когда говорят о точности представления десятичных вещественных чисел, нужно помнить следующее: десятичное число невозможно записать точно в любом из машинных вещественных типов. Объясняется это тем, что конечные десятичные дроби часто оказываются бесконечными периодическими двоичными дробями (см. упражнения по переводу правильных дробей из десятичной системы счисления в двоичную). Следовательно, в нормализованном виде такое число будет иметь бесконечную мантиссу и не может быть представлено точно. При записи подобной мантиссы в память компьютера число *округляется*.

Если под мантиссу отведено n бит и n+1 значащая цифра двоичной нормализованной мантиссы равна 0, то цифры начиная с n+1-й просто отбрасываются, если же n+1-я цифра равна единице, то к целому числу, составленных из n значащих цифр мантиссы, прибавляется единица.

Запись чисел в файле данных

При записи данных на диск первым пишется самый младший байт числа, затем более старший, и т. д. Таким образом, при чтении числа в формате «Integer» из файла данных сначала читается младший байт, затем старший. При чтении числа в формате single первым стоит младший байт мантиссы, затем более старший байт, а последний из 4 байтов содержит знаковый бит и первые 7 бит порядка. Наиболее распространенные программы, предоставляющие возможности для просмотра и редактирования файлов данных в кодах ЭВМ, – это файловые менеджеры Far и Total Commander, а также битовый редактор HexEdit. В них наиболее часто применяется шестнадцатеричное представление чисел.

Рассмотрим представление чисел целых форматов в файлах данных.

Формат Shortint занимает всего один байт и поэтому последовательность чисел этого формата соответствует последовательности байтов.

Формат Integer занимает два байта. В качестве примера рассмотрим последовательный вывод двух целых чисел: 2315 и 1280 в файл данных типа Integer. Переведем эти числа в двоичную и шестнадцатеричную системы счисления: $2315_{10} = 100100001011_2 = 90B_{16}$; $1280_{10} = 10100000000_2 = 500_{16}$. В файл данных будет выведено: 0B 09 00 05.

Для проверки напишите самостоятельно программу на Паскале, выводящую в файл данных числа формата Integer и сравните результаты работы программы и данные, полученные вручную.

Контрольные вопросы и задания

1. Дайте определение дополнительного кода целого числа.
2. Чем различаются представление чисел в дополнительном 8-разрядном и 16-разрядном кодах?
3. Найдите дополнительный 8-разрядный и дополнительный 16-разрядный коды десятичных чисел –123, –95, –48.
4. Напишите на Паскале программу сложения двух чисел типа shortint. Найдите результат сложения следующих двух чисел: 100 + 99; (–2) + (–127); 110 + (–96); 65 + (–118); 94 + 11; (–31) + (–27). Проанализируйте результаты. Проведите вычисления, пользуясь алгоритмом машинной арифметики. Сравните компьютерные и полученные вручную результаты.
5. Напишите аналогичную программу на Паскале для типа данных integer, longint, word. Определите диапазон чисел, представляемых в этих типах данных.
6. Напишите на языке Паскаль программу вывода в файл чисел типа byte, shortint, integer, longint. Попробуйте вывести различные числа в файлы данных. Просмотрите файлы данных в Far в режиме двоичного просмотра (HEX). Проанализируйте результаты.
7. Ответьте на вопросы:
   Сколько байт данных необходимо для представления чисел в различных форматах?
   Как перевести число из его представления его в файле к привычному виду?
   Как представляются в файле различные значения переменных?
8. Напишите программу ввода данных из файла в память компьютера. Проверьте, как читаются данные из файла. Возможна ли ситуация, когда числа из файла будут прочитаны неправильно?
9. Выведите в файл числа, определенные как single и double. Просмотрите их в Far. Переведите числа из представленных в компьютере в обычный вид. Проанализируйте результаты.

Список литературы к лекции 4

1. *Григорьев В.Л.* Архитектура и программирование арифметического сопроцессора. – М. : Энергоатомиздат, 1991. – 208 с. : ил.
2. *Поворознюк А.И.* Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств : учеб. пособие. Ч. 1. – Харьков : Торнадо, 2004. – 355 с. : ил.

Лекция 5. АЛГЕБРА ЛОГИКИ И ЛОГИЧЕСКИЕ ФУНКЦИИ

Алгебра логики – один из основных разделов математической логики, в которых методы алгебры используются в логических преобразованиях высказываний, рассматриваемых относительно истинности их значений (высказывания могут иметь значения «истина» и «ложь»). Алгебра логики может использоваться не только для логических преобразований, но и для арифметических вычислений, что очень важно при разработке устройств обработки информации, ведь тогда одно и то же физическое устройство может проводить и логические, и арифметические преобразования. Основы математической логики заложил немецкий ученый и философ *Готфрид Вильгельм Лейбниц* (1646–1716). Он сделал попытку построить первые логические исчисления, считал, что можно заменить простые рассуждения действиями со знаками и привел соответствующие правила. Но Лейбниц высказал только идею, а развил ее окончательно ирландский математик англичанин *Джордж Буль* (1815–1864). Буль разработал *логическое исчисление*, в котором применяются законы и операции математики. Эта логическая система способствовала возникновению алгебры логики. Алгебра логики явилась первой системой математической логики, в которой алгебраическая символика применялась к логическим выводам. Создатель этой системы ставил перед собой цель решать логические задачи с помощью методов, применяемых в алгебре. Любое суждение он пытался выразить в виде уравнений с символами, в которых действуют логические законы, подобные законам алгебры (например, законы коммутативности, ассоциативности, дистрибутивности и др.).

Большой вклад в развитие и усовершенствование алгебры логики внесли немецкий логик и математик Давид Гильберт, а в дальнейшем английский философ и логик Бертран Рассел с английским математиком Альфредом Уайтхедом придали математической логике ее современный вид[2]. В дальнейшем Клод Шеннон показал, как можно использовать алгебру логики для описания работы релейных схем, и эти достижения использованы в современной компьютерной технике.

Основные положения алгебры логики

Основными понятиями алгебры логики являются логический аргумент и логическая функция. Логический аргумент (или высказывание) в зависимости от смысла может принимать значение «истина» или «ложь».

[2] *Кондаков Н.И.* Логический словарь-справочник. М. : Наука, 1975.

Это соответствует значениям «1» и «0». Логический аргумент входит в состав сложного высказывания – логической функции, зависящей от истинности или ложности аргумента. Логическая функция также принимает значения «1» или «0».

Таблица 5.1. Пример логической функции трех аргументов

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

К основным логическим операциям в алгебре логики относятся дизъюнкция (ИЛИ, логическое сложение, «+», «∨»), конъюнкция (И, логическое умножение, «·», «∧») и отрицание (НЕ, инверсия, $\bar{x}$). Кроме того, определено отношение эквивалентности (=). Оно удовлетворяет свойству рефлексивности: $x = x$, симметричности: если $x = y$, то $y = x$; и транзитивности: если $x = y$ и $y = z$, то $x = z$. Это соотношение дает принцип подстановки: если $x = y$, то в любой формуле, где есть x, можно заменить его на y, и будет получена эквивалентная формула.

Алгебра логики использует следующую систему аксиом:

$$\left.\begin{array}{l} x = 1, \text{если } x \neq 0, \\ x = 0, \text{если } x \neq 1. \end{array}\right\} \qquad (5.1)$$

$$\left.\begin{array}{l} 1+1=1, \\ 0\cdot 0=0, \\ 0+0=0, \\ 1\cdot 1=1, \\ 0+1=1+0=1, \\ 1\cdot 0=0\cdot 1=0. \end{array}\right\} \qquad (5.2)$$

$$\left.\begin{array}{l} \bar{0}=1, \\ \bar{1}=0. \end{array}\right\} \qquad (5.3)$$

Первая аксиома утверждает, что в алгебре логики рассматриваются только двоичные переменные, принимающие значения 0 или 1. Вторая группа аксиом определяет операции дизъюнкции и конъюнкции (логического сложения и умножения). Третья группа аксиом определяет операцию отрицания (инверсии).

С помощью этих аксиом доказываются все теоремы алгебры логики. При этом в алгебре логики существует возможность доказательства утверждений методом перебора. Теорема истинна, если при подстановке

любых значений переменных она превращается в верное тождество. Этот метод перебора не слишком трудоемок, поскольку переменные могут принимать только значения 0 и 1.

Логическая функция от n аргументов может быть задана таблицей, в которой перечислены все возможные наборы из 0 и 1 длины n и для каждого из них указано значение функции. Наборы обычно перечисляются в порядке возрастания чисел, двоичными записями которых они являются. В табл. 5.1 приведен пример логической функции от трех аргументов.

Таблицы для функций от n аргументов x_1, ..., x_n имеют 2^n строк (по числу двоичных наборов длины n). Различные таблицы отличаются лишь последним столбцом и, поскольку количество различных двоичных столбцов длины 2^n составляет 2^{2^n}, число функций от n аргументов x_1, ..., x_n равно 2^{2^n}. В это число включены все возможные функции, в том числе и те, которые зависят от некоторых аргументов фиктивно.

В простейшем примере логическая функция зависит от одной переменной (см. табл. 5.2). Здесь функции $F_0(x)$, $F_1(x)$, $F_3(x)$ зависят от аргумента фиктивно.

Таблица 5.2. Возможные логические функции одного аргумента

Аргумент	Функция			
x	$F_0(x)$	$F_1(x)$	$F_2(x)$	$F_3(x)$
0	0	0	1	1
1	0	1	0	1

Приведем названия этих функций.
$F_0(x)$ – константа 0,
$F_1(x)$ – переменная x,
$F_2(x)$ – инверсия x,
$F_3(x)$ – константа 1.
Интересной является только функция $F_2(x)$.
Функций двух аргументов уже 16 (см. табл. 5.3).

Таблица 5.3. Логические функции двух аргументов

Аргумент		Функция															
x_1	x_2	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Одни из этих функций тривиальны, другие активно используются в алгебре логики. Названия этих функций и их условные обозначения приведены в табл. 5.4.

Таблица 5.4. Названия и условные обозначения логических функций двух переменных

	Название	**Условное обозначение**
F_0	константа 0	0
F_1	конъюнкция, логическое умножение, И	$x_1 \cdot x_2; x_1 \wedge x_2; x_1 \& x_2; x_1 x_2$
F_2	запрет по x_1, отрицание импликации	$x_1 \Delta x_2$
F_3	переменная x_1	x_1
F_4	запрет по x_2, отрицание импликации	$x_2 \Delta x_1$
F_5	переменная x_2	x_2
F_6	сумма по модулю 2, логическая неравнозначность	$x_1 \oplus x_2$
F_7	дизъюнкция, логическое сложение, ИЛИ	$x_1 + x_2; x_1 \vee x_2$
F_8	стрелка Пирса, отрицание дизъюнкции	$x_1 \downarrow x_2$
F_9	эквивалентность	$x_1 \equiv x_2; x_1 \sim x_2$
F_{10}	отрицание, инверсия x_2	$\overline{x_2}$
F_{11}	импликация от x_2 к x_1	$x_2 \rightarrow x_1$
F_{12}	отрицание, инверсия x_1	$\overline{x_1}$
F_{13}	импликация от x_1 к x_2	$x_1 \rightarrow x_2$
F_{14}	штрих Шеффера, отрицание конъюнкции	$x_1 \mid x_2$
F_{15}	константа 1	1

Если у функции 3 аргумента, то число возможных функций возрастает до 256, но в соответствии с законами алгебры можно представить их в виде функций двух аргументов, поэтому более сложные логические функции задаются с помощью более простых функций одного или двух аргументов. Для выражения сложных логических функций используют более простые, и оказывается, что можно использовать не все элементарные функции, а только часть.

Рассмотрим подробнее наиболее интересные логические функции одной и двух переменных.

Логическое умножение, или конъюнкция (от лат. «conjunctio» – связываю).

Таблица истинности конъюнкции имеет следующий вид:

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Конъюнкция двух логических переменных истинна тогда и только тогда, когда обе логических переменных истинны (принимают значение логической 1). Это определение можно обобщить для любого количества логических переменных, объединенных конъюнкцией: $x_1 \cdot x_2 \cdot x_3 = 1$, только если $x_1 = 1$, $x_2 = 1$, $x_3 = 1$.

Логическое сложение, или дизъюнкция (от лат. «disjunctio» – различаю) Таблица истинности дизъюнкции имеет следующий вид:

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Дизъюнкция двух логических переменных ложна(равна логическому 0) тогда и только тогда, когда обе переменных ложны (принимают значение 0).

Это определение можно обобщить для любого количества логических переменных, объединенных дизъюнкцией: $x_1 + x_2 + x_3 = 0$, только если $x_1 = 0$, $x_2 = 0$, $x_3 = 0$.

Следующие логические законы можно назвать свойствами дизъюнкции.

Логическое отрицание, или инверсия (от лат. «inversion» – переворачиваю). Таблица истинности инверсии имеет вид:

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

Инверсия логической переменной истинна (равна 1), если сама переменная ложна (равна 0), и, наоборот, инверсия ложна (равна 0), если переменная истинна (равна 1).

Импликация или логическое следование (от лат. «implicatio» – тесно связываю). Высказывание $x_1 \to x_2$ ложно в том и только в том случае, когда условие (первое высказывание x_1) истинно, а следствие (второе высказывание x_2) ложно.

x_1	x_2	$x_1 \to x_2$
0	0	1
0	1	1
1	0	0
1	1	1

Импликацию можно представить через дизъюнкцию и инверсию: $x_1 \to x_2 = \overline{x_1} + x_2$.

Свойства импликации:

$$x \to 0 = \bar{x};$$
$$x \to x = 1;$$
$$0 \to x = 1;$$
$$1 \to x = x.$$

Эквивалентность или равнозначность (от фр. «equivalence» – равноценность). Выражение $x_1 \leftrightarrow x_2$ истинно (равно 1) в том и только в том случае, когда оба исходных высказывания одновременно истинны (равны 1) или одновременно ложны (равны 0).

x_1	x_2	$x_1 \leftrightarrow x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Эквивалентность можно представить через конъюнкцию, дизъюнкцию и инверсию $x_1 \leftrightarrow x_2 = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$.

Свойства эквивалентности:

$$x \leftrightarrow x = 1;$$
$$x \leftrightarrow \overline{x} = 0;$$
$$x \leftrightarrow 0 = \overline{x};$$
$$x \leftrightarrow 1 = x.$$

Строгая дизъюнкция, или Сложение по модулю «2». Выражение $x_1 \oplus x_2$ истинно (равно 1) в том и только в том случае, когда переменные x_1 и x_2 не равны между собой.

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Представление эквивалентности через конъюнкцию, дизъюнкцию и инверсию $x_1 \oplus x_2 = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2$.

Сравнив таблицы истинности операций эквивалентности и сложения по модулю 2, можно сделать вывод, что эти операции являются инверсией друг друга, то есть $x_1 \leftrightarrow x_2 = \overline{x_1 \oplus x_2}$.

Свойства строгой дизъюнкции:

$$x \oplus x = 0;\ x \oplus \overline{x} = 1;\ x \oplus 0 = x;\ x \oplus 1 = \overline{x}.$$

Стрелка Пирса (символ Лукашевича). Выражение $x_1 \downarrow x_2$ истинно в том и только в том случае, когда обе переменных x_1 и x_2 ложны: $x_1 \downarrow x_2 = \overline{x_1 + x_2}$.

x_1	x_2	$x_1 \downarrow x_2$
0	0	1
0	1	0
1	0	0
1	1	0

Штрих Шеффера. Выражение $x_1 \mid x_2$ ложно в том и только в том случае, когда обе переменных x_1 и x_2 истинны: $x_1 \mid x_2 = \overline{x_1 \cdot x_2}$.

x_1	x_2	$x_1 \mid x_2$
0	0	1
0	1	1
1	0	1
1	1	0

По формуле логической функции легко рассчитать ее таблицу истинности. Необходимо только учитывать порядок выполнения логических операций (приоритет) и скобки. Операции в логическом выражении выполняются слева направо с учетом скобок. Для уменьшения количества скобок в формулах вводят «старшинство» для знаков логических операций. Принято считать, что знак дизъюнкции старше знаков импликации, эквивалентности и сложения по модулю «2», знак конъюнкции старше всех перечисленных, а знак инверсии старше всех остальных.

Определим, к примеру, таблицу истинности логической функции: $F(x_1, x_2, x_3) = x_1 + x_2 \cdot \overline{x_3}$

Определяем количество строк в таблице: $Q = 2^3 = 8$.

Определяем количество логических операций (их всего 3) и последовательность их выполнения. Затем определяем количество столбцов: три переменные плюс три результата логических операций (всего 6). Строим таблицу:

x_1	x_2	x_3	$\overline{x_3}$	$x_2 \cdot \overline{x_3}$	$x_1 + x_2 \cdot \overline{x_3}$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

Если у двух логических функций совпадают таблицы истинности, то есть на всех наборах значений входных переменных они принимают одинаковое значение, то их называют равносильными или эквивалентными. Это обозначается знаком тождества (=).

Пример. $x_1 + x_2 + x_3 = x_1 + (x_2 + x_3)$.

Логические функции, истинные на всех наборах значений входных переменных, называются тождественно-истинными.

Логические функции, ложные на всех наборах значений входных переменных, называются тождественно-ложными.

$F = x + 1 = 1$ – тождественно-истинная функция;
$F = x \cdot 0 = 0$ – тождественно-ложная функция.

Законы логики. Упрощение логических выражений

Рассмотрев основные положения алгебры логики, представим теперь математический формализм, в рамках которого рассматриваются логические и арифметические выражения.

Среди многочисленных законов логики есть четыре основных. Для трех из них можно найти аналогию в алгебре чисел. В их правильности легко убедиться, применяя метод перебора. Эти законы сведены в следующую таблицу:

Логические выражения	Алгебраические выражения
Переместительный закон. Закон коммутативности	
$x_1 + x_2 = x_2 + x_1$	$x_1 + x_2 = x_2 + x_1$
$x_1 \cdot x_2 = x_2 \cdot x_1$	$x_1 \cdot x_2 = x_2 \cdot x_1$
Сочетательный закон. Закон ассоциативности	
$(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$	$(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$
$(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$	$(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$
Распределительный закон. Закон дистрибутивности	
$(x_1 + x_2) \cdot x_3 = (x_1 \cdot x_3) + (x_2 \cdot x_3)$	$(x_1 + x_2) \cdot x_3 = (x_1 \cdot x_3) + (x_2 \cdot x_3)$
$(x_1 \cdot x_2) + x_3 = (x_1 + x_3) \cdot (x_2 + x_3)$	*аналога нет*
$(x_1 \oplus x_2) \cdot x_3 = (x_1 \cdot x_3) \oplus (x_2 \cdot x_3)$	*аналога нет*
Закон инверсии. Формулы де Моргана	
$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$ $\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$	*аналога нет*
$\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$ $\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$	*аналога нет*

Из закона ассоциативности следует, что можно рассматривать многоместную конъюнкцию (произведение многих аргументов):

$$\prod_{i=1}^{n} x_i = x_n \cdot \ldots \cdot x_1,$$

и многоместную дизъюнкцию (сумму):

$$\sum_{i=1}^{n} x_i = x_n + \ldots + x_1.$$

Законы де Моргана могут быть обобщены, соответственно, для любого числа аргументов:

$$\overline{\sum_{i=1}^{n} x_i} = \prod_{i=1}^{n} \overline{x_i}, \qquad \overline{\prod_{i=1}^{n} x_i} = \sum_{i=1}^{n} \overline{x_i}.$$

Клод Шеннон предложил обобщение этих теорем, позволяющее отыскивать инверсию любой функции $f(v)$, где $v = (x_n, \ldots x_1)$. Закон двойственности, установленный Шенноном, имеет вид: $\overline{f(v/+,\cdot)} = f(\bar{v}/\cdot,+)$, где $v = (x_n, \ldots x_1)$, $\bar{v} = (\overline{x_n}, \ldots \overline{x_1})$. Другими словами, инверсию любой функции $f(v)$ можно получить взаимной заменой переменных x_p и их инверсий $\overline{x_p}$ ($p = 1 \ldots n$) и операций дизъюнкции и конъюнкции.

Для упрощения логических выражений полезно знать следующие свойства:

Свойства идемпотентности:

$$x + x = x;$$
$$x \cdot x = x.$$

Закон двойного отрицания:

$$\overline{\overline{x}} = x.$$

Соотношения с участием констант:

$$x \cdot 0 = 0;\ x \cdot 1 = x;\ x \cdot \overline{x} = 0;$$
$$x + 1 = 1;\ x + 0 = x;\ x + \overline{x} = 1.$$

Кроме того, удобно также использовать формулы склеивания и поглощения.

Формулы склеивания (закон исключения):

$$(x_1 \cdot x_2) + (x_1 \cdot \overline{x_2}) = x_1 \cdot (\underbrace{x_2 + \overline{x_2}}_{1}) = \underbrace{x_1 \cdot 1}_{x_1} = x_1;$$

$$(x_1 + x_2) \cdot (x_1 + \overline{x_2}) = x_1 + (\underbrace{x_2 \cdot \overline{x_2}}_{0}) = \underbrace{x_1 + 0}_{x_1} = x_1.$$

Формулы поглощения:

$$x_1 + (\overline{x_1} \cdot x_2) = (\underbrace{x_1 + \overline{x_1}}_{1}) \cdot (x_1 + x_2) = \underbrace{1 \cdot (x_1 + x_2)}_{x_1 + x_2} = x_1 + x_2;$$

$$x_1 \cdot (\overline{x_1} + x_2) = (\underbrace{x_1 \cdot \overline{x_1}}_{0}) + (x_1 \cdot x_2) = \underbrace{0 + x_1 \cdot x_2}_{x_1 \cdot x_2} = x_1 \cdot x_2;$$

$$x_1 + (x_1 \cdot x_2) = (\underbrace{x_1 \cdot 1}_{x_1}) + (x_1 \cdot x_2) = x_1 \cdot (\underbrace{1 + x_2}_{1}) = \underbrace{x_1 \cdot 1}_{x_1} = x_1;$$

$$x_1 \cdot (x_1 + x_2) = (\underbrace{x_1 + 0}_{x_1}) \cdot (x_1 + x_2) = x_1 + (\underbrace{0 \cdot x_2}_{0}) = \underbrace{x_1 + 0}_{x_1} = x_1.$$

Используя законы логики, формулы склеивания и поглощения и свойства логических операций, можно сложную логическую функцию заменить более простой, но равносильной ей функцией. Этот процесс называется минимизацией функции. Минимизация необходима для того, чтобы функциональные схемы не были слишком громоздкими и не использовали лишних элементов. Чем меньше в функции, получаемой при минимизации, входных переменных и используемых логиче-

ских операций, тем проще логическая схема, меньше в ней логических элементов. Минимизация необходима и при составлении сложных логических выражений в программах.

Пример. Является ли функция $F(A,B,C) = (A \leftrightarrow C) \to (C + \overline{A+B})$ тождественно-истинной?

Решение: Решить данную задачу можно двумя способами.

Первый способ – минимизация логической функции.

$$F(A,B,C) = (A \leftrightarrow C) \to (C + \overline{A+B}).$$

Избавимся от операций импликации и эквивалентности, заменив эти операции на комбинацию конъюнкции, дизъюнкции и инверсии.

$$F(A,B,C) = (A \leftrightarrow C) \to (C + \overline{A+B}) = \overline{(A \leftrightarrow C)} + (C + \underbrace{\overline{A+B}}_{\overline{A}\cdot\overline{B}}) =$$

$$= (A \oplus C) + C + \overline{A} \cdot \overline{B} = A \cdot \overline{C} + \overline{A} \cdot C + C + \overline{A} \cdot \overline{B}$$

Последовательно несколько раз применим формулы поглощения:

$$F(A,B,C) = A \cdot \overline{C} + \underbrace{\overline{A} \cdot C + C}_{C} + \overline{A} \cdot \overline{B} = \underbrace{A \cdot \overline{C} + C}_{A+C} + \overline{A} \cdot \overline{B} = C + \underbrace{A + \overline{A} \cdot \overline{B}}_{A+\overline{B}} = A + \overline{B} + C.$$

Следовательно, данная функция не является тождественно-истинной.

Второй способ – построение таблицы истинности. У тождественно-истинной функции в последнем столбце таблицы истинности должны стоять все единицы.

У функции три переменные, следовательно, количество строк в таблице $2^3 = 8$.

Подсчитаем количество операций и установим порядок их выполнения.

$$F(A,B,C) = (A \overset{4}{\leftrightarrow} C) \overset{5}{\to} (C \overset{3}{+} \overset{2}{\overline{A \underset{1}{+} B}}).$$

Пять логических операций, следовательно, количество столбцов в таблице истинности: 3 + 5 = 8.

A	B	C	$A+B$	$\overline{A+B}$	$C+\overline{A+B}$	$A \leftrightarrow C$	$F(A,B,C)$
0	0	0	0	1	1	1	1
0	0	1	0	1	1	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	1	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	1	0	0	0	1
1	1	1	1	0	1	1	1

Анализ построенной таблицы показывает, что существует набор входных переменных, при котором функция равна 0. Следовательно, данная функция не является тождественно-истинной.

Представление логических функций

Набор логических функций называется функционально полной системой, если любую функцию алгебры логики можно записать в виде формулы через эти функции.

Примеры полных систем:

$$F_7(x_1, x_2) = x_1 + x_2,\ F_1(x_1, x_2) = x_1 \cdot x_2,\ F_{12}(x_1, x_2) = \overline{x_1};$$

$$F_8(x_1, x_2) = \overline{x_1 + x_2};$$

$$F_{14}(x_1, x_2) = \overline{x_1 \cdot x_2}.$$

Все другие функции могут быть выражены через функции полной системы. Это важное свойство полных систем широко используется при синтезе логических схем.

Первичные термы. Переменные x_p и их инверсии $\overline{x_p}$ называются первичными термами, для которых используется символическое обозначение степени:

$$x_p^{e_p} = \begin{cases} \overline{x_p}, \text{ если } e_p = 0, \\ x_p, \text{ если } e_p = 1. \end{cases}$$

Данное символическое обозначение объединяет в одном символе оба первичных терма x_p и $\overline{x_p}$.

Только благодаря введению данного символического обозначения удается формализовать вывод общих соотношений для переключательных функций. Очевидно, что два первичных терма $x_p^{e_p}$ и $x_p^{e'_p}$ равны только в том случае, если $e_p = e'_p$ (если $e_p \neq e'_p$, то $e_p = \overline{e'_p}$). Для первичных термов справедливы соотношения:

$$x_p^1 = \overline{x_p^0} = x_p,\ x_p^0 = \overline{x_p^1} = \overline{x_p};$$

$$\overline{x_p^{e_p}} = x_p^{\overline{e_p}} = \overline{x}_p^{e_p};$$

$$x_p^{e_p} \cdot x_p^{\overline{e_p}} = 0,\ x_p^{e_p} + x_p^{\overline{e_p}} = 1;$$

$$x_p^{e_p} = \begin{cases} 0, \text{ если } x_p = \overline{e_p}, \\ 1, \text{ если } x_p = e_p. \end{cases}$$

Минтермом (конституентой единицы) называется функция n переменных вида:

$$K_i(v) = x_n^{e_n} \cdot \ldots \cdot x_1^{e_1} = \prod_{p=1}^{n} x_p^{e_p},$$

где $v = (x_n, \ldots x_1)$, $e_p = 0$ или 1.

Поскольку индексы e_p принимают значение 0 или 1, то из них можно составить двоичное число $i = e_n \ldots e_1$. Из этого следует, что имеется 2^n различных минтермов n переменных, так как имеется 2^n различных n-разрядных двоичных чисел $i = 0, 1, \ldots, 2^n - 1$. Минтермы обладают следующим свойством:

$$K_i(v) = \begin{cases} 1, \text{ если } v = v_i, \\ 0, \text{ если } v = v_j \neq v_i. \end{cases}$$

Таким образом, любой минтерм принимает значение 1 при единственном из всех возможных наборов аргументов и значение 0 при всех остальных.

Макстерм (конституента нуля) – это функция *n* переменных

$$M_i(v) = \overline{K_i(v)} = \overline{\prod_{p=1}^{n} x_p^{e_p}} = \sum_{p=1}^{n} \overline{x_p^{e_p}}.$$

Макстерм обладает свойством

$$M_i(v) = \begin{cases} 0, \text{ если } v = v_i, \\ 1, \text{ если } v = v_j \neq v_i. \end{cases}$$

Таким образом, макстермы – это функции, принимающие значение 0 в одном из возможных наборов v_i и 1 при всех других.

Число переменных (аргументов), входящих в минтерм или макстерм, называется его рангом.

Примеры:

$x_1 \cdot \overline{x_2} \cdot x_3 \cdot x_4$ – минтерм 4-го ранга.

$x_1 + x_2 + \overline{x_3}$ – макстерм 3-го ранга.

Функция в дизъюнктивной нормальной форме (ДНФ) является логической суммой минтермов.

Пример: $\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2} + x_1 \cdot \overline{x_2} \cdot x_3$.

Совершенная дизъюнктивная нормальная форма (СДНФ) – это такая ДНФ, в которой каждый член суммы содержит ровно по одному разу все имеющиеся переменные (или их инверсии) и не содержит двух одинаковых слагаемых.

Пример: $\overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3}$.

Конъюнктивная нормальная форма (КНФ) является логическим произведением элементарных дизъюнкций (макстермов).

Пример: $(x_1 + x_2 + x_3) \cdot (x_1 + \overline{x_2})$.

Совершенная конъюнктивная нормальная форма (СКНФ) представляет такую конъюнктивную нормальную форму, в которой в каждом сомножителе все переменные или их инверсии встречаются по одному разу и нет двух одинаковых сомножителей.

Пример: $(x_1 + x_2 + x_3) \cdot (\overline{x_1} + x_2 + x_3) \cdot (\overline{x_1} + x_2 + \overline{x_3})$.

Запись логической функции по таблице

Любая логическая функция может быть выражена в виде СДНФ или СКНФ. В качестве примера рассмотрим произвольную функцию *f*, и покажем принцип построения СДНФ для нее (табл. 5.5).

Таблица 5.5. Построение СДНФ произвольной функции

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	G_0	G_1	G_4	G_5	G_7
0	0	0	1	1	0	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0
1	0	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	1

Функции G_0, G_1, G_4, G_5, G_7 – это минтермы (см. определение). Каждая из этих функций является произведением трех переменных или их инверсий и принимает значение 1 только в одной ситуации. Видно, что для того, чтобы получить 1 в значении функции f, нужен один минтерм. Следовательно, количество минтермов, составляющих СДНФ этой функции, равно количеству единиц в значении функции:

$$f = G_0 + G_1 + G_4 + G_5 + G_7.$$

Первый среди минтермов – $G_0 = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$. Он равен логической 1 только в случае, когда все логические переменные равны 0. Аналогично строятся формулы для каждого из минтермов, составляющих эту функцию.

Таким образом, СДНФ функции f имеет вид:

$$f(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3.$$

С использованием обозначения степени (которое во многих случаях бывает более удобным) эта формула выглядит следующим образом:

$$f(x_1, x_2, x_3) = x_1^0 \cdot x_2^0 \cdot x_3^0 + x_1^0 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^0 \cdot x_3^0 + x_1^1 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^1 \cdot x_3^1.$$

Аналогично можно построить СКНФ. Количество сомножителей равно количеству нулей в значениях функции:

$$f(x_1, x_2, x_3) = (x_1 + \overline{x_2} + x_3) \cdot (x_1 + \overline{x_2} + \overline{x_3}) \cdot (\overline{x_1} + \overline{x_2} + x_3).$$

Запишем это выражение также с использованием обозначения степени:

$$f(x_1, x_2, x_3) = (x_1^1 + x_2^0 + x_3^1) \cdot (x_1^1 + x_2^0 + x_3^0) \cdot (x_1^0 + x_2^0 + x_3^1).$$

Таким образом, можно записать в виде формулы любую логическую функцию, заданную в виде таблицы.

В общем виде переход от табличной формы функции n аргументов $x_1, x_2, \ldots x_n$ к СДНФ (правило записи функции по единицам) можно представить в виде следующего алгоритма:

1. Выбрать те наборы аргументов, на которых $f(x_1, x_2, \ldots x_n) = 1$.
2. Выписать все конъюнкции (логические произведения) для этих наборов. Если при этом x_i имеет значение «1», то этот множитель пишется без инверсии, если «0» – то с инверсией.
3. Все конъюнктивные члены соединить знаком дизъюнкции (логического сложения).

Аналогично можно записать алгоритм перехода от табличной формы задания функции к СКНФ (правило записи функции по нулям).

1. Выбрать те наборы аргументов, на которых $f(x_1, x_2, \ldots x_n) = 0$.
2. Объединить дизъюнкцией логические переменные. Если при этом x_i имеет значение «0», то переменная остается без изменений. Если «1», то она берется с отрицанием.
3. Все дизъюнктивные члены соединить знаком конъюнкции (логического умножения).

Способ записи СДНФ по СКНФ и обратно

В предыдущем разделе из табл. 5.5 были получены две записи одной и той же функции – СКНФ и СДНФ:

$$f(x_1, x_2, x_3) = x_1^0 \cdot x_2^0 \cdot x_3^0 + x_1^0 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^0 \cdot x_3^0 + x_1^1 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^1 \cdot x_3^1 =$$
$$= (x_1^1 + x_2^0 + x_3^1) \cdot (x_1^1 + x_2^0 + x_3^0) \cdot (x_1^0 + x_2^0 + x_3^1)$$

Таким образом, видно, что общее число членов в этих двух формах равно сумме нулей и единиц функции, то есть равно 2^n. Если в исходной форме функции, записанной в СКНФ или СДНФ, содержится z членов, то в другой ее форме (т. е. СДНФ или СКНФ) их будет $(2^n - z)$.

Покажем на примере рассмотренной функции, как можно перейти от одной формы записи к другой. Пусть дана СДНФ функции f из табл. 5.5. Для того чтобы получить ее эквивалентную запись, воспользуемся следующим приемом. Найдем инверсию функции f, записанную в таблице: $F = \overline{f}$. Для этого нужно заменить значения «0» на «1», а «1» на «0». СДНФ для нее будет состоять из трех членов, 010, 011, 101. Это все недостающие до 2^n члены, причем их легко определить по степеням, записанным в СДНФ для функции f.

$$F(x_1, x_2, x_3) = \overline{f(x_1, x_2, x_3)} = x_1^0 \cdot x_2^1 \cdot x_3^0 + x_1^0 \cdot x_2^1 \cdot x_3^1 + x_1^1 \cdot x_2^0 \cdot x_3^1 .$$

Для того чтобы получить из инверсии саму функцию f, от суммы этих членов нужно взять инверсию. Далее, пользуясь правилами де Моргана, получим выражение для эквивалентной СКНФ:

$$f(x_1, x_2, x_3) = x_1^0 \cdot x_2^0 \cdot x_3^0 + x_1^0 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^0 \cdot x_3^0 + x_1^1 \cdot x_2^0 \cdot x_3^1 + x_1^1 \cdot x_2^1 \cdot x_3^1 =$$
$$= \overline{x_1^0 \cdot x_2^1 \cdot x_3^0 + x_1^0 \cdot x_2^1 \cdot x_3^1 + x_1^1 \cdot x_2^0 \cdot x_3^1} = (x_1^1 + x_2^0 + x_3^1) \cdot (x_1^1 + x_2^0 + x_3^0) \cdot (x_1^0 + x_2^0 + x_3^1)$$

Аналогично можно перейти от СКНФ к СДНФ.

Составление СДНФ и СКНФ необходимо при проектировании (синтезе) цифровых схем, выполняющих ту или иную логическую функцию. Следующей основной задачей при синтезе цифровых схем является минимизация логических функций (в результате получают минимальные ДНФ или КНФ, МДНФ или МКНФ). Чем проще логическое выражение, описывающее функцию, тем проще и дешевле будет схема. Метод минимизации может основываться только на тождественном преобразовании логических выражений.

Наконец, конечной целью проектирования является построение схемы устройства.

Контрольные вопросы и задания

1. Дайте определение минтерма и макстерма.
2. Как в виде формулы представить логическую функцию, записанную в таблице?
3. Что такое СДНФ и СКНФ?
4. Составьте таблицу истинности функции:
   - $F = ABC + \overline{A}BC + AC\overline{D}$
   - $F = \overline{A}BC + \overline{A}\overline{B}C + A\overline{D}$
   - $F = A \to B + B \to C$
5. Упростите логические выражения:
   - $(A \to B)(A\overline{C} \vee \overline{B})A\overline{C}$
   - $(\overline{A} \to B)(B\overline{C} \vee \overline{B})AB$
   - $(\overline{A} \to B)(AC \vee B)\overline{A}C$
   - $(\overline{A} \to B)(A\overline{C} \vee \overline{B})AC$
   - $(A \to \overline{B})(AC \vee B)B\overline{C}$
   - $(AB \vee \overline{A}) \to \overline{(A \vee B)}$
   - $(A \to B)(\overline{AB} \to \overline{A}B)$
   - $((B \vee \overline{(A \vee B)}) \to \overline{AB})AB$
6. Какие из ниже перечисленных логических формул являются тождественно-истинными (тавтологиями)?
   - $AB \to (A \vee (B \leftrightarrow C))$
   - $(A \leftrightarrow C) \to (C \vee \overline{A \vee B})$
   - $(B \leftrightarrow C) \to (C \vee \overline{AB})$
   - $\overline{ABC} \to (A \vee B)$
   - $BC \to (A \vee (B \oplus C))$
7. Заданы логические функции $F_1 = \overline{x}_1 x_2 \overline{x}_3 \vee x_1 x_2 \overline{x}_3 \vee x_1 x_2$ и $F_2 = (x_1 x_2 \vee \overline{x}_3 x_2)(\overline{\overline{x}_1 \overline{x}_2 \vee x_2 x_3} \vee x_1 x_3)$. Путем тождественных преобразований получите минимальную форму записи функций и проверьте, является ли функция F_2 тождественной функции F_1.
8. Заданы логические функции $F_1 = \overline{x}_1 \vee x_2$ и $F_2 = \left(\overline{\overline{x}_3 x_1} \vee \overline{\overline{x}_3 x_2}\right)\left(\overline{x_1 \overline{x}_2} \vee x_1 \overline{x}_3 \vee x_2 \overline{x}_3\right)$. Необходимо:
   а) путем тождественных преобразований получить минимальную форму записи функций;
   б) проверить, является ли функция F_2 тождественной функции F_1.
9. Является ли тождественно-истинной данная формула:
   - $(AB \to C) \leftrightarrow (A\overline{C} \to \overline{B})$.
   - $(A \to (B \vee C)) \leftrightarrow ((A \to B) \vee (A \to C))$?

Лекция 6. ЛОГИЧЕСКИЕ СИГНАЛЫ И ЛОГИЧЕСКИЕ МИКРОСХЕМЫ

Сферы применения компьютера

В профессиональной деятельности в первую очередь компьютер предназначен для вычислений. С этой задачей он справляется весьма успешно. Тем не менее при наличии соответствующего оборудования (и это важно для исследователей) компьютер может быть использован как средство активного вмешательства в эксперимент. Он может не только запоминать данные эксперимента и обрабатывать их, но и влиять на проведение эксперимента. Для этого его надо снабдить соответствующими устройствами. К таким устройствам относятся аналого-цифровой и цифроаналоговый преобразователи, приборный интерфейс и др. Эти устройства обеспечивают преобразование аналогового сигнала в понятный компьютеру цифровой и обратно, передают команды управления.

Компьютер как средство обработки информации

Информация является одним из важнейших понятий современного мира. Под информацией понимают совокупность сведений о материальном мире и происходящих в нем процессах, являющихся объектом хранения, передачи или преобразования. С середины XX века в связи с разворачивающейся научно-технической революцией остро встал вопрос о природе и сущности информации, методах ее измерения. Первым, кто более или менее обстоятельно ответил на вопрос о количестве информации, был американский ученый Клод Шеннон. В 1948 году он опубликовал статью «Математическая теория связи», в которой количество информации определялось как уменьшение неопределенности. Информация может храниться только в материальных телах в виде соответствующего описания запоминаемого события. Передается информация при помощи *сигналов*, среди которых можно выделить аналоговые, дискретные и цифровые (см. рис. 6.1).

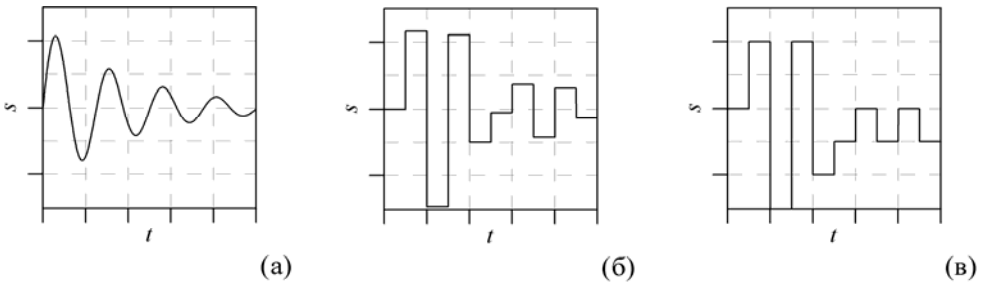


Рис. 6.1. Виды сигналов: непрерывный или аналоговый (*а*), дискретный по времени, но непрерывный по уровню (*б*) и цифровой (*в*)

Аналоговыми называются сигналы, в которых изменение физической величины, используемой для передачи информации, происходит непрерывно. Вследствие этого они подвержены влиянию внешних шумов и помех особенно сильно. Например, аналоговый электрический сигнал, снимаемый с термопары, несет информацию об изменении температуры, сигнал с микрофона – о быстрых изменениях давления в звуковой волне, электроэнцефалограмма – об электрических колебаниях в мозге.

Дискретным называется сигнал, описываемый дискретной функцией времени. Дискретный сигнал образуется из аналогового путем квантования по времени или одновременно по времени и уровню. Часто дискретные сигналы используются для представления знаков систем кодирования (например, двоичной системы счисления). В этом случае их называют *цифровыми* сигналами.

Для двоичной системы счисления достаточно последовательно передавать всего две цифры (ноль и единица), из которых можно составлять слова, представляющие собой последовательности нулей и единиц. В области существования цифровых сигналов введены специальные зоны, значения сигналов в которых либо равнозначны, либо не могут существовать вообще. Правда, это не означает, что физическая величина (например, ток или напряжение) не существует в запрещенной зоне или не отличается по величине друг от друга. При этом для всех, кто пользуется таким цифровым сигналом, вводится соглашение, в соответствии с которым уровень сигнала полагается «низким» или «высоким», нулем или единицей.

При обработке аналогового сигнала используются средства аналоговой электроники (усилители, фильтры, умножители и др.). Цифровые сигналы используются в цифровых устройствах, и в том числе в компьютерах. Для цифровой обработки аналогового сигнала необходимо его преобразовать сначала в цифровую форму, а после обработки – обратно в аналоговую (см. рис. 6.2). Именно так поступают при обработке звука, видео, физиологических данных. Цифровые сигналы широко используются в системах связи, поскольку они менее подвержены воздействию шумов.

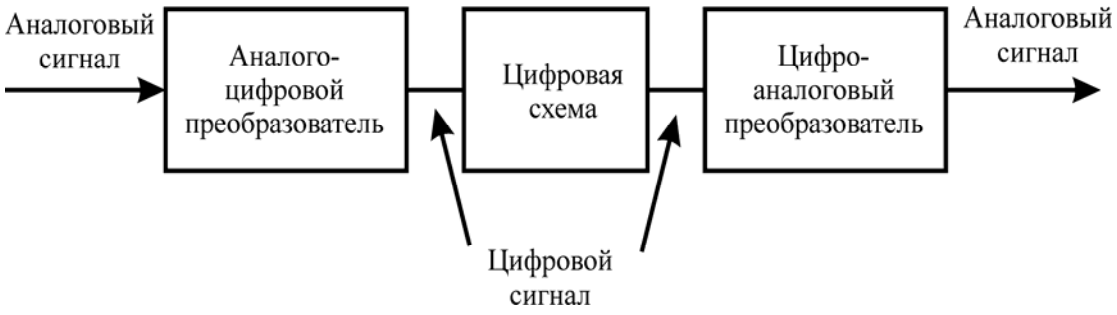


Рис. 6.2. Обработка аналогового сигнала цифровыми методами

Логические микросхемы

Логическими называются микросхемы, работающие с логическими (двоичными, цифровыми) сигналами и выполняющие логические функции. Они вырабатывают двоичные выходные сигналы в ответ на определенные двоичные входные сигналы. К этому классу относятся микросхемы от простейших наборов логических элементов до больших микросхем памяти, микроконтроллеров и микропроцессоров. В этой лекции будут рассмотрены сигналы, с которыми работают логические микросхемы, а также способы реализации простейших логических функций и схем на их основе.

Параметры цифрового сигнала

Вообще говоря, количество разрешенных и запрещенных зон может быть произвольным и определяться принятой системой счисления. При этом каждой из разрешенных зон условно присваивается значение некоторой цифры.

Однако соглашение о представлении цифровой информации и реализация этого соглашения в технических устройствах не одно и то же. Дело в том, что физические процессы в электронных приборах протекают непрерывно и транзистор «не знает», что он должен реализовать некоторое цифровое представление. Отсюда следуют ограничения на количество зон при цифровой форме представления информации. При современном уровне развития электроники компромисс между желаемым в возможным реализован на двоичном представлении информации.

При двоичном представлении информации существуют две разрешенные зоны, которым могут быть присвоены значения логического «0» и «1», вообще говоря, произвольно (существуют, конечно, некоторые схемные ограничения, но о них поговорим ниже), и одна запрещенная зона, заключенная между «дном» зоны высоких уровней и «потолком» зоны низких уровней. Размеры этих зон определяются при конкретной технической реализации устройства. Например, для устройств на ТТЛ-микросхемах (транзисторно-транзисторная логика) напряжение питания составляет 5 В, логическим нулем считают уровень напряжения $U^0_{вых.}$ не более 0.4 В, а логической единицей – напряжение $U^1_{вых.}$ не менее 2.4 В. Эти параметры относятся к статическим параметрам микросхем.

Далее отметим еще одно очень важное обстоятельство. Некоторое цифровое устройство должно не только формировать двоичные уровни, но и распознавать их при приеме. А это означает, что в соглашение о цифровой форме представления информации необходимо ввести еще одну зону – зону порогового напряжения – некоторого напряжения, величина которого не зависит от величины логических уровней (в принятой технической реализации). Конкретная величина порогового напря-

жения определяется схемой цифрового элемента и характеристиками полупроводниковых приборов, используемых в нем.

Как правило, превышение порогового напряжения (при «нулевом» входном напряжении) приводит к отпиранию транзистора формирователя уровня в логическом элементе (отпирающая помеха). При «единичном» входном напряжении соответственно возможна запирающая помеха. Следовательно, с введением порогового напряжения запрещенная зона распадается на две, может быть неравные, зоны: зону отпирающих и зону запирающих помех. Величина этих зон определяется заданной помехоустойчивостью – способностью логического элемента достоверно различать высокие и низкие уровни сигналов при наличии помех. Можно говорить о статической и динамической помехоустойчивости, имея в виду, что при определении величины статической помехоустойчивости не учитываются временные параметры помехи (время действия помехи существенно больше времени переключения цифрового элемента). Если время действия помехи сравнимо со временем переключения цифрового элемента, говорят о динамической помехоустойчивости.

Изменение напряжений на входе и выходе некоторого инвертирующего цифрового элемента приведено на рис. 6.3. Ниже приводятся определения основных динамических параметров. Эти времена связаны с длительностью переходных процессов, происходящих при переключениях в логических схемах.

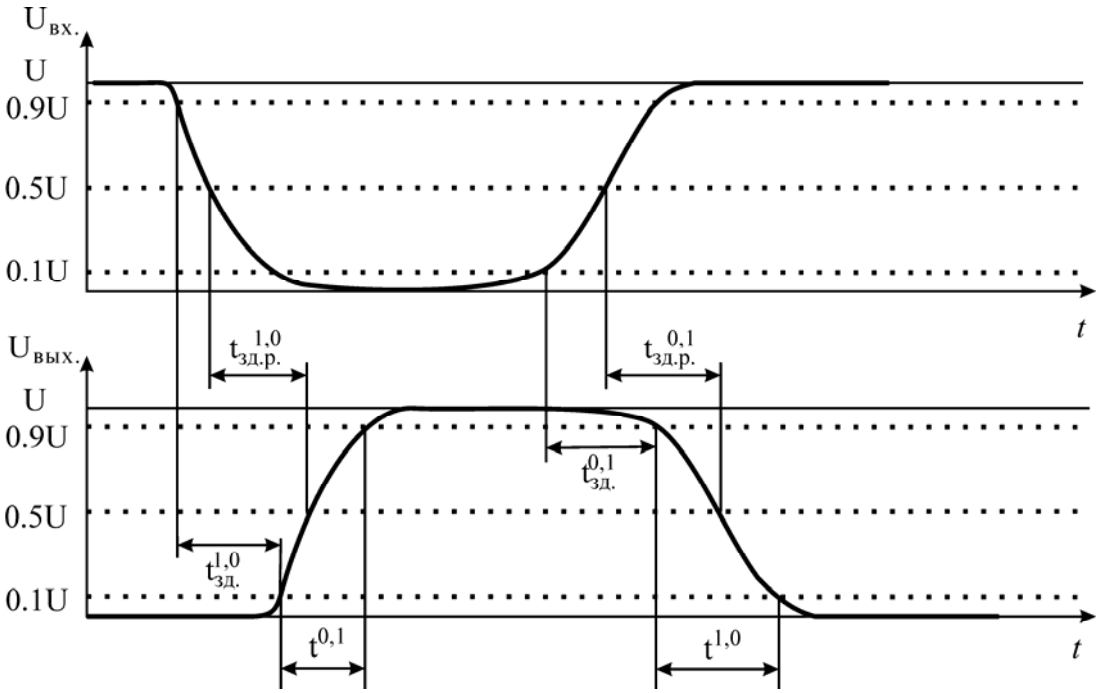


Рис. 6.3. Динамические параметры логического элемента

Время $t^{1,0}$ – время перехода элемента из состояния «1» в состояние «0» – интервал времени, в течение которого напряжение на выходе

элемента переходит от уровня «1» к уровню «0», измеренных при значениях 0.9 и 0.1 логического перепада U (разности напряжения «1» и напряжения «0»). Уровни 0.9 и 0.1 приняты для исключения особенностей (нехарактерных уровней напряжения, например, затухающие колебания) вблизи плато (плоского участка) сигнала.

Другие параметры – время $t^{0,1}$ – это время перехода из состояния «0» в состояние «1», измеренных при значениях 0.9 и 0.1 логического перепада U; $t_{зд.}^{1,0}$ – время задержки выключения элемента, интервал времени между входными и выходным сигналами при переходе напряжения на выходе элемента от уровня «1» к уровню «0». Время задержки $t_{зд.}^{0,1}$ включения элемента – интервал времени между входным и выходным сигналами при переходе напряжения на выходе элемента переходит от уровня «0» к уровню «1». Время $t_{зд.р.}^{1,0}$ задержки распространения сигнала при выключении логического элемента – интервал времени между входными и выходными сигналами при переходе напряжения на выходе элемента от напряжения «1» к «0», измеренный на уровне 0.5 логического перепада входного и выходного сигналов. Время $t_{зд.р.}^{0,1}$ задержки распространения сигнала при включении логического элемента – интервал времени между входным и выходным сигналами при переходе напряжения на выходе элемента от напряжения «0» к «1», измеренный на уровне 0.5 логического перепада входного и выходного сигналов.

Помимо вышеприведенных «стандартных» определений динамических параметров цифровых элементов, часто используются термины «время включения» и «время выключения», которые подразумевают полное время формирования соответственно низкого и высокого уровней выходных напряжений. Усредненным параметром быстродействия служит среднее время задержки распространения $t_{зд.р.ср.} = 0.5(t_{зд.р.}^{0,1} + t_{зд.р.}^{1,0})$. Этот параметр используется при расчете временных характеристик последовательно включенных цифровых микросхем.

Потенциальные и импульсные сигналы

Сигнал называется *потенциальным*, если интервал времени между соседними изменениями сигнала значительно больше времен включения-выключения и времен задержки составляющих элементов схемы, в которой он наблюдается.

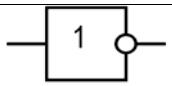
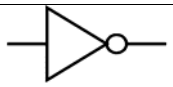
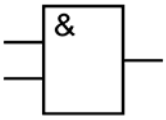
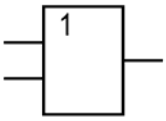
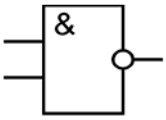
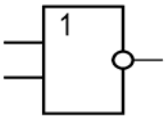
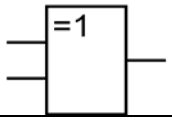
Сигнал называется *импульсным*, если длительность его активного уровня того же порядка, что и время реакции схемы. Схема реагирует на импульсный сигнал, и он должен закончиться сразу после окончания переходного процесса в схеме. При аналитическом описании схем, на которые воздействуют импульсные сигналы, используется понятие абстрактного импульсного сигнала, длительность которого считается бесконечно малой. Реальные импульсные сигналы всегда имеют конечную длительность, которая определяется временем реакции схемы. От техно-

логии изготовления и физических параметров отдельных элементов зависит время реакции схемы на воздействие. Понятие абстрактного импульсного сигнала позволяет абстрагироваться от физических параметров конкретных схем. Реальный импульсный сигнал порождается изменением потенциального сигнала из 1 в 0 или (и) из 0 в 1.

Базовые логические элементы

Все устройства цифровой электроники могут быть изготовлены с использованием простейших элементов, выполняющих нужные элементарные функции. Их называют базовыми логическими элементами (или вентилями). В табл. 6.1 приведены условные графические обозначения базовых логических элементов, соответствующие различным стандартам. В первом столбце показаны обозначения в соответствии с ГОСТ, который применим в России. Похожий стандарт (DIN) используется в странах Европы. Второй столбец соответствует стандарту ANSI, разработанному Американским национальным институтом стандартизации и применимому в Америке.

Таблица 6.1. Условные обозначения и выполняемые логические функции базовых элементов

Условное обозначение		Логическая функция
ГОСТ	ANSI	
		НЕ
		И
		ИЛИ
		И-НЕ
		ИЛИ-НЕ
		Исключающее ИЛИ (сумма по модулю 2)

Входы (логические переменные) в схеме нарисованы слева, а выходы (логические функции) – справа. Обозначения внутри прямоуголь-

ников соответствуют выполняемой логической функции. Кружочек означает инверсию сигнала.

Элементная база цифровых устройств состоит из цифровых микросхем различной степени сложности, от самых простых, содержащих базовые логические элементы, до сложнейших микропроцессоров и других специализированных микросхем. Внутренняя структура микропроцессора может быть представлена в виде логической схемы, отражающей логику работы устройства. Тем не менее собрать современный микропроцессор из отдельных элементов – задача практически неразрешимая, поскольку количество вентилей составляет миллионы штук. Но с другой стороны, знание основных принципов работы цифровых схем на логических элементах во многом проясняет функционирование вычислительных систем и других технических средств информатики.

Схемотехника логических элементов

В зависимости от технологии изготовления интегральные схемы подразделяются на серии, различающиеся физическими параметрами базовых элементов, а также числом и функциональным назначением входящих в их состав микросхем. В настоящее время разработано несколько десятков технологий изготовления микросхем. К ним относятся: диодно-транзисторная логика (ДТЛ), транзисторно-транзисторная логика (ТТЛ), эмиттерно связанная логика (ЭСЛ), логика на МДП (металл – диэлектрик – полупроводник) транзисторах и на КМДП (комплементарных МДП-транзисторах).

Основной электронной схемой в микроэлектронике является электронный вентиль, или ключ на транзисторе (это может быть биполярный или полевой транзистор). Самая простая логическая операция (инверсия, операция НЕ) реализуется схемой, которая преобразует высокий потенциал на входе в низкий потенциал на выходе, и наоборот.

Вообще говоря, *транзистор* – это электронный прибор, предназначенный для усиления и коммутации (переключения) сигналов и имеющий три или более выводов. По принципу действия транзисторы разделяют на биполярные и униполярные. К биполярным относятся транзисторы, в которых используются носители зарядов двух типов – основные и неосновные, положительные и отрицательные. В униполярных транзисторах используются носители только одного знака. Их называют также полевыми транзисторами. Обозначения некоторых транзисторов приведены на рис. 6.4. В биполярном транзисторе (рис. 6.4, *а*) база является управляющим электродом и при подаче положительного напряжения на базу относительно эмиттера (для n-p-n транзистора) проводимость коллектор-эмиттер увеличивается. В полевом транзисторе также есть управляющий электрод – затвор. Изменение управляющего напряжения на нем позволяет регулировать проводимость транзистора.

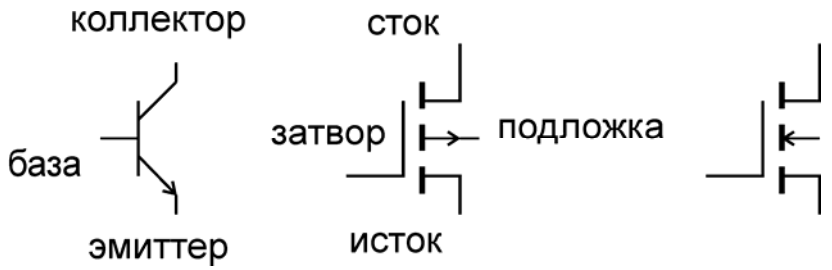


Рис. 6.4. Графическое обозначение биполярного n-p-n транзистора (*а*), графическое обозначение полевого транзистора с изолированным затвором и каналом p-типа (*б*) и полевого транзистора с каналом n-типа (*в*)

Рассмотрим более подробно работу полевого транзистора с изолированным затвором и индуцированным каналом n-типа (см. рис. 6.5). На рисунке показан массив полупроводника, легированный примесями p-типа (подложка), в котором технологическими приемами выполнено легирование примесями n-типа. Черным показаны металлические электроды, под затвором – диэлектрик. Действие транзистора заключается в следующем. Электрическое поле, создаваемое положительным потенциалом затвора, индуцирует соответствующий отрицательный заряд в массиве полупроводника, который отделен от затвора диэлектриком и служит второй пластиной конденсатора. Таким образом, между стоком и истоком образуется слой с проводимостью типа *n*. Через этот индуцированный канал может протекать ток от стока к истоку. Чем больше напряжение на затворе, тем больше поперечное сечение индуцированного канала и тем меньше его сопротивление. При подаче напряжения питания изменение напряжения на затворе позволяет регулировать ток сток-исток.

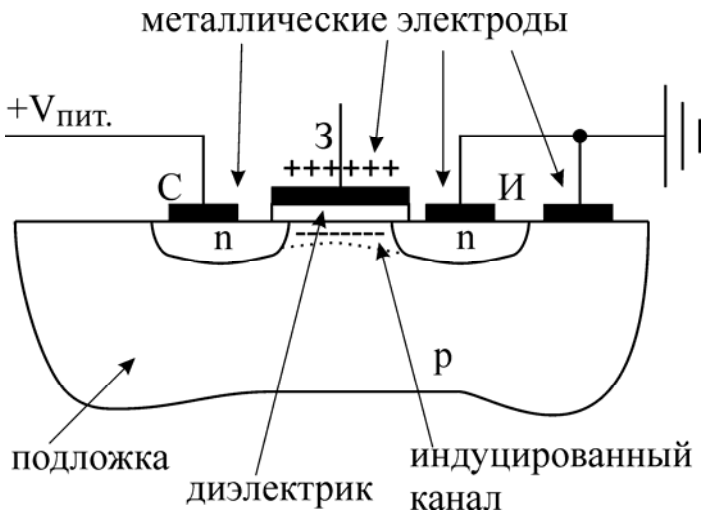


Рис. 6.5. Устройство полевого транзистора с изолированным затвором и индуцированным каналом n-типа

Аналогичным образом работает полевой транзистор с каналом p-типа. Отличие в том, что этот транзистор начинает пропускать ток при подаче отрицательного напряжения на затвор. Пара таких транзисторов с близкими параметрами называется комплементарной парой. «Комплементарный» в переводе с английского значит «дополняющий».

В данном курсе мы будем рассматривать работу транзисторов в ключевом режиме. Это означает, что транзистор работает как ключ (т. е. выключатель). При подаче управляющего напряжения на базу (затвор) транзистора он открывается (при этом сопротивление его низкое), при снятии управляющего напряжения транзистор закрывается (сопротивление высокое). Это свойство транзистора используется в схеме простого инвертора на рис. 6.6. Такое приближение вполне допустимо, поскольку в цифровой технике используются цифровые сигналы, у которых определены только два уровня – низкий и высокий. Рассмотрим работу этой схемы при различных величинах входного напряжения. При управляющем напряжении, меньшем порогового (рис. 6.6, *а*), транзистор закрыт, напряжение на обкладках конденсатора[3] высокое, и заряд накапливается большой. При управляющем напряжении, большем порогового (рис. 6.6, *б*), транзистор открыт, напряжение на конденсаторе невелико и заряд на обкладках конденсатора мал.

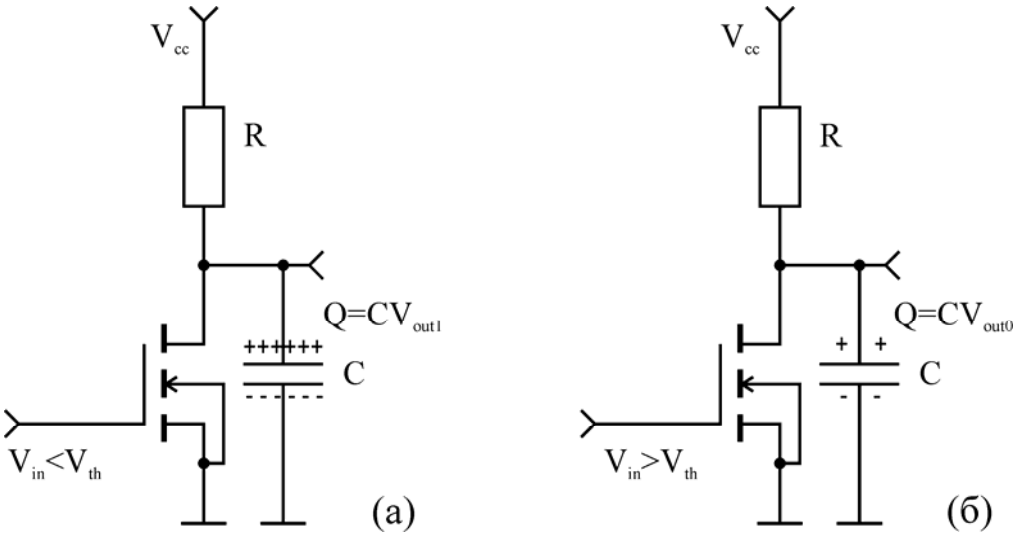


Рис. 6.6. Схема электронного ключа (простого инвертора)

Мы можем сопоставить высокому напряжению логическую единицу (хотя для реальных схем это не всегда так), а низкому напряжению – логический ноль. Разность потенциалов $\Delta V = V_{out1} - V_{out0}$ называется *логическим перепадом*.

[3] Конденсатор *C* в схеме отражает наличие емкости соединительных цепей и входов последующих каскадов.

Приведенная схема обладает существенным недостатком, который увеличивает потребление энергии и, соответственно, усложняет проблему теплоотвода. Дело в том, что в открытом состоянии по резистору R протекает ток, приводящий к потерям энергии. Снизить потери энергии можно, увеличивая значение R. Но при увеличении R будет снижаться быстродействие, связанное с тем, что в закрытом состоянии конденсатор C должен зарядиться через резистор R.

Для того чтобы избежать подобных недостатков, используют логические схемы на КМДП (эти потери для них практически исключены). КМДП-ключи являются основными элементами микромощной электроники. Интегральные схемы, изготовленные по этой технологии, – самые экономичные по расходу электроэнергии и (что особенно важно) по тепловыделению в процессе работы. Поэтому КМДП интегральные схемы завоевали ведущее положение в цифровой электронике: в схемах оперативной памяти и в конструкциях процессоров как персональных компьютеров, так и больших ЭВМ.

Простейший инвертирующий элемент КМДП ИС образован двумя полевыми транзисторами: один с каналом *p*-типа, другой с каналом *n*-типа (см. рис. 6.7). Рассмотрим более подробно работу этого устройства. Пусть $V_{вх}$ близко к $V_{пит}$ (на входе логическая 1). Тогда *n*-канал открыт, *p*-канал закрыт. На выходе $V_{вых}$ практически равно 0 (на выходе логический 0). Если $V_{вх}$ близко к 0 (на входе логический 0), тогда *n*-канал закрыт, *p*-канал открыт. На выходе $V_{вых}$ близко к $V_{пит}$ (логическая 1). Таким образом, эта схема инвертирует входной сигнал, т. е. если на вход подать логическую 1, то на выходе будет логический 0, и наоборот. Отметим, что потребляемый ток (ток покоя) этой схемы крайне мал, поскольку всегда один из транзисторов закрыт, а другой открыт. Кроме того, входное сопротивление инвертора на КМДП очень велико, поскольку определяется крайне малым током затвора полевого транзистора.

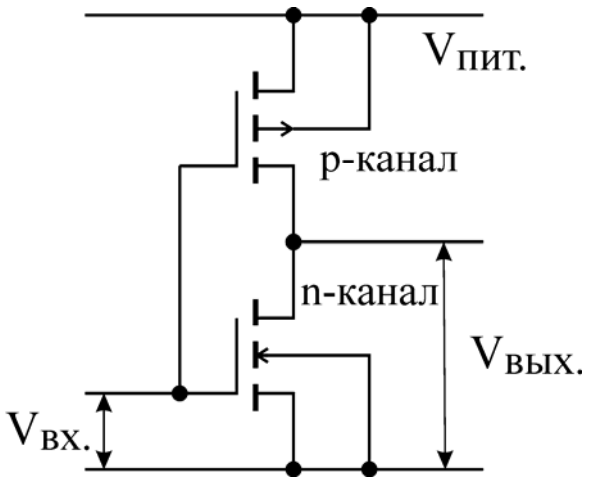


Рис. 6.7. Инвертор на КМДП. Принципиальная схема

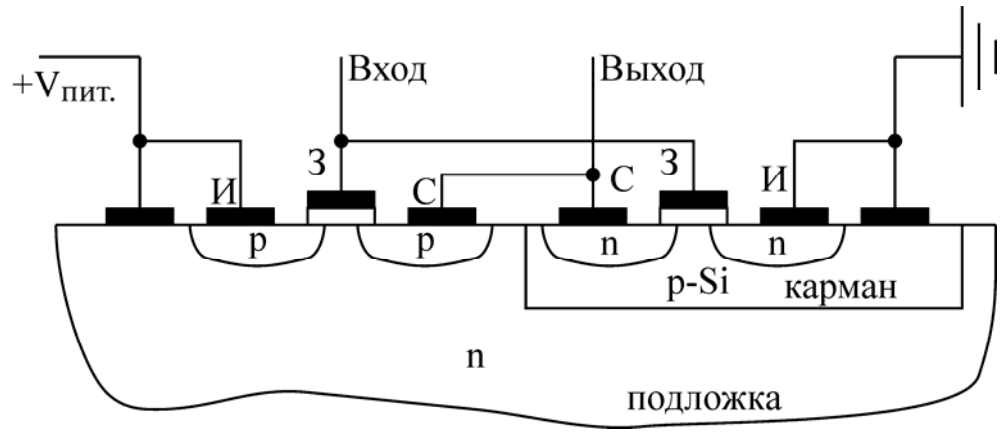


Рис. 6.8. Технологическое изготовление КМДП инвертора

Построение инвертора на КМДП-элементах стало возможным с развитием полупроводниковой технологии. Идея конструктивного решения представлена на рис. 6.8. При таком способе легирования на единой пластине полупроводника можно разместить элементы различной проводимости и, таким образом, построить полноценный инвертор.

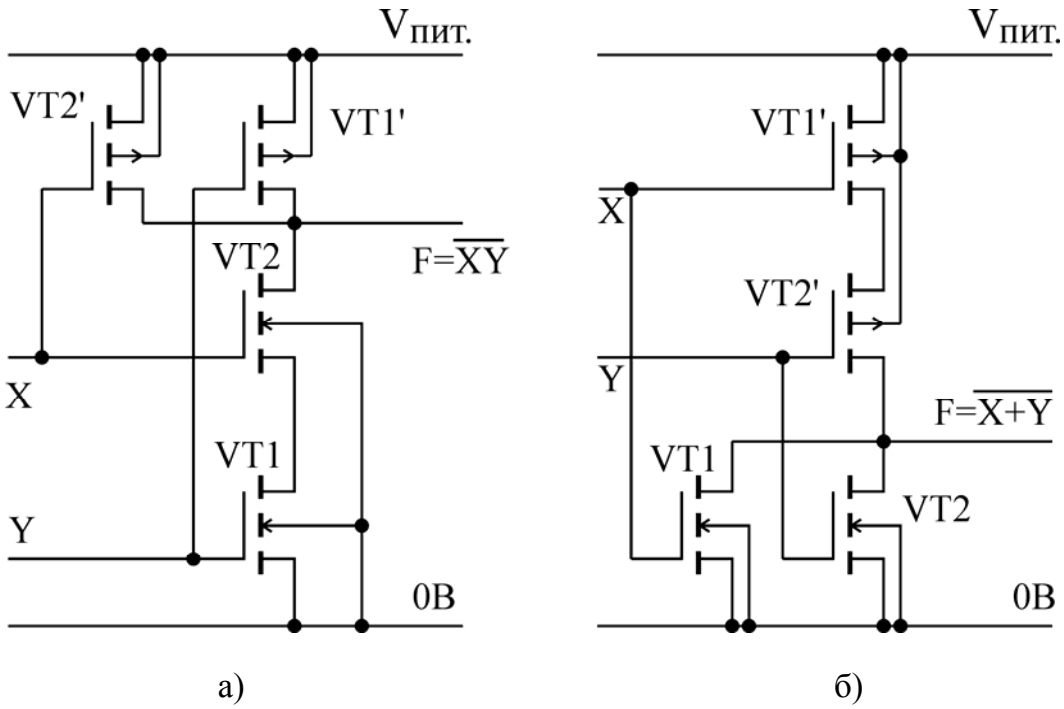


Рис. 6.9. Реализация базового элемента И-НЕ (*а*)
и базового элемента ИЛИ-НЕ (*б*)

Последовательное и параллельное соединение ключей дает возможность реализовать логические функции И-НЕ и ИЛИ-НЕ. При этом количество входов логического элемента может быть более двух. Рассмотрим подробнее работу двухвходового логического элемента, схема которого представлена на рис. 6.9, *а*. Выходное значение переменной *F* будет соответствовать логическому 0 в одном случае – когда транзи-

сторы VT1, VT2 открыты одновременно. При этом на входах X, Y – логическая 1, а значит, транзисторы VT1', VT2' закрыты. Во всех остальных случаях сигнал на выходе соответствует логической 1, поскольку хотя бы один из транзисторов VT1', VT2' открыт, а хотя бы один из транзисторов VT1, VT2 закрыт. Таким образом, в этой схеме реализована логическая функция «штрих Шеффера», или отрицание конъюнкции. Вторая схема (рис. 6.9, *б*) реализует логическую функцию «стрелка Пирса», или отрицание дизъюнкции. Такие схемотехнические решения используются при построении не только небольших интегральных микросхем, но и мощных процессоров и контроллеров.

Построение логической схемы

В соответствии с полученными логическими выражениями МДНФ или МКНФ можно построить логическую схему. При этом полученные логические выражения следует представить в виде комбинации операций, выполняемых элементами базового набора, а затем построить логическую схему. В случае использования базового набора, содержащего элементы И, ИЛИ, НЕ, построение логической схемы производится непосредственно в соответствии с записанной логической функцией. Дизъюнкция двух и более элементов заменяется логическим элементом ИЛИ с двумя или более входами, конъюнкция – элементами И, инверсия – элементами НЕ.

Существует также другая возможность построить логическую схему. Можно использовать элементы только одного типа, выполняющие функцию И-НЕ или функцию ИЛИ-НЕ. Поскольку функция И-НЕ представляет собой полный набор логических функций (то есть используя только эту логическую функцию, можно построить любую другую), то на базе элементов И-НЕ можно построить любую логическую схему. То же самое относится к логическим элементам ИЛИ-НЕ.

При реализации на элементах И-НЕ следует произвести двойную инверсию над полученной ДНФ и преобразовать по теореме де Моргана инверсию дизъюнкций в конъюнкцию инверсий. Например:

$$F = \overline{A}B + B\overline{C}\,\overline{D} + \overline{B}\overline{C}D + A\overline{B}\overline{C}\,\overline{D} = \overline{\overline{\overline{A}B + B\overline{C}\,\overline{D} + \overline{B}\overline{C}D + A\overline{B}\overline{C}\,\overline{D}}} = \overline{\overline{(\overline{A}B)}\,\overline{(B\overline{C}\,\overline{D})}\,\overline{(\overline{B}\overline{C}D)}\,\overline{(A\overline{B}\overline{C}\,\overline{D})}}.$$

Полученное в результате логическое выражение содержит только операции И-НЕ. Его можно реализовать на одном типе элементов (см. рис.6.10). Построение принципиальной схемы устройства, реализующего логическую функцию *F*, производится по следующему алгоритму.

Алгоритм построения логической схемы в базисе И-НЕ.

1. Построить все нужные инверсии логических переменных (этому соответствует левый ряд логических элементов).
2. Построить все элементарные логические перемножения с инверсиями (логические элементы И-НЕ в среднем ряду).

3. При помощи выходного логического элемента И-НЕ перемножить и взять инверсию от элементарных конъюнкций.

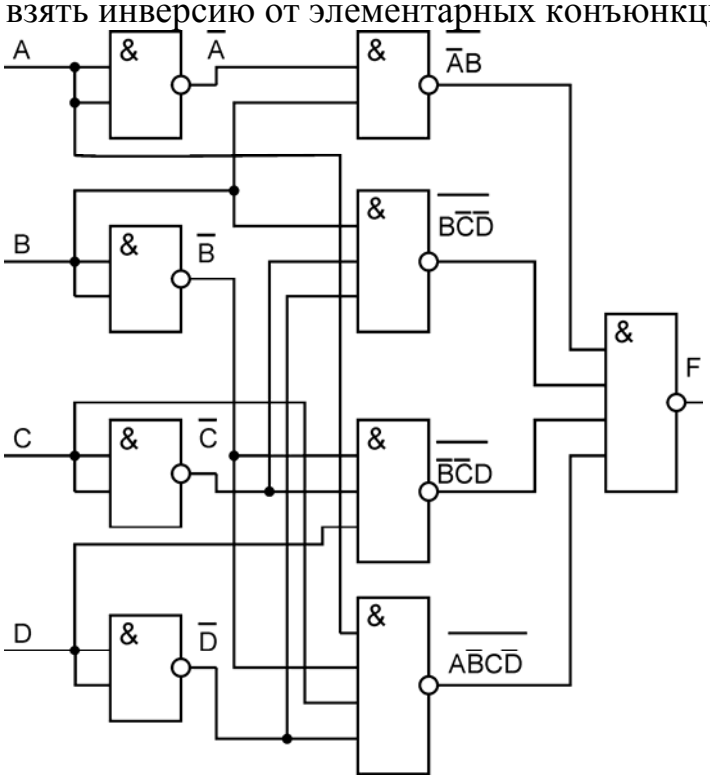


Рис. 6.10. Реализация схемы в базисе И-НЕ

При реализации на элементах ИЛИ-НЕ полученную КНФ необходимо также дважды инвертировать и преобразовать инверсию конъюнкции в дизъюнкцию инверсий.

$$F = (A + \overline{B} + \overline{C})(\overline{C} + \overline{D}) = \overline{\overline{(A + \overline{B} + \overline{C})(\overline{C} + \overline{D})}} = \overline{\overline{(A + \overline{B} + \overline{C})} + \overline{(\overline{C} + \overline{D})}}.$$

В полученном выражении содержатся только операции ИЛИ-НЕ, поэтому такая функция может быть реализована на одних элементах ИЛИ-НЕ (см. рис. 6.11).

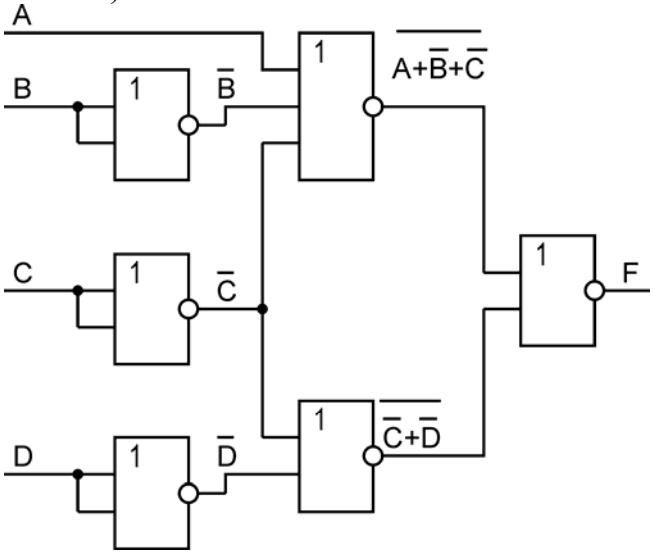


Рис. 6.11. Реализация схемы в базисе ИЛИ-НЕ

Контрольные вопросы и задания

1. Какие виды сигналов вы знаете?
2. Как работает полевой транзистор?
3. Что такое электронный ключ?
4. Как устроены двухвходовые логические элементы И-НЕ, ИЛИ-НЕ?
5. Напишите таблицы истинности для трехвходовых логических элементов И-НЕ, ИЛИ-НЕ.
6. Каков принцип построения логической схемы в базисе И-НЕ? В базисе ИЛИ-НЕ?
7. Представьте функцию $F = \overline{A}BC + \overline{B}\overline{C}D + \overline{B}C\overline{D} + AB\overline{C}$ в виде, удобном для построения в базисе И-НЕ. Нарисуйте соответствующую схему.

Лекция 7. КОМБИНАЦИОННЫЕ ЛОГИЧЕСКИЕ СХЕМЫ. Часть 1

Комбинационной схемой называется логическая схема, выходные сигналы которой описываются системой логических функций

$$z_q = f_q(x_n, \ldots, x_1),$$

где x_p – входные сигналы, $p = 1 \ldots n$, $q = 1 \ldots k$.

Из этого определения следует, что комбинационная схема (КС) реализует однозначное соответствие между значениями входных и выходных сигналов. Базовые логические элементы являются простейшими комбинационными схемами. На основе базовых логических элементов, которые являются элементарными «кирпичиками», строятся более сложные комбинационные схемы. Количество входов и выходов, вообще говоря, может быть произвольным.

При создании компьютеров, периферийных устройств и любых других цифровых систем используются различные комбинационные схемы, поэтому важно знать принципы их построения и выполняемые ими функции. В следующих двух лекциях будут рассмотрены основные типы комбинационных схем, используемые при построении технических средств информатики – компьютеров и периферийных устройств, и рассмотрены возможности их применения.

Дешифратор

Полным дешифратором с прямыми выходами называется комбинационная схема, имеющая n входов и реализующая 2^n минтермов. Таким образом, любой полный дешифратор выполняет функции:

$$f_i = K_i(v) = \prod_{p=1}^{n} x_p^{e_p}, \qquad (7.1)$$

где $v = (x_n, \ldots x_1)$, $i = e_n \ldots e_1$ – двоичное число, $i = 0 \div 2^n - 1$ – соответствующее ему десятичное число. Это дешифратор $n \times 2^n$. В соответствии со свойствами минтермов при каждой комбинации значений входных сигналов x_p только один выход принимает значение, равное логической 1.

Словесный алгоритм работы дешифратора таков. На его вход поступает n-разрядный двоичный код, и в соответствии с этим кодом активируется выход с нужным номером.

Поэтому дешифраторы широко используются в коммутаторах электронных устройств, обеспечивая включение (активизацию) одного устройства на выходе, соответствующего адресу на входе. При помощи дешифратора можно также осуществлять адресацию ячеек памяти.

Примеры построения схем дешифраторов представлены на рис. 7.1. В схеме слева адресный вход один. Если на него подать логический 0, то активным будет 0-й выход, а если логическую 1, то активным будет 1-й выход. У схемы справа два адресных входа. Если на входе 00, то на 0-м выходе логическая 1, на остальных выходах – логический 0. Если на входе 01, то на 1-м выходе логическая 1, а на остальных выходах – логический 0, и т. д.

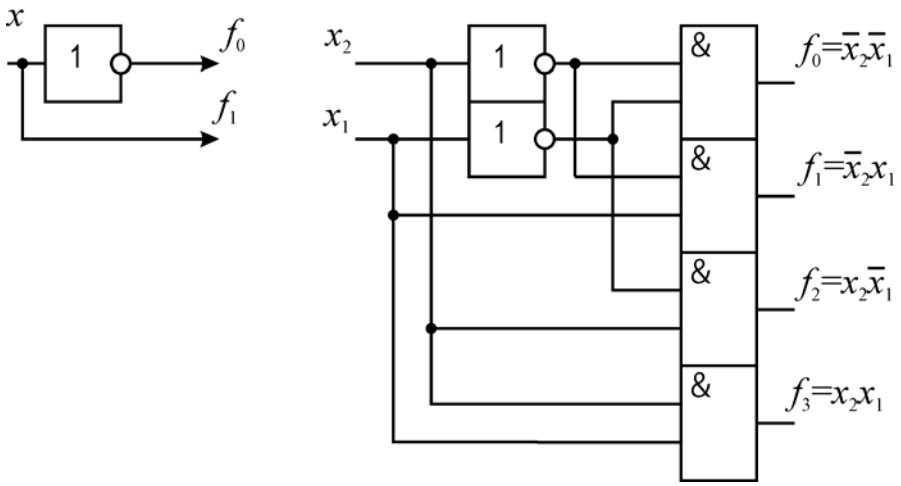


Рис. 7.1. Схемы дешифраторов 1×2 и 2×4

Таблицы истинности выходных функций дешифратора 2×4 приведены в табл. 7.1.

Таблица 7.1. Выходные функции дешифратора 2×4

Входы		**Выходы**			
x_2	x_1	f_0	f_1	f_2	f_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Дешифратор вместе со схемами ИЛИ можно использовать для реализации произвольных логических функций. Это возможно потому, что на выходах дешифратора вырабатываются все возможные минтермы *n* входных переменных. Поскольку логическая функция, представленная в СДНФ, есть дизъюнкция минтермов, то собирая нужные минтермы с помощью элементов ИЛИ, можно получить любую функцию, и не только одну, а несколько.

На практике часто используются так называемые неполные дешифраторы. Неполным дешифратором называется комбинационная схема, имеющая *n* входов и реализующая $N < 2^n$ минтермов *n* переменных (см. рис. 7.2). В условном обозначении дешифраторов в поле основного прямоугольника находится обозначение DC (от англ. DeCoder).

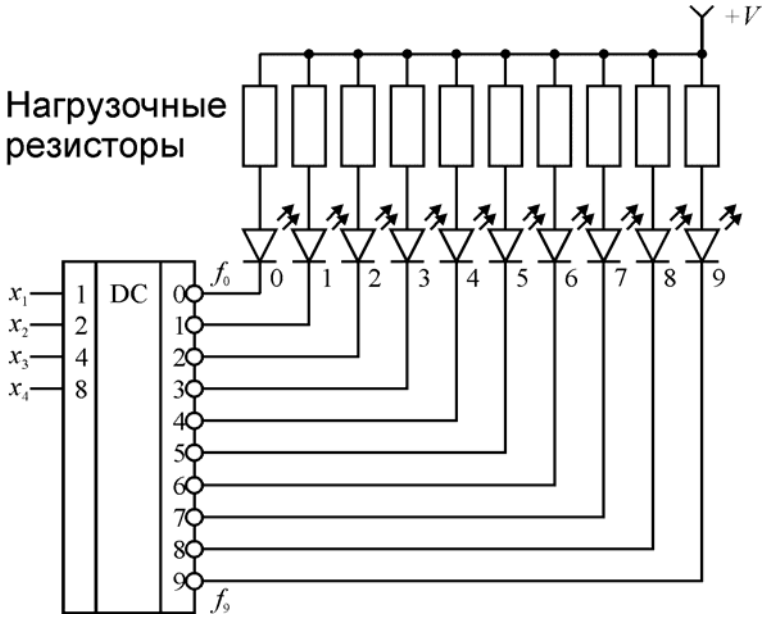


Рис. 7.2. Неполный дешифратор (условное обозначение и применение для индикации)

Входы микросхемы-дешифратора (x_1–x_4) называются адресными входами, поскольку их сигналы характеризуют номер (адрес) активного выхода. Выходы дешифратора могут быть прямыми (в этом случае на активном выходе установлена логическая 1) и инверсными (на активном выходе установлен логический 0). Инверсные выходы обозначаются кружочками, как и в логических элементах с инвертированием выхода.

Неполные дешифраторы 4×10 также являются весьма полезными схемами. Одно из их применений – это отображение информации на цифровом индикаторе. Каждый выход дешифратора зажигает десятичную цифру индикатора (см. рис. 7.2). При поступлении на вход кода 0000 загорается светодиод, отображающий цифру 0, при поступлении кода 0001 – светодиод с цифрой 1, и т. д. Для того чтобы отображать многоразрядное число, необходимо несколько таких дешифраторов. Многоразрядные индикаторы часто используются в цифровых измерительных приборах.

Демультиплексоры

Демультиплексор – это комбинационная схема, обеспечивающая переключение одного входа на большое число выходных каналов. Она предназначена для того, чтобы передать данные с одного информационного входа на выход, адрес которого указан на линиях адреса. Принцип действия демультиплексора поясняется на рис. 7.3. Номер выхода, на который необходимо подать информационный сигнал *E*, задается комбинацией логических сигналов на адресных входах дешифратора. Выход с нужным номером активируется и замыкает соответствующий ключ. На этот выход передается информационный сигнал *E*.

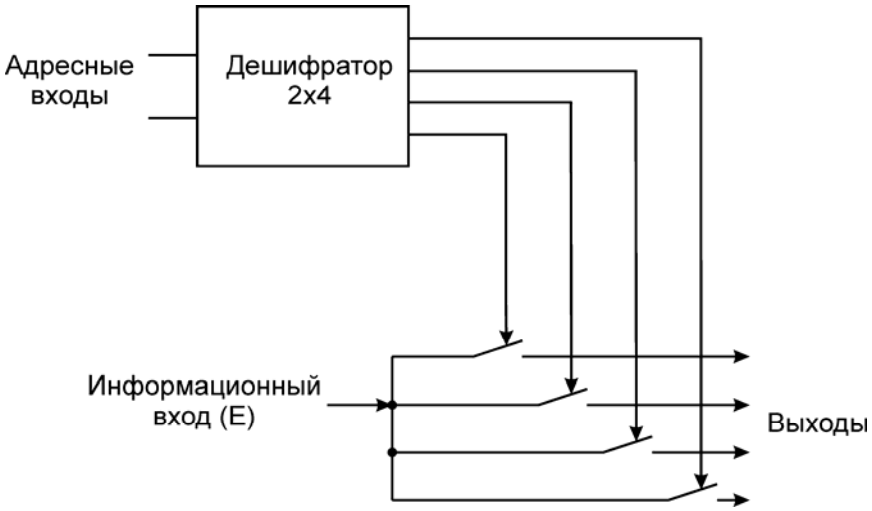


Рис. 7.3. Принцип действия демультиплексора

Математическое определение демультиплексора дает соотношение, описывающее выходные функции вида:

$$f_i = E \cdot K_i(v) = E \cdot \prod_{p=1}^{n} x_p^{e_p}, \qquad (7.2)$$

где E – коммутируемый на один из 2^n выходов сигнал, $v = (x_n, \ldots x_1)$, $i = e_n \ldots e_1$ – двоичное число, $i = 0 \div 2^n - 1$ – соответствующее ему десятичное число. Если $K_i(v) = 1$, то $f_i = E$, а если $K_i(v) = 0$, то $f_i = 0$. Совокупность значений сигналов v определяет адрес (номер) выходного канала, к которому подключен сигнал E. Демультиплексоры, имеющие n адресных сигналов, 1 вход и 2^n выходов, называются демультиплексорами $1 \to 2^n$. Если положить $E \equiv 1$ (сравните полученную формулу с (7.1)), то демультиплексор $1 \to 2^n$ превращается в дешифратор $n \times 2^n$. Информационными входами дешифратора в этом случае являются адресные входы демультиплексора $x_p^{e_p}$.

Рассмотрим, как построить схему демультиплексора $1 \to 2$. Из анализа рис. 7.3 следует, что дешифратор является частью схемы демультиплексора. Из математического определения видно, что i принимает значения 0 или 1, $n = 1$, соответственно в схеме один адресный вход (сравните со схемой дешифратора 1×2), а выходных функций f всего две: $f_0 = E \cdot \bar{x}$ и $f_0 = E \cdot x$.

Таким образом, реализация схема демультиплексора $1 \to 2$ может быть представлена в виде, показанном на рис. 7.4.

Аналогично мы можем построить функции демультиплексора $1 \to 4$. Таких функций четыре:

$$f_0 = E \cdot \bar{x}_2 \cdot \bar{x}_1,$$
$$f_1 = E \cdot \bar{x}_2 \cdot x_1,$$
$$f_2 = E \cdot x_2 \cdot \bar{x}_1,$$
$$f_3 = E \cdot x_2 \cdot x_1.$$

Соответствующая реализация схемы демультиплексора $1 \to 4$ представлена на рис. 7.5.

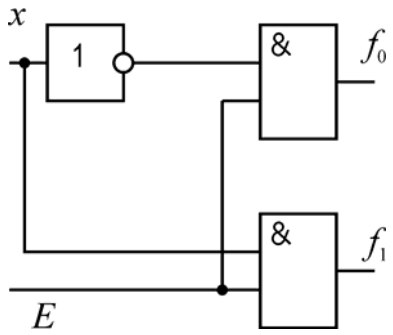


Рис. 7.4. Схема демультиплексора $1 \to 2$

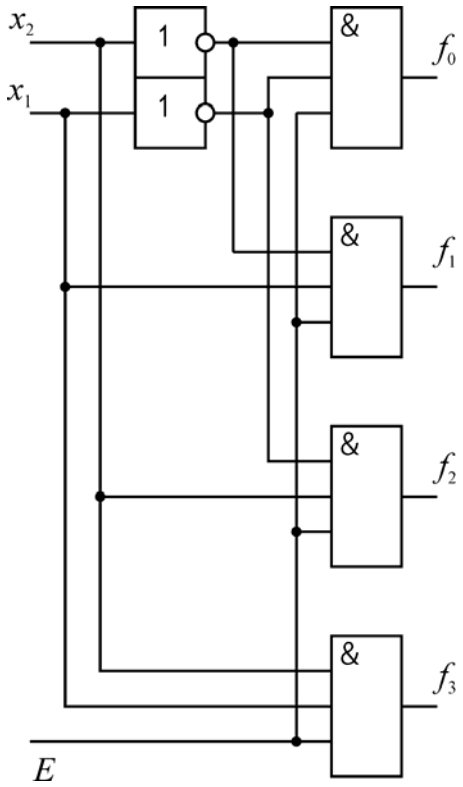


Рис. 7.5. Схема демультиплексора $1 \to 4$:
E – вход информации; x_1, x_2 – адресные входы; f_i – выходы

Если логические элементы И заменить на элементы с инверсией И-НЕ, то получится демультиплексор $1 \to 4$ с инверсными выходами. Такие демультиплексоры изготавливают в виде специальных микросхем. На рис. 7.6 представлено условное обозначение одной из таких микросхем типа К555ИД7.

Сигнал E в этом случае представляет собой конъюнкцию трех сигналов: $E = \overline{E}1 \cdot \overline{E}2 \cdot E3$.

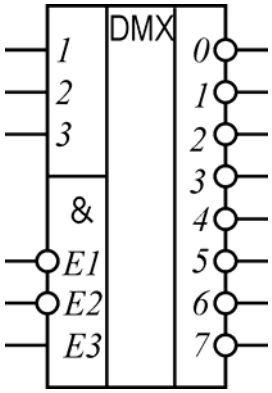


Рис. 7.6. Условное обозначение микросхемы-демультиплексора типа 555ИД7

Мультиплексоры

В цифровых системах одной из важнейших процедур является операция, обеспечивающая подачу цифровых данных из различных линий связи в нужное место. Мультиплексор фактически выполняет функцию, обратную демультиплексору. Если на адресные входы мультиплексора подается цифровой код, то он передает данные со входа, имеющего этот адрес, на единственный выход.

Следуя принципу действия схемы демультиплексора, представленной на рис. 7.3, для реализации мультиплексора достаточно поменять местами информационные входы и выходы. Результат такой замены представлен на рис. 7.7.

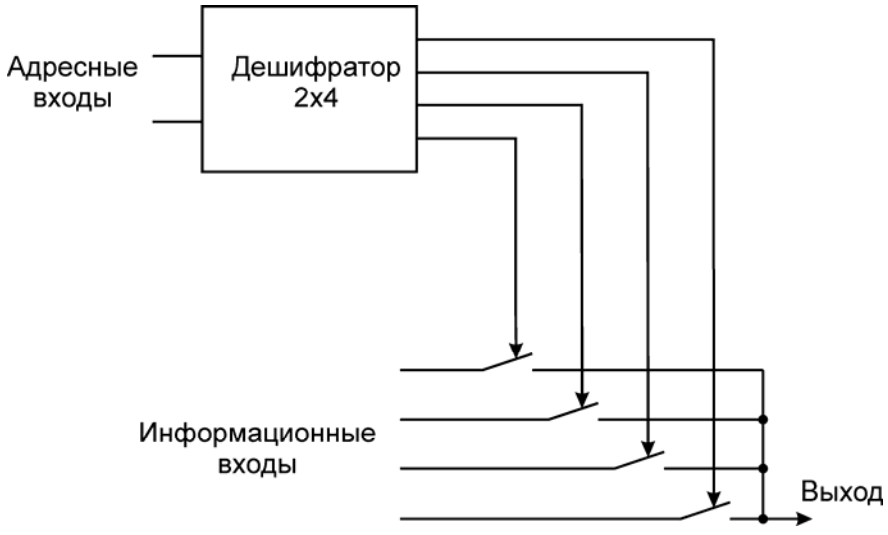


Рис. 7.7. Принцип действия мультиплексора

Более строгое определение мультиплексора таково: комбинационная схема называется мультиплексором, если она выполняет функцию:

$$DO = \sum_{i=0}^{2^n-1} DI_i \cdot K_i(v) = \sum_{i=0}^{2^n-1} DI_i \cdot \prod_{p=1}^{n} x_p^{e_p} , \qquad (7.3)$$

где $v = (x_n,...x_1)$; $i = e_n...e_1$; *DI* – Data Input, информационные входные сигналы, *DO* – Data Output, выходной сигнал.

Мультиплексор является коммутатором 2^n сигналов DI_i на один выход. Действительно, если $K_i(v) = 1$, то $K_j(v) = 0$ при $j \neq i$, и $DO = DI_i$. На рис. 7.8 представлены схемы мультиплексоров $2 \to 1$ и $4 \to 1$.

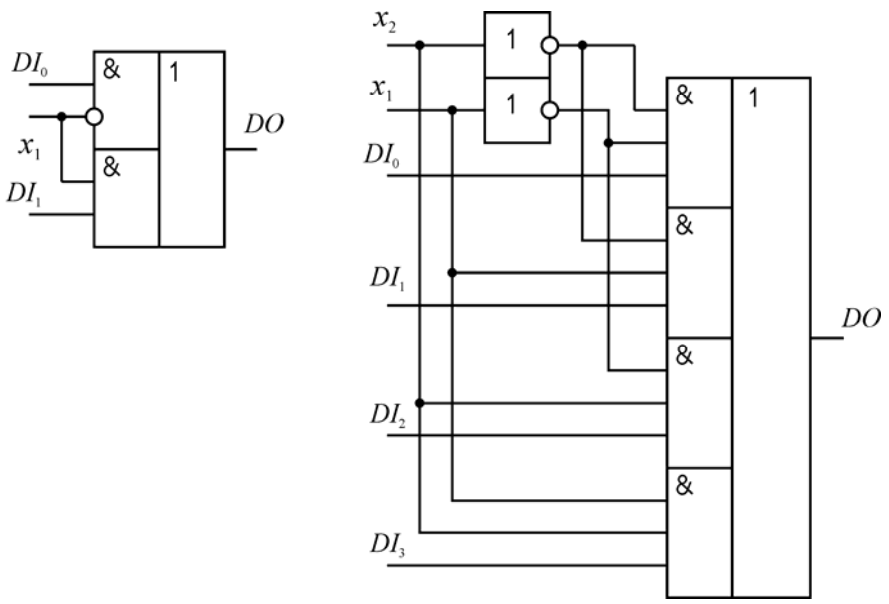


Рис. 7.8. Схемы мультиплексоров $2 \to 1$ и $4 \to 1$, построенные с использованием логических элементов И, ИЛИ, НЕ:
DI (Data Input) – информационные входы; x_1, x_2 – адресные входы;
DO (Data Output) – выход.

На основании уравнения можно построить схему мультиплексора с любым числом адресных входов x_p. Комбинации значений адресных входов определяют номер i информационного входа DI_i, подключенного к выходу *DO*.

Пример условного обозначения мультиплексора представлен на рис. 7.9.

Мультиплексоры могут иметь дополнительный вход управления *E* (*OE*): Enable (Output Enable): $DO = E \cdot \sum_{i=0}^{2^n-1} DI_i \cdot K_i(v)$. Сигнал *E* производит стробирование выхода *DO*, и схема этого мультиплексора может быть получена добавлением еще одного входа у логических элементов И для подачи сигнала *E*.

Мультиплексоры обладают следующим интересным свойством. Функция, выполняемая ими, по структуре совпадает с совершенной дизъюнктивной нормальной формой (СДНФ) представления функций *n* переменных. Из этого следует, что любую переключательную функцию *n* переменных можно реализовать на мультиплексоре $2^n \to 1$, подав на входы DI_i набор констант $a_i = f(v_i) = 0$ или 1.

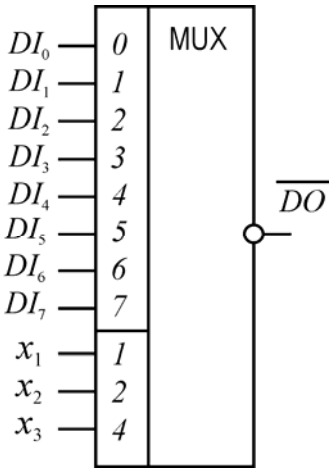


Рис. 7.9. Условное обозначение мультиплексора типа 155КП7

Шифратор

В отличие от дешифратора, который является преобразователем двоичного n-разрядного кода в унарный 2^n-разрядный код (то есть тот, у которого все разряды, за исключением одного, равны 0), шифратор выполняет обратное преобразование. Таким образом, таблица истинности шифратора 4×2 имеет вид, представленный в табл. 7.2. При подаче активного сигнала на вход шифратора I_n на выходе шифратора генерируется код этого входа. Как видно из табл. 7.2, если на входе I_1 установлена логическая 1, то на выходе – двузначный код 01.

Таблица 7.2. Таблица истинности шифратора 4×2

Входы				**Выходы**	
I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Для шифраторов должно выполняться условие $I_i \cdot I_j = 0$ при $i \neq j$. Таким образом, во входных сигналах использованы не все возможные варианты. В ситуациях, когда сигналы поступают от независимых источников, это условие невыполнимо. Такое состояние может возникать при решении задач определения приоритетного претендента на пользование каким-либо ресурсом. Несколько конкурентов выставляют свои запросы на обслуживание, которые не могут быть удовлетворены одновременно. Нужно выбрать такое устройство, которому будет предоставлено право первоочередного обслуживания. В простейшем варианте каждому входу назначается свой приоритет. Считается, что чем больше номер входа, тем выше его приоритет. В этом случае шифратор должен выдавать на выходе двоичный код числа i, если на входе I_i установлена логическая 1, а на все входы I_j, имеющие больший приоритет, поданы

нули. При этом сигналы на входах с меньшим приоритетом значения не имеют. Такие шифраторы называются приоритетными шифраторами. Таблица истинности приоритетного шифратора приведена в табл. 7.3.

Символом *Ф* обозначены состояния, которые не определены, то есть значение этого входа в данной ситуации не важно, и может быть либо логическим 0, либо логической 1.

Таблица 7.3. Таблица истинности приоритетного шифратора 4×2.

Входы				**Выходы**	
I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	1	0	0
0	0	1	Ф	0	1
0	1	Ф	Ф	1	0
1	Ф	Ф	Ф	1	1

Запись передаточной функции для выходов A_1 и A_0 будет иметь следующий вид:

$$A_0 = \bar{I}_3 \bar{I}_2 I_1 + I_3,$$
$$A_1 = \bar{I}_3 I_1 + I_3.$$

При записи этой формулы использовано правило записи СДНФ логической функции по единицам (см. предыдущие лекции). Отличие состоит в том, что неопределенные входы не учитываются, поэтому в результате получается ДНФ, в которой ранг минтермов не обязательно одинаков.

Рассмотрим работу конкретного приоритетного шифратора 8×3 типа 155ИВ1. Его функционирование описывается табл. 7.4, а условное обозначение приведено на рис. 7.10. Назначение сигналов следующее: *E* – сигнал включения шифратора (входной); *G* – выходной сигнал, свидетельствующий о наличии хотя бы одного возбужденного входа I_i при включенном состоянии шифратора; *EO* – выходной сигнал разрешения, он равен 1 при отсутствии возбужденных входов при включенном состоянии шифратора. Из таблицы видно, что 3-разрядный двоичный код $A_2A_1A_0$ можно считывать только при наличии сигнала $G = 1$. Этот сигнал может быть использован в ЭВМ для *запроса на прерывание*, которое подается процессору от внешнего устройства с целью занять время процессора и провести процедуру обработки. Приоритетность шифратора позволяет различить устройства, которые могут подождать (обычно это медленные устройства), и устройства, требующие немедленного обслуживания.

Запишем передаточные функции табл. 7.3 в виде формул. При этом также необходимо пользоваться правилами записи СДНФ, но те переменные, от которых функция не зависит (обозначены символом Ф), в формуле не учитывать. На основе приведенных формул (7.4) можно построить схему данного шифратора (рис. 7.10).

Таблица 7.4. Таблица истинности приоритетного шифратора 8×3 типа 155ИВ1

Входы									Выходы				
E	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	G	EO
0	Ф	Ф	Ф	Ф	Ф	Ф	Ф	Ф	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	1	Ф	0	0	1	1	0
1	0	0	0	0	0	1	Ф	Ф	0	1	0	1	0
1	0	0	0	0	1	Ф	Ф	Ф	0	1	1	1	0
1	0	0	0	1	Ф	Ф	Ф	Ф	1	0	0	1	0
1	0	0	1	Ф	Ф	Ф	Ф	Ф	1	0	1	1	0
1	0	1	Ф	Ф	Ф	Ф	Ф	Ф	1	1	0	1	0
1	1	Ф	Ф	Ф	Ф	Ф	Ф	Ф	1	1	1	1	0

$$G = E \cdot \sum_{i=0}^{7} I_i,$$

$$EO = E \cdot \prod_{i=0}^{7} \overline{I}_i,$$

$$A_2 = E \cdot \sum_{i=4}^{7} I_i, \qquad (7.4)$$

$$A_1 = E \cdot (I_7 + \overline{I}_7 \cdot I_6 + \overline{I}_7 \cdot \overline{I}_6 \cdot \overline{I}_5 \cdot \overline{I}_4 \cdot I_3 + \overline{I}_7 \cdot \overline{I}_6 \cdot \overline{I}_5 \cdot \overline{I}_4 \cdot \overline{I}_3 \cdot I_2),$$

$$A_0 = E \cdot (I_7 + \overline{I}_7 \cdot \overline{I}_6 \cdot I_5 + \overline{I}_7 \cdot \overline{I}_6 \cdot \overline{I}_5 \cdot \overline{I}_4 \cdot I_3 + \overline{I}_7 \cdot \overline{I}_6 \cdot \overline{I}_5 \cdot \overline{I}_4 \cdot \overline{I}_3 \cdot \overline{I}_2 \cdot I_1).$$

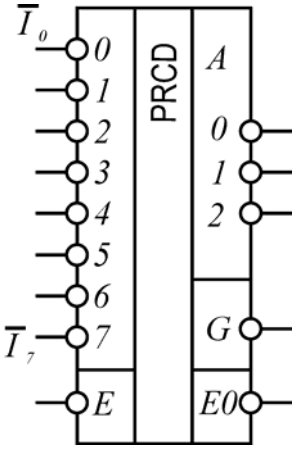


Рис. 7.10. Условное обозначение приоритетного шифратора 8×3 типа 155ИВ1

Сумматор

Сумматор – это логическая схема, выполняющая арифметическое (не логическое!) сложение и вычитание чисел. Сумматор имеет самостоятельное значение и выпускается в виде отдельной микросхемы, а также является главным действующим лицом в арифметико-логическом устройстве (АЛУ).

Синтезируем схему параллельного сумматора. Функция, которую выполняет это устройство, сводится к следующему. У него два входа и два выхода. На входы поступают одноразрядные двоичные числа (биты), на первом выходе находится бит суммы, на втором – бит переноса в старший значащий разряд.

Для синтеза схемы в первую очередь построим таблицу истинности параллельного одноразрядного полусумматора. Как мы уже определили, в этой таблице два входа (слагаемые биты) и два выхода (бит переноса и бит суммы), и табл. 7.5 задает вид двух функций в зависимости от двух аргументов.

Таблица 7.5. Таблица истинности полусумматора

Входы		**Выходы**	
x_1	x_2	y_1 (бит переноса)	y_2 (бит суммы)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Следующий шаг синтеза схемы – запись функций y_1, y_2 по таблице.

Построим СДНФ для бита переноса:

$$y_1 = x_1 \cdot x_2 . \qquad (7.5)$$

Это – известная функция логического умножения, которая реализуется при помощи логического элемента И.

Теперь СКНФ для бита суммы:

$$y_2 = (\overline{x_1} + \overline{x_2}) \cdot (x_1 + x_2) .$$

Эта функция также известна, она называется «сумма по модулю два». Преобразуем это выражение с использованием законов де Моргана. Это позволяет использовать функцию (7.5) в вычислении y_2 и упрощает логическую схему устройства:

$$y_2 = \overline{\overline{(\overline{x_1} + \overline{x_2})} + \overline{(x_1 + x_2)}} = \overline{(x_1 \cdot x_2) + \overline{(x_1 + x_2)}} = \overline{y_1 + \overline{(x_1 + x_2)}} . \qquad (7.6)$$

Схема, полученная в соответствии с выражениями (7.5), (7.6), представлена на рис. 7.11.

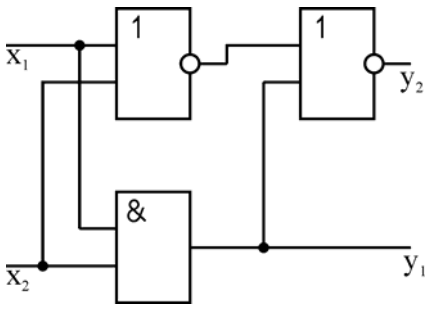


Рис. 7.11. Принципиальная схема полусумматора

Эта схема не является единственно возможной. Можно построить аналогичную схему в базисе И-НЕ или ИЛИ-НЕ. Важным этапом

проектирования является нахождение минимальной формы логической функции. Микросхема с минимальным количеством элементов будет иметь лучшие показатели по потребляемой мощности, занимаемой на кристалле площади, быстродействию. В рамках данного курса вопросы минимизации логических функций не рассматриваются (см. литературу для их самостоятельного изучения[4]).

Поскольку полусумматор складывает только два однобитных числа, но не учитывает бит переноса из предыдущего разряда, для полноценного суммирования требуется еще другая схема, называемая полным сумматором. Его функция заключается в следующем: на вход поступают два одноразрядных двоичных числа и бит переноса из предыдущего разряда (см. табл. 7.6). На выходе должен появляться бит суммы и бит переноса в следующий разряд. Таким образом, полный сумматор – это комбинационная схема с 3 входами и 2 выходами. Один из способов построения полного сумматора состоит в применении двух полусумматоров и элемента ИЛИ, как показано на рис. 7.12.

Таблица 7.6. Таблица истинности полного сумматора

Входы			**Выходы**	
x_1	x_2	c (бит переноса)	y_1 (бит переноса)	y_2 (бит суммы)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

На этом рисунке входы и выходы полного сумматора обозначены в соответствии с табл. 7.6. Промежуточные выходы первого и второго полусумматора обозначены дополнительно верхними индексами: y_1^1 и y_2^1 – бит переноса и бит суммы первого полусумматора, y_1^2 и y_2^2 – соответствующие выходы второго. Логика его работы заключается в следующем. Выход полусумматора 2 соответствует выходу всей схемы, полного сумматора. Логическая 1 здесь должна быть в тех случаях, когда на ее вход приходит либо логическая 1 от сумматора 1, либо логическая 1 со входа бита переноса, но не когда оба этих входа равны логической 1 или логическому 0, на выходе должен быть логический 0. Бит переноса полного сумматора формируется при помощи схемы ИЛИ из битов переноса полусумматоров 1 и 2, поскольку он должен быть равен логической 1, если хотя бы один из этих битов переноса равен 1.

[4] *Алексенко А.Г., Шагурин И.И.* Микросхемотехника : учеб. пособие для вузов / под ред. И.П. Степаненко. – М. : Радио и связь, 1982. – 416 с. : ил.

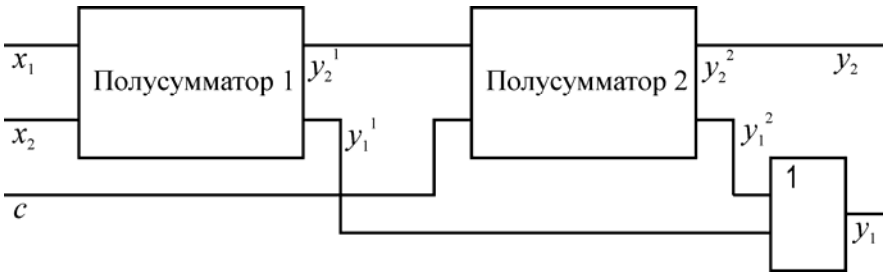


Рис. 7.12. Одна из возможных схем полного одноразрядного сумматора

Сложение двух многоразрядных чисел можно выполнить с помощью цепочки сумматоров (см. рис. 7.13). На рисунке представлен 4-разрядный сумматор. Во всех разрядах, за исключением младшего, должны быть полные сумматоры, чтобы учитывать бит переноса из предыдущего разряда. Такое устройство называется параллельным сумматором, поскольку все цифры представлены одновременно. Существуют также схемы последовательных сумматоров, в которых числа представляются последовательностью импульсов, и сложение осуществляется начиная с младшего разряда последовательно во времени.

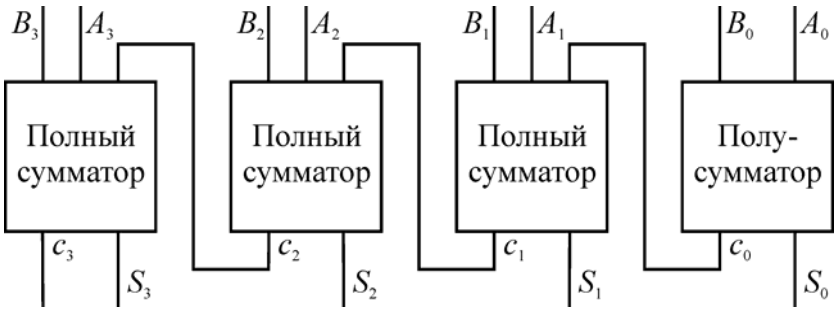


Рис. 7.13. Схема полного 4-разрядного сумматора

Контрольные вопросы и задания

1. Дайте определение комбинационной схемы.
2. Разработайте логическую схему дешифратора 3×8. Как построить эту схему в базисе И-НЕ? Какие выходы больше подходят для такой реализации – инверсные или прямые?
3. Постройте таблицу истинности дешифратора 3×8 с инверсными выходами.
4. Постройте схему приоритетного шифратора 4×2 и 8×3. Проверьте, как она работает.
5. Разработайте схему одноразрядного полусумматора и полного сумматора в базисе И-НЕ.
6. Разработайте схему 2-разрядного сумматора в базисе И-НЕ. Проверьте, как складываются 2-разрядные числа. Когда возникает переполнение сумматора?

Лекция 8. КОМБИНАЦИОННЫЕ ЛОГИЧЕСКИЕ СХЕМЫ. Часть 2

Схемы контроля четности

Эти схемы используются для обнаружения однократных ошибок при передаче данных по линиям связи. В передатчике к *n*-разрядному слову перед его посылкой в линию связи добавляется контрольный разряд с таким значением, чтобы сумма единиц в *n*+1-разрядном слове была четной. В приемнике производится контроль всего *n*+1-разрядного слова на четность. Если число единиц в принятом *n*+1-разрядном слове будет нечетно, то фиксируется ошибка при передаче данных. Передачу данных нужно повторить.

Для рассмотрения принципа работы этой схемы вспомним функцию двух аргументов «сумма по модулю два» или «логическая неравнозначность». Она равна логическому 0 в случае, когда два входных аргумента одинаковы и равна логической 1, когда входные аргументы разные. Ее условное обозначение $y = x_1 \oplus x_2$, и в результате ее применения к произвольному числу аргументов получают так называемую функцию четности:

$$\varphi_0 = \overline{\sum_{p=0}^{n-1}{}^{*} I_p} = \overline{I_{n-1} \oplus I_{n-2} \oplus \ldots \oplus I_0}, \qquad (8.1)$$

поскольку $\varphi_0 = 1$ только при четном числе аргументов I_p, равных 1 (четными считаются 0, 2, 4 и т. д.). Приведенная в формуле (8.1) звездочка означает не простое суммирование, а суммирование по модулю 2. Функция $\overline{\varphi_0}$ называется функцией нечетности.

Конкретным примером такой схемы является 8-разрядная микросхема контроля четности типа 155ИП2. Ее условное графическое обозначение приведено на рис. 8.1.

Микросхема имеет 8 информационных входов $I_0 - I_7$, два управляющих входа: *EE* (Even Enable, в переводе разрешение четности) и *OE* (Odd Enable, разрешение нечетности), а также два выхода: *PE* (Parity even, четный паритет), *PO* (Parity Odd, нечетный паритет). Функции выхода можно записать следующим образом:

$$\begin{aligned} PE &= \overline{EE} \cdot \sum_{p=0}^{7}{}^{*} I_p + \overline{OE} \cdot \overline{\sum_{p=0}^{7}{}^{*} I_p}, \\ PO &= \overline{EE} \cdot \overline{\sum_{p=0}^{7}{}^{*} I_p} + \overline{OE} \cdot \sum_{p=0}^{7}{}^{*} I_p. \end{aligned} \qquad (8.2)$$

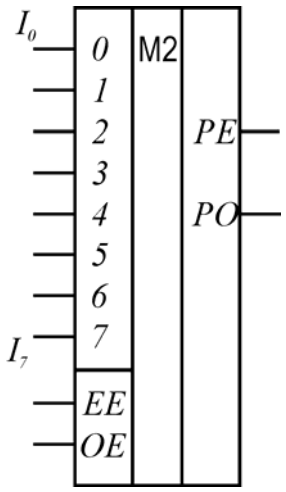


Рис. 8.1. Условное обозначение схемы контроля четности 155ИП2

Из формул (8.2) видно, что $PE = PO = 1$ при $EE = OE = 0$, и $PE = PO = 0$ при $EE = OE = 1$, независимо от значений информационных сигналов I_p. Если $EE = \overline{OE} = 1$, то $PE = \varphi_0$, функции четности, $PO = \overline{\varphi_0}$, функции нечетности. При $EE = \overline{OE} = 0$ эти два выхода меняются местами, PE становится функцией нечетности, а PO – функцией четности. При передаче данных можно использовать либо контроль четности, либо контроль нечетности.

Передача данных по линии связи

На рис. 8.2 поясняется процесс передачи данных по линии связи с контролем нечетности. Для проверки правильности передачи необходимо использовать две микросхемы контроля четности. Левая интегральная схема (ИС) является генератором контрольного разряда, передаваемого двумя сигналами (с учетом того, что $EE = 0$, $OE = 1$):

$$\begin{aligned} PE_1 &= \sum_{p=0}^{7} {}^{*} D_p, \\ PO_1 &= \overline{\sum_{p=0}^{7} {}^{*} D_p} = \overline{PE_1}. \end{aligned} \tag{8.3}$$

Схема справа производит контроль нечетности принятых данных $D'_7 - D'_0$ на основании контрольного разряда PE'_1 и PO'_1:

$$PE_2 = \overline{PE'_1} \cdot \sum_{p=0}^{7} {}^{*} D'_p + \overline{PO'_1} \cdot \overline{\sum_{p=0}^{7} {}^{*} D'_p}. \tag{8.4}$$

При отсутствии ошибок в линии связи $D'_p = D_p$ для всех p, $PE'_1 = PE_1$, $PO'_1 = PO_1 = \overline{PE_1}$. Тогда из (8.4) следует:

$$PE_2 = \overline{PE_1} \cdot \sum_{p=0}^{7} {}^{*} D_p + PE_1 \cdot \overline{\sum_{p=0}^{7} {}^{*} D_p} = PE_1 \oplus \sum_{p=0}^{7} {}^{*} D_p .$$

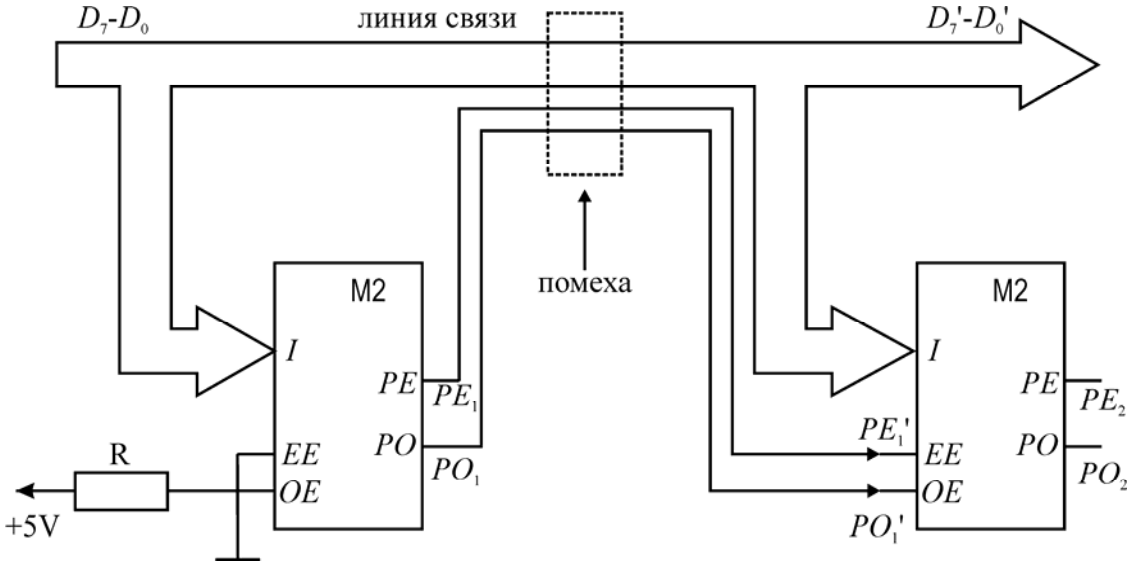


Рис. 8.2. Контроль данных при передаче по линиям связи

С учетом выражения (8.3) для PE_1:

$$PE_2 = \sum_{p=0}^{7} {}^{*}D_p \oplus \sum_{p=0}^{7} {}^{*}D_p = 0,$$

$$PO_2 = \overline{PE_2} = 1.$$

Значения сигналов PO_2 и PE_2 изменяются на инверсные при возникновении ошибок в нечетном числе разрядов линии связи. Контрольный разряд можно передавать и одним проводом (например, PE_1), а другой сигнал получать инвертированием принятого контрольного разряда на другом конце линии передачи.

Схемы равнозначности кодов

Пусть заданы две совокупности переменных $v' = (x_n ... x_1)$ и $v'' = (y_n ... y_1)$. x_p, y_p могут принимать значения 0 или 1. Комбинационная схема, реализующая функцию $f(v) = f(v', v'')$, которая равна 1 только в том случае, когда $x_p = y_p$ для всех $p = 1...n$, называется схемой равнозначности кодов.

Разряды x_p, y_p равны только в том случае, если $x_p \oplus \overline{y_p} = 1$, поэтому функция $f(v) = \prod_{p=1}^{n} (x_p \oplus \overline{y_p}) = \overline{\sum_{p=1}^{n} x_p \oplus y_p}$ принимает значение 1 только при попарном равенстве одинаковых разрядов кодов v' и v''.

Такие схемы необходимы при выполнении операций сравнения. Варианты реализации схемы равнозначности с использованием различных логических элементов приведены на рис. 8.3.

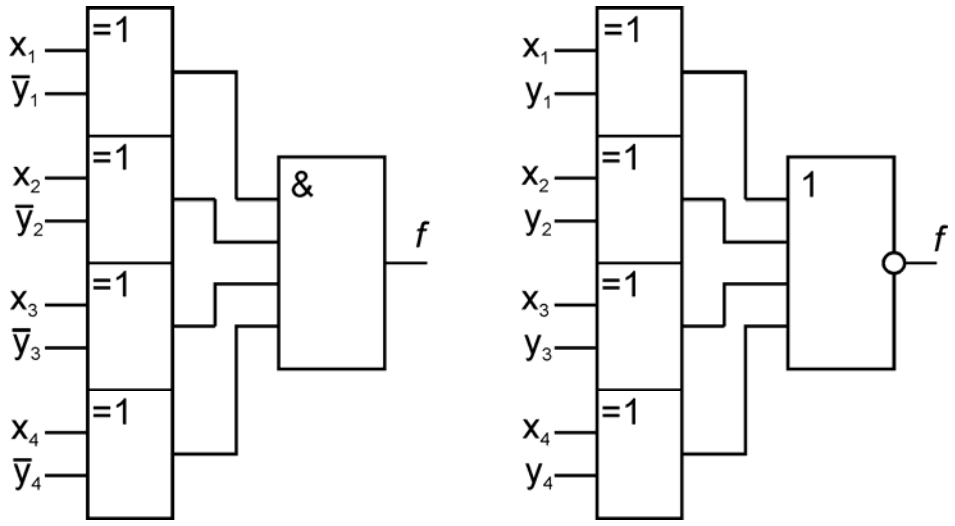


Рис. 8.3. Варианты реализации схемы равнозначности кодов

В общем виде схема сравнения имеет более сложный вид и используется микропроцессором в специализированных операциях.

Арифметико-логические устройства (АЛУ)

Эти устройства широко используются при построении арифметических узлов, в частности, АЛУ является составной частью любого микропроцессора.

АЛУ выполняет арифметические операции и 16 логических операций. Переключение режима работы осуществляется сигналом Mode (M). При M = 0 АЛУ выполняет арифметические операции, при M = 1 – логические. Выбор одной из логических операций задается кодом $E = (E_3, E_2, E_1, E_0)$. Логические операции выполняются поразрядно.

Функционально АЛУ состоит из двух регистров, сумматора и схем управления. Один из способов построения сумматора описан в лекции 7.

Регистры выполняют функции хранения операндов и результата, а также (при необходимости) сдвига. Один регистр должен иметь разрядность двойного машинного слова, поскольку в нем будет храниться результат выполнения арифметической операции. Второй может иметь разрядность одного машинного слова. Схемы управления принимают по шинам инструкций управляющие сигналы от устройства управления и вырабатывают сигналы управления регистрами и сумматором.

АЛУ может быть построено на микросхеме (см. рис. 8.4), а в персональном компьютере является составной частью микропроцессора. Обычно АЛУ имеет 4 двоичных разряда, а для наращивания разрядности их объединяют с использованием линий формирования переносов (на рис. 8.4 выводы G, P, C0, C4).

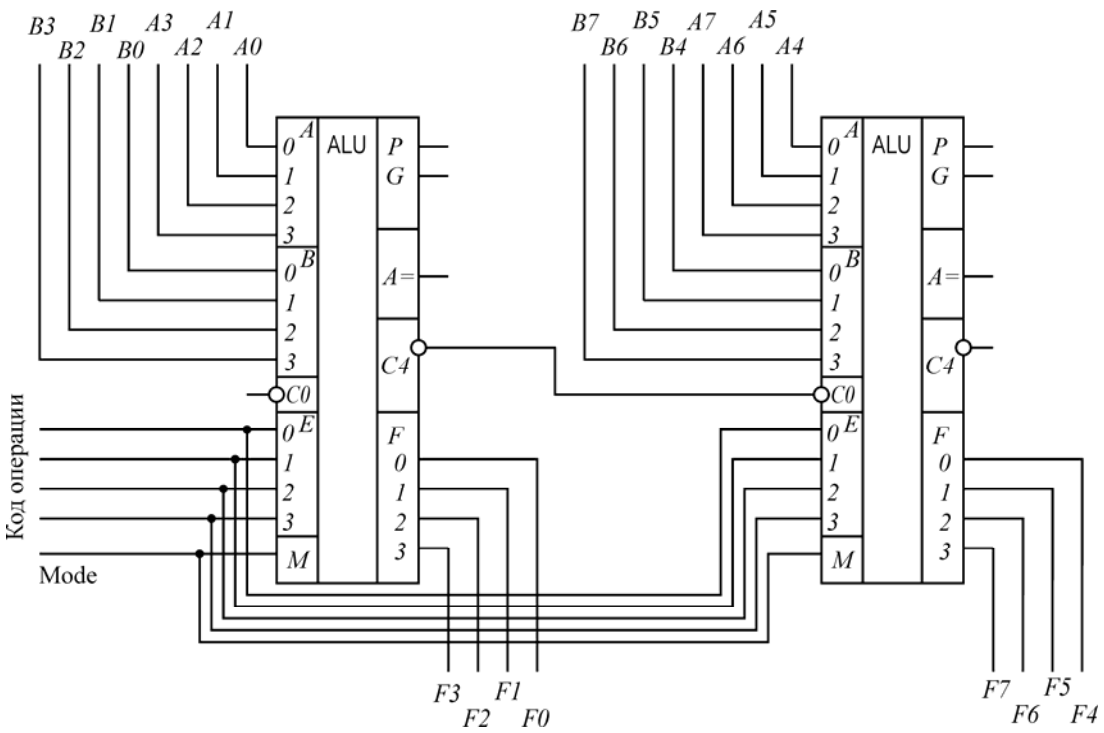


Рис. 8.4. Графическое обозначение арифметико-логических устройств и схема их включения для увеличения разрядности

Знакогенераторы и индикаторные устройства

Индикаторные устройства служат для визуальной индикации внутреннего состояния электронных устройств или цифровой информации, преобразуемой электронными устройствами и ЭВМ.

Обычно индицируется цифровая информация, но в принципе речь может идти и о текстовой информации (как это сделано в индикаторах типа «бегущая строка»). Существуют также индикаторы, отображающие информацию на шкале. Индикация основана на том, что под действием электрического тока в устройстве индикации либо изменяются оптические свойства, либо это устройство само генерирует свет.

Наиболее простой пример – шкальный индикатор. Если комбинационная схема – дешифратор 3×8, то в зависимости от входного кода горит соответствующий светодиод. Дешифратор может иметь другую функцию: когда горят все светодиоды, номер которых меньше поданного на вход кода.

Из нескольких светодиодов можно составить индикаторы или матрицы, отображающие буквы и цифры. В семисегментных индикаторах семь излучающих сегментов-светодиодов (обозначенных буквами a – g) расположены так, что зажигая их в определенной комбинации, можно высветить все десять (и даже шестнадцать) цифр.

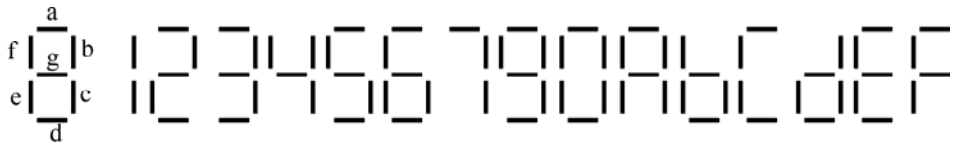


Рис. 8.5. Семисегментный индикатор и представление 16 цифр

Для того чтобы изготовить схему, управляющую работой такого индикатора, необходимо составить таблицу истинности для каждого отдельного сегмента и по этой таблице построить комбинационную схему с 4 входами и 7 выходами, управляющую работой индикатора.

Шинная структура ЭВМ

Для сопряжения и связи устройств компьютера между собой служит системная шина. Впервые она появилась в компьютерах второго поколения. Системная шина современного компьютера представляет собой целый набор соединительных проводов и специальных схем и включает в себя:

- шину данных, содержащую соединительные провода и схемы сопряжения для параллельной передачи разрядов машинного слова;
- шину адреса, содержащую соединительные провода и схемы сопряжения для параллельной передачи разрядов адреса ячеек запоминающего устройства или портов ввода-вывода внешних устройств;
- шину инструкций, содержащую соединительные провода и схемы сопряжения для передачи управляющих сигналов (инструкций) в различные узлы и блоки машины;
- шину питания, предназначенную для подключения блоков и узлов компьютера к системе энергоснабжения.

Системная шина обеспечивает передачу и прием информации, соединяя различные устройства:

- микропроцессор – оперативная память;
- микропроцессор – внешние устройства (порты ввода-вывода внешних устройств);
- оперативная память – порты ввода-вывода внешних устройств (в режиме прямого доступа памяти под управлением контроллера прямого доступа к памяти).

Все устройства подключены к шине либо непосредственно, либо через специальные схемы – контроллеры. Управление работой шины осуществляется обычно специальным устройством, которое называется контроллером шины. Структурная схема шины данных представлена на рис. 8.6. При подключении к шине различных устройств типична ситуация, когда соединяющие провода имеют большую длину, а следова-

тельно, индуктивность и емкость. Кроме того, один активный элемент работает на большое число входов, имеющих реальное входное сопротивление. Это приводит к тому, что необходимо использовать логические элементы с повышенной нагрузочной способностью (усилители тока). В качестве таких устройств служат буферные усилители и приемопередатчики.

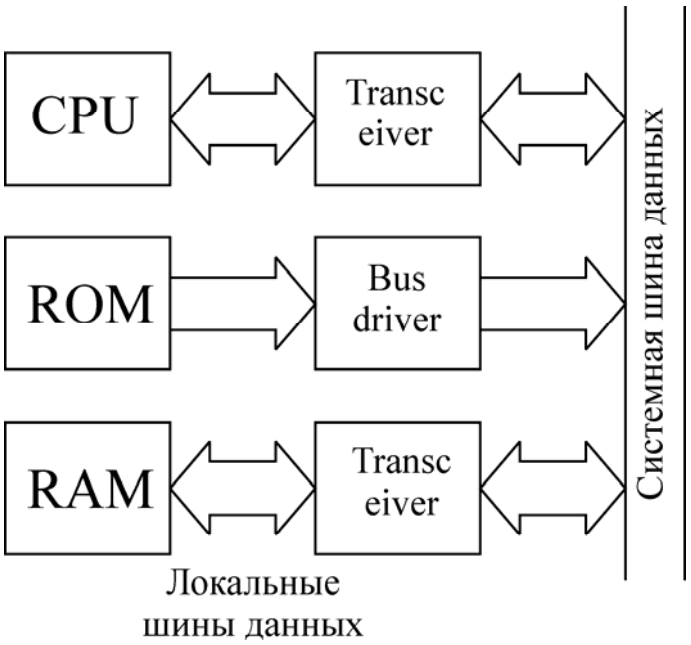


Рис. 8.6. Структурная схема шины данных

Буферные усилители и приемопередатчики

Для подключения микропроцессора, памяти, внешних устройств к системным шинам адреса и данных необходимы буферные усилители (драйверы) с тремя состояниями выхода (0, 1 и Z-состояние). Такие драйверы называются шинными формирователями (Bus Driver). Драйверы применяются для буферирования шины данных и адреса в одном направлении.

$$DO_i = \begin{cases} DI_i, \ OE = 1 \\ Z \text{ - состояние}, \ OE = 0 \end{cases}.$$

На рис. 8.7 представлено условное графическое изображение драйверов и схема их подключения к шине данных.

Приемопередатчики (transceivers) широко используются при построении микропроцессорных систем для буферирования двунаправленных шин. Шинная архитектура в ЭВМ позволяет структурировать схему и является более выгодной, чем соединение каждого элемента с каждым. Выходы буферных усилителей и приемопередатчиков можно включать параллельно при условии, что в любой момент времени активным является только один из них.

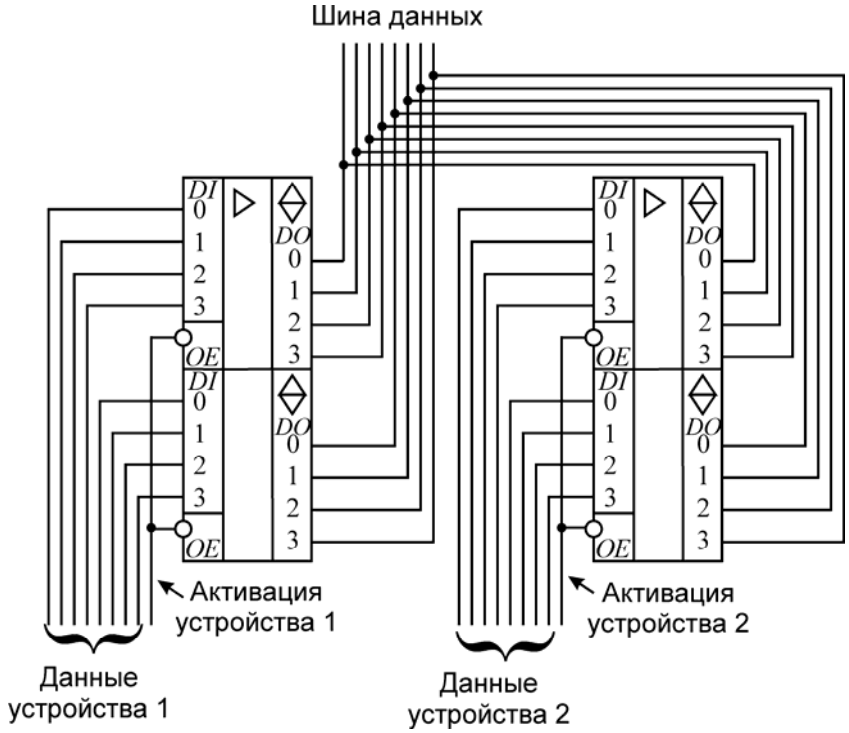


Рис. 8.7. Подключение устройств к шине данных с использованием буферного усилителя 555АП5

В структурной схеме шины данных на рис. 8.6 представлены как буферные усилители, так и приемопередатчики. Те узлы, которые используются только для чтения данных, подключаются при помощи буферных усилителей. Там, где требуется двунаправленная передача данных, используется приемопередатчик.

Контрольные вопросы и задания

1. Какие основные типы комбинационных схем, используемых при построении цифровых устройств, вам известны?
2. Опишите принцип контроля четности (нечетности) данных при передаче по линиям связи.
3. Напишите таблицу истинности для дешифратора семисегментного индикатора, отображающего десятичные цифры 0–9. Представьте функции выхода в виде СДНФ.
4. Разработайте логическую схему дешифратора, управляющего семисегментным индикатором.

Лекция 9. СХЕМЫ С ПАМЯТЬЮ

Все обсуждавшиеся ранее схемы откликались на фиксированную статическую комбинацию сигналов на входе (поэтому они и назывались комбинационными). Например, на выходе логического элемента И-НЕ появится логическая единица в тот и только в том случае, когда на его входы поданы логические нули. Однако при помощи этих схем можно сделать не все действия, необходимые при обработке сигналов.

Часто возникает нужда в таких схемах, значения логического сигнала на выходах которых зависят не только от текущего состояния входа, но и от предыдущих выходных состояний (а значит, и от предыдущих входных состояний). В качестве примера можно привести полезное устройство, называемое двоичным счетчиком. Трехразрядный двоичный счетчик имеет как минимум три выхода и один вход. На входе через определенные промежутки времени инициируется логический перепад из 0 в 1 или из 1 в 0, и по этому сигналу (он называется тактовым сигналом или clock signal) изменяется состояние счетчика. Начальное состояние счетчика 000, после прихода тактового сигнала его значение изменяется на 001, следующий сигнал приводит к изменению выхода на 010, затем 011, 100, 101 и т. д.

Для выполнения такого рода задач существует большой класс цифровых схем, функционирование которых определяется помимо набора входных сигналов, еще и состояниями, в которых они пребывали раньше. Это схемы с памятью. Их также называют последовательностными (в некоторых источниках последовательными) схемами или цифровыми автоматами. Последовательностная схема помнит о прошедших событиях, и поэтому при одних и тех же входных сигналах выходные сигналы могут отличаться, в отличие от комбинационной схемы, в которой состояние выхода определятся только состоянием входа в данный момент времени.

На рис. 9.1 изображен общий вид последовательностной схемы. Она состоит из комбинационной схемы и блока задержек. Блок задержек включен в обратную связь комбинационной схемы. Элементы задержки задерживают (запоминают) внутренние переменные Y на время Δt. Эти сигналы появляются на входах комбинационной схемы (выходах блока задержек) через время Δt и способны вызвать изменение ее выходных сигналов. При фиксированных значениях внутренних переменных Y последовательностная схема ведет себя как обычная комбинационная, т. е. реализует однозначное соответствие между входными и выходными сигналами. Однако при изменении входных сигналов X могут измениться внутренние состояния схемы Y. Если после этого

подать прежние значения входных сигналов, то выходные значения могут получиться совсем другими, поскольку изменилось внутреннее состояние.

Рис. 9.1. Общая модель последовательностных схем

Простейшая схема с обратной связью и двумя состояниями представлена на рис. 9.2. Это кольцо, состоящее из двух инверторов. У этой схемы одна внутренняя переменная, а входных сигналов нет совсем. При подаче питания в этой схеме может быть реализовано одно из двух устойчивых состояний единственной внутренней переменной, как это показано на рисунке. Каждое из этих устойчивых состояний может сохраняться, пока подано питание на микросхему, потому что сигнал с выхода одного элемента поддерживает состояние второго логического элемента.

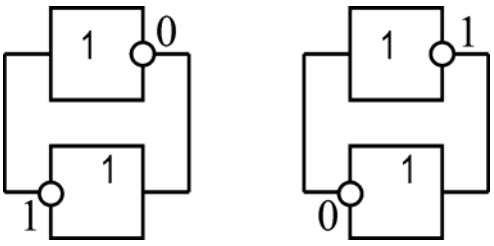


Рис. 9.2. Простейшая последовательностная схема, содержащая два инвертора, замкнутых в кольцо

Таким элементом неудобно пользоваться, поскольку его состоянием невозможно управлять. Типичным примером последовательностной схемы является триггер. Это схема, имеющая два устойчивых состояния и позволяющая осуществлять управляемый переход из одного состояния в другое при помощи управляющих сигналов.

RS-триггер

Рассмотрим пример логического проектирования *RS*-триггера, задачей которого является раздельная установка входами в состояние 0 или 1. Одновременная подача двух управляющих сигналов считается несовместимой. Его состояния могут быть изображены в виде *таблицы переходов* (в отличие от *таблицы истинности* для комбинационных схем, таблица переходов отражает состояние выхода после любого перехода, вызванного входным сигналом).

Рассмотрим таблицу переходов такого триггера (табл. 9.1).

Таблица 9.1. Таблица переходов *RS*-триггера

S	R	Qt	Q	$\overline{Q}$	название состояния
0	0	0	0	1	хранение 0
0	1	0	0	1	подтверждение 0
1	0	0	1	0	установка 1
1	1	0	Ф	Ф	неопр. состояние
0	0	1	1	0	хранение 1
0	1	1	0	1	установка 0
1	0	1	1	0	подтверждение 0
1	1	1	Ф	Ф	неопр. состояние

Табл. 9.1 отражает логику работы по входам (R, S) с учетом внутреннего состояния Q_t до прихода управляющего сигнала. Q – выходной сигнал. Входы триггера названы по первым буквам английских слов Set (установка) и Reset (сброс). Сигнал логической единицы на входе установки (S) заставляет выход Q перейти (или остаться) в состояние логической 1. Сигнал логической 1 на входе сброса (R) заставляет выход Q перейти (или остаться) в состояние логического 0. Одновременная подача двух управляющих сигналов недопустима. При составлении таблицы переходов эта ситуация отражается символом Ф и означает, что состояние этой переменной может быть любым (принимать значения либо 0, либо 1).

По таблице переходов можно получить, составляя СДНФ, следующую зависимость:

$$Q = \overline{R}S\overline{Q_t} + \overline{R}\overline{S}Q_t + \overline{R}SQ_t .$$

Упростим это выражение, вынося за скобки $\overline{R}Q_t$:

$$Q = \overline{R}S\overline{Q_t} + \overline{R}Q_t .$$

Далее проведем преобразования при условии, что неопределенные состояния принимают значение 1 (как будет видно далее, это сильно упрощает зависимости выходных сигналов от входных). Тогда:

$$Q = RS\overline{Q_t} + \overline{R}S\overline{Q_t} + RSQ_t + \overline{R}SQ_t + \overline{R}Q_t .$$

Преобразуя выражение, получим:

$$Q = S(R\overline{Q_t} + \overline{R}\,\overline{Q_t} + RQ_t + \overline{R}Q_t) + \overline{R}Q_t = S + \overline{R}Q_t = S + S + \overline{R}Q_t = S + Q\,.$$

$$\overline{Q} = \overline{S + Q}\,. \qquad (9.1)$$

Инвертируем левую и правую части, и, используя закон де Моргана, найдем также выражение для Q:

$$Q = \overline{\overline{S}\,\overline{Q}}\,. \qquad (9.2)$$

Теперь по аналогии запишем СДНФ для выходной переменной $\overline{Q}$.

$$\overline{Q} = R + \overline{Q_t}S = R + \overline{Q}\,.$$

$$Q = \overline{R + \overline{Q}}\,. \qquad (9.3)$$

$$\overline{Q} = \overline{\overline{R}Q}\,. \qquad (9.4)$$

Получены уравнения, в которых осталась зависимость только от входных и выходных переменных. По уравнениям (9.1) и (9.3) можно построить схему, которая работает в соответствии с табл. 9.1. По виду функции можно заключить, что для ее работы потребуется 2 элемента ИЛИ-НЕ (см. рис. 9.3, *а*). Формулы (9.2) и (9.4) используем для построения двойственной схемы (см. рис. 9.3, *б*). Для нее необходимы два элемента И-НЕ.

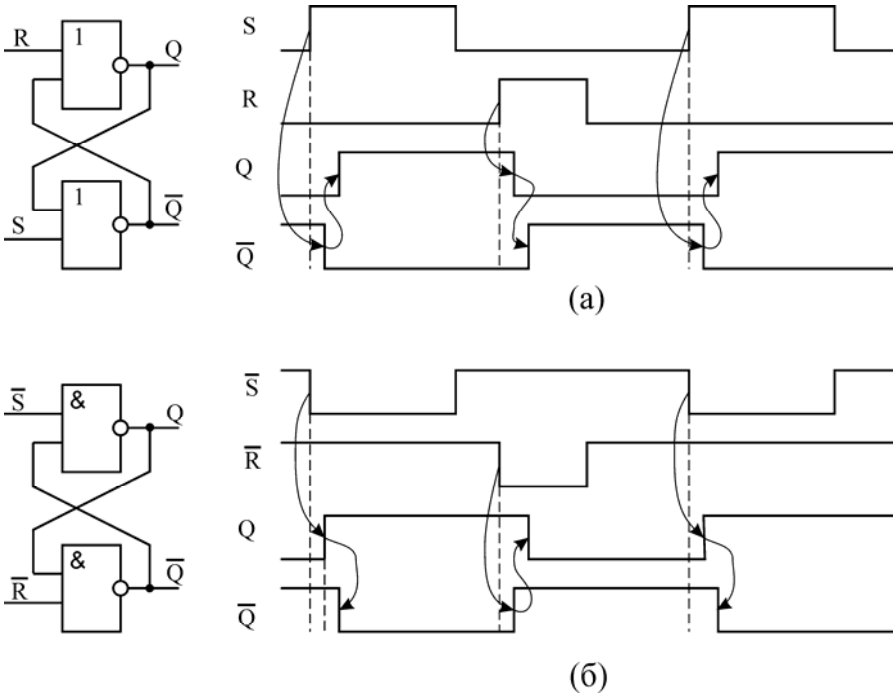


Рис. 9.3. Схемы *RS*-триггеров и временные диаграммы, поясняющие их работу

Проанализируем работу *RS*-триггера, подавая на его входы управляющие сигналы. Пусть изначально $Q = 0$, $\overline{Q} = 1$ (триггер сброшен в 0). На входах сигналы $R = S = 0$. Подадим а вход установки S логическую 1. На выходе нижнего элемента ИЛИ-НЕ обязательно появится логический 0 ($\overline{Q} = 0$, см. рис. 9.3, *а*) через время задержки, которым характери-

зуется логический элемент. Поскольку этот сигнал подается на верхний логический элемент, то вместе с $R = 0$ они дадут на выходе $Q = 1$ (триггер установлен в 1). При этом состояние выхода $\overline{Q} = 0$ не изменится, ведь на вход S приходит логическая 1, и $\overline{Q} = 0$ вне зависимости от того, что приходит на второй вход нижнего логического элемента. Триггер находится в устойчивом состоянии. Даже если теперь S перевести в состояние 0, то состояние триггера не изменится (режим хранения). Для того, чтобы триггер сбросить в 0, необходимо подать логическую 1 на вход R. При этом на выходе Q логический 0, который вместе с $S = 0$ дает $\overline{Q} = 1$. Таким образом, триггер оказывается сброшенным.

Применение RS-триггера

Поскольку триггер может хранить внутреннее состояние сколь угодно долго (пока есть питание), это устройство можно использовать как элементарную ячейку памяти. Такую ячейку памяти называют *статической*, поскольку она не требует регенерации в процессе работы. Более распространенной (и более простой по конструкции) является *динамическая* память, основанная на сохранении заряда конденсатора. Поскольку конденсатор в процессе работы может самопроизвольно разряжаться, эта память требует регенерации (обновления).

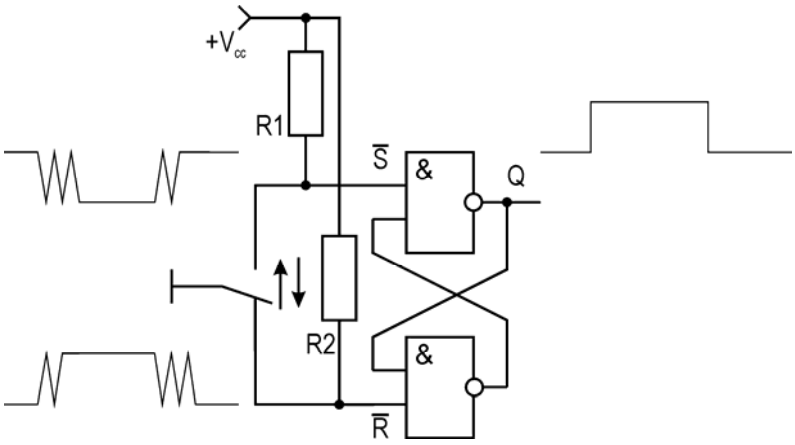


Рис. 9.4. Применение RS-триггера для подавления дребезга механических контактов

Другим оригинальным применением RS-триггера является устройство для устранения дребезга механических контактов. Оно предназначено для устранения множественных ложных срабатываний схем, подключенных к механическому контакту. Такой дребезг является неизбежным в механическом переключателе. За время одного переключения контакт может нарушиться и восстановиться много раз. Принцип работы такой схемы продемонстрирован на рис. 9.4. Механический переключатель подключен к входам RS-триггера. При нажатии на кнопку переключателя сначала размыкается нижний контакт. При этом ничего не происходит, так как триггер сброшен. Затем начинает замы-

каться верхний контакт. При первом же появлении логического нуля на входе $\overline{S}$ триггер переключается в состояние логической 1 и остается в устойчивом состоянии. Обратный переход (в логический ноль) происходит при отпускании механического контакта.

При вычерчивании логических схем RS-триггер обычно изображают в виде, представленном на рис. 9.5.

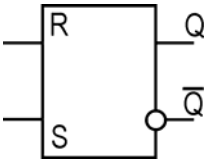


Рис. 9.5. Графическое изображение RS-триггера

Синхронный RS-триггер

Существенной чертой большинства последовательностных логических схем является необходимость осуществлять переходы в определенные моменты времени. Обычно это достигается с помощью регулярной последовательности тактовых импульсов. Для этого в триггерах предусматривается специальный вход CP (Clock Pulse) или просто Clock (C). Его еще называют тактовым входом, входом синхронизации или строб-входом. Соответствующая схема имеет вид, представленный на рис. 9.6.

На этом рисунке представлена схема синхронного RS-триггера, который устроен так, что может изменять свое состояние либо при помощи асинхронных входов (независимо от тактового сигнала), либо тогда, когда входные импульсы *CP* принимают значение 1.

Когда на входе *CP* действует логический 0, на входах схем И тоже логический 0, и при наличии нулей на асинхронных входах значение выходных сигналов Q и $\overline{Q}$ не изменяется.

Когда же на входе *CP* появляется логическая 1, у каждой из схем И на одном из входов логический уровень высокий, и логические уровни со входов *R* и *S* передаются на входы R' и S'. Состояние триггера в этом случае будет устанавливаться в соответствии с таблицей переходов.

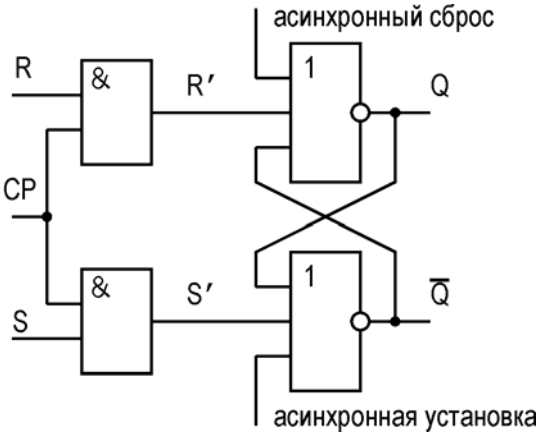


Рис. 9.6. Возможная схема синхронного RS-триггера

Когда сигнал *CP* возвращается в 0, состояние триггера, соответствующее комбинации входных значений *S* и *R* сохраняется, или защелкивается. Дополнительные входы асинхронного сброса и установки, на которых в состоянии нормальной работы поддерживается значение логического 0, обеспечивает возможность сброса и установки триггера вне зависимости от состояния тактового сигнала.

D-триггер

D-триггер, или триггер данных, можно сделать из синхронного *RS*-триггера, для нового входа *D* которого выполняется условие: $D = S = \overline{R}$. Отличительной особенностью такой схемы включения является то, что входы S и R не могут одновременно принимать значение логической единицы, и, соответственно, не может быть неопределенного состояния. Состояние этого триггера сохраняется, пока на входе *CP* присутствует логический 0; когда *CP* = 1, логический уровень со входа *D* передается на выход *Q*. Таким образом, состояние выхода D-триггера описывается уравнением:

$$Q = D \cdot CP + Q_t \cdot \overline{CP}.$$

Здесь Q_t – внутреннее состояние триггера; *Q* – выход триггера; *D* – информационный вход; *CP* – тактовый вход. Видно, что на выходе триггера состояние сохраняется, пока на тактовом входе ноль. Если на тактовом входе единица, то выход устанавливается в соответствии с состоянием информационного входа *D*.

Этот триггер предназначен для считывания и хранения данных (см. рис. 9.7).

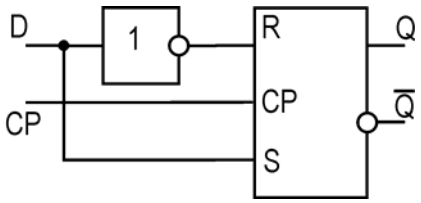


Рис. 9.7. Возможная схема D-триггера, построенная на базе синхронного RS-триггера (сравните с рис. 9.6)

JK-триггер

Это наиболее гибкое усовершенствование *RS*-триггера. В отличие от *D*-триггера здесь два входа, но удается избежать неопределенности запоминаемого логического состояния при *S* = *R* = 1 путем подачи обратной связи на входы со стороны соответствующего выхода (рис. 9.8).

Принято вход установки называть *J*, а вход сброса – *K*. Триггер чувствителен ко входным сигналам *J*, *K* только тогда, когда тактовый сигнал *CP* принимает высокий уровень (логической 1) и перестает быть чувствительным к состоянию входа, когда *CP* = 0 (низкий уровень).

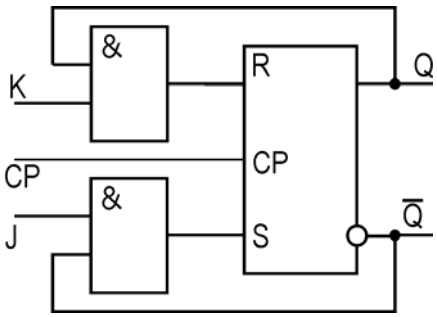


Рис. 9.8. Возможная схема *JK*-триггера, построенная на базе синхронного *RS*-триггера (сравните с рис. 9.6, 9.7)

Таблица переходов (см. табл. 9.2) выглядит подобно таблице переходов *RS*-триггера, за исключением случая $J = K = 1$, которое более не является запрещенным. В этом случае происходит переключение, в результате которого сигнал на выходе изменяет свое значение на противоположное всякий раз, когда тактовый сигнал принимает высокий уровень ($CP = 1$).

Таблица 9.2. Таблица переходов *JK*-триггера

J	K	Q_{t+1}
0	0	Q_t
1	0	1
0	1	0
1	1	$\overline{Q_t}$

Таким образом, в случае $J = K = 1$ всякий раз, когда приходит тактовый импульс CP, состояние схемы на выходе изменяет свое значение на противоположное. Однако, как видно из схемы, при этом необходимо соблюдать жесткие временные ограничения, наложенные на длительность импульса стробирования. Тактовый импульс *CP* должен заканчиваться прежде, чем изменится значение сигналов на выходе триггера и на входные схемы И поступят изменившиеся сигналы со стороны выхода. Поэтому большинство общеупотребительных триггеров включают в себя дополнительное усовершенствование, делающее их чувствительными не к уровню входного напряжения, а исключительно к его перепаду. Для этого достаточно сформировать короткий импульс («продифференцировать» входной сигнал). Эти входы называются динамическими, или импульсными. Схема для реализации такого входа представлена на рис. 9.9. Принцип работы схемы заключается в следующем. Если на вход подается логический 0, то на выходе – логический 0. Если на вход подается логическая 1, то на выходе – тоже логический 0, поскольку на второй вход элемента И приходит логический 0, получившийся в результате инверсии. Однако, если на вход схемы

приходит перепад из логического 0 в логическую 1, то на короткое время на входах элемента И появляются логические 1. Это связано с тем, что инвертор обладает некоторой задержкой, и на его выходе сигнал изменяется с 1 в 0 только через время задержки. Таким образом, на выходе элемента И появится импульс с длительностью, равной времени задержки инвертора.

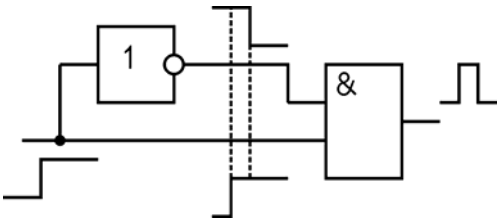


Рис. 9.9. Схема реализации динамического входа

Рассмотренные триггеры могут иметь такие динамические входы, реагирующие только на логический перепад, а сами входы могут быть прямыми (реагируют на перепад из 0 в 1) или инверсными (реагируют на перепад из 1 в 0). Некоторые условные обозначения триггеров с динамическими входами представлены на рис. 9.10.

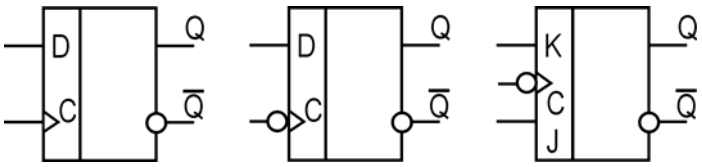


Рис. 9.10. Триггеры с динамическими входами

Контрольные вопросы и задания

1. В чем принципиальное отличие последовательностных и комбинационных схем?
2. Дайте определение триггера.
3. Какие различные виды триггеров вам известны?
4. Нарисуйте схему RS-триггера и поясните его работу.
5. Что такое синхронный и асинхронный триггеры?
6. Чем отличаются статические и динамические входы?

Лекция 10. РЕГИСТРЫ И ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Регистры

Регистром называется последовательностное логическое устройство, предназначенное для хранения многоразрядных двоичных чисел и выполнения преобразований над ними. Регистр представляет собой совокупность триггеров, число которых соответствует числу разрядов в слове. Количество триггеров соответствует разрядности регистра. С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами. Фактически любое цифровое устройство можно представить в виде совокупности регистров, соединенных друг с другом при помощи комбинационных цифровых устройств.

Функциональное назначение регистра определяет набор операций, которые может выполнять этот регистр. Типичными являются следующие операции:

- прием слова в регистр;
- передача слова из регистра;
- сдвиг слова влево или вправо на заданное число разрядов;
- преобразование последовательного кода слова в параллельный и обратно;
- установка регистра в начальное состояние (сброс).

Регистры классифицируются по следующим видам:

- накопительные (регистры памяти);
- сдвигающие.

В свою очередь сдвигающие регистры делятся:

- по способу ввода-вывода информации:
  - параллельные – запись и считывание информации происходит одновременно на все входы и со всех выходов;
  - последовательные – запись и считывание информации происходит в первый триггер, а та информация, которая была в этом триггере, перезаписывается в следующий – то же самое происходит и с остальными триггерами;
  - комбинированные;
- по направлению передачи информации:
  - однонаправленные;
  - реверсивные.

Регистры для хранения данных

Регистры хранения используются для приема, хранения и выдачи многоразрядного кода. Они представляют собой совокупность одноступенчатых триггеров (как правило, D-типа) с общим входом синхронизации. Иногда в регистре имеется также и общий вход асинхронной установки всех триггеров в «0».

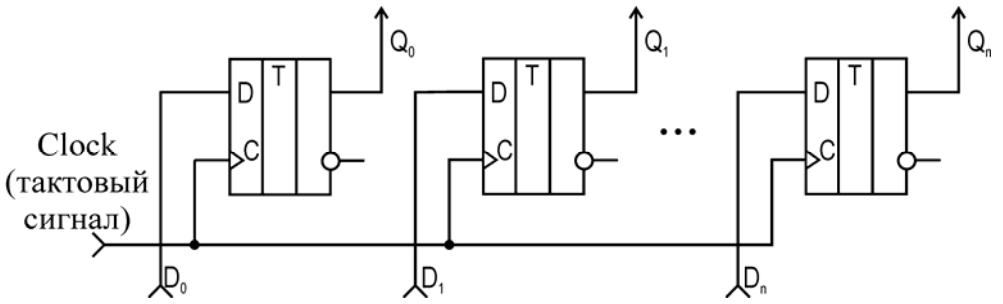


Рис. 10.1. Регистр данных на D-триггерах

Мы рассмотрели, как D-триггер запоминает один бит информации, когда на вход тактового сигнала приходит фронт импульса. Для того чтобы обеспечить хранение байта данных (или большего количества информации), необходимо объединить восемь (или более) триггеров и подать на них единый управляющий сигнал.

Такой регистр является неотъемлемой частью любой вычислительной системы (см. рис. 10.1).

Регистры сдвига

Регистр сдвига – регистр, обеспечивающий, помимо хранения информации, сдвиг влево или вправо всех разрядов одновременно на одинаковое число позиций. При этом выдвигаемые за пределы регистра разряды теряются, а в освобождающиеся разряды заносится информация, поступающая по отдельному внешнему входу регистра сдвига. Обычно эти регистры при подаче синхросигнала обеспечивают сдвиг кода на одну позицию влево или вправо. Но существуют и универсальные регистры сдвига, которые выполняют сдвиг как влево, так и вправо в зависимости от значения сигнала на специальном управляющем входе или при подаче синхросигналов на разные входы регистра. Регистр сдвига может быть спроектирован и таким образом, чтобы выполнять сдвиг одновременно не на одну, а на несколько позиций.

Регистры сдвига, или сдвигающие регистры, могут быть выполнены на синхронных RS-, D- или JK-триггерах. Два примера представлены на рис. 10.2.

Любой сдвигающий регистр имеет вход последовательного ввода информации и выход последовательного вывода информации. Выходы (входы) параллельного вывода (ввода) информации могут отсутствовать. Когда на вход Clock поступает управляющий сигнал (перепад

из 0 в 1), в триггерах 0–3 происходит одновременно переписывание данных со входа триггера на его выход. Таким образом, данные в регистре сдвигаются вправо на одну позицию. Такие регистры называют еще регистрами FIFO (First Input – First Output). Это название отражает порядок перемещения информации в регистре (первым вошел – первым вышел). Помимо обычных, существуют *реверсивные* регистры сдвига. Они предназначены для сдвига как в одну, так и в другую сторону.

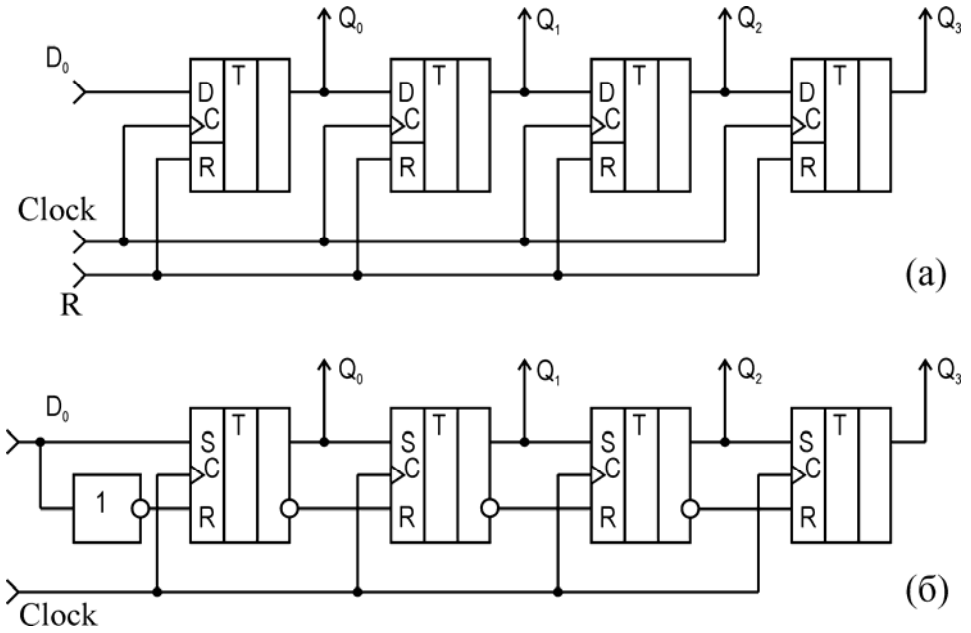


Рис. 10.2. Схема регистра сдвига на синхронных D-триггерах (*а*) и синхронных RS-триггерах (*б*)

На сдвигающих регистрах с последовательным входом можно построить запоминающее устройство с последовательным доступом информации. Хранение одного бита информации осуществляется в каждом отдельном триггере. Тактовый сигнал обеспечивает сдвиг данных. Рециркуляция информации обеспечивается подключением выхода регистра сдвига к его входу через мультиплексор 2×1 (см. рис. 10.3). Если $A = 0$, то при подаче импульсов на вход Clock обеспечивается рециркуляция информации, а если $A = 1$, то производится запись значения X в регистр сдвига.

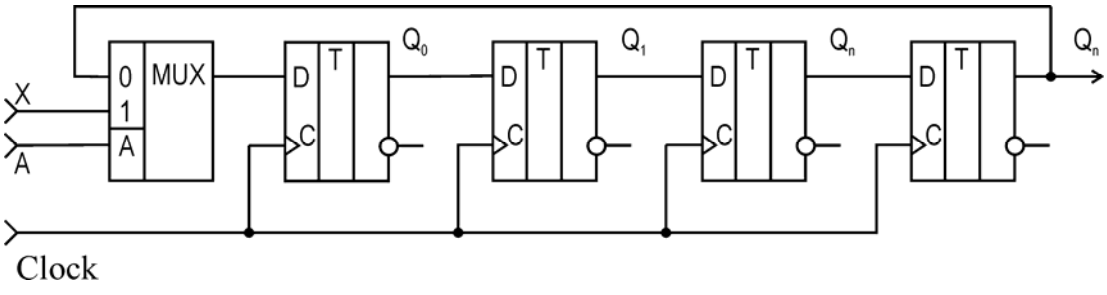


Рис. 10.3. Хранение информации в регистре сдвига

Таким образом, область применения регистров сдвига – это преобразование последовательного кода в параллельный и наоборот. Это нужно, например, при работе с диском (при записи преобразование параллельного кода в последовательный, при считывании – последовательного в параллельный), при работе с модемом, для других устройств последовательного ввода-вывода.

Кроме того, регистры сдвига могут использоваться при умножении чисел. Вспомните запись числа в двоичной системе счисления: если вы хотите умножить его на 10_2, это означает, что все цифры нужно сдвинуть влево на 1 разряд, а справа приписать 0. Рассмотрим процедуру умножения двух чисел в двоичной системе счисления:

$$\begin{array}{rcccc} & 1 & 0 & 1 \\ \times & & 1 & 1 \\ \hline & 1 & 0 & 1 \\ 1 & 0 & 1 & \\ \hline 1 & 1 & 1 & 1 \end{array}$$

Видно, что процедура умножения столбиком состоит из поразрядного умножения, сдвига и сложения. Таким образом, для реализации арифметического умножения двух чисел можно применить схему логического умножения, регистр сдвига и сумматор. Для деления необходимы, соответственно, действия вычитания и сдвига в обратном направлении. У некоторых микропроцессоров операции умножения и деления даже не включены в основную систему команд, и их приходится реализовывать программным способом.

Двоичный счет

Это одна из наиболее важных функций цифровой электроники. Подсчет числа импульсов логично выполнять в двоичной системе счисления. Основной элемент этой схемы – триггер, срабатывающий по фронту (или срезу) импульса. На рис. 10.4, *а* представлена одна из возможных схем двоичного счетчика.

Его работа основана на том, что JK-триггер изменяет состояние выхода, когда на вход *C* приходит тактовый импульс при условии $J = K = 1$. Если есть вход асинхронного сброса, то первоначально можно установить все триггеры в состояние «0». Временная диаграмма на рис. 10.4, *б* иллюстрирует принцип работы двоичного счетчика.

На выходах счетчика информация изменяется в зависимости от количества входных импульсов, что отражено в табл. 10.1. Отметим интересное свойство этой схемы: на выходе очередного триггера частота сигнала в два раза меньше, чем на входе. Поэтому двоичные счетчики применяются для деления частоты на число, кратное двум.

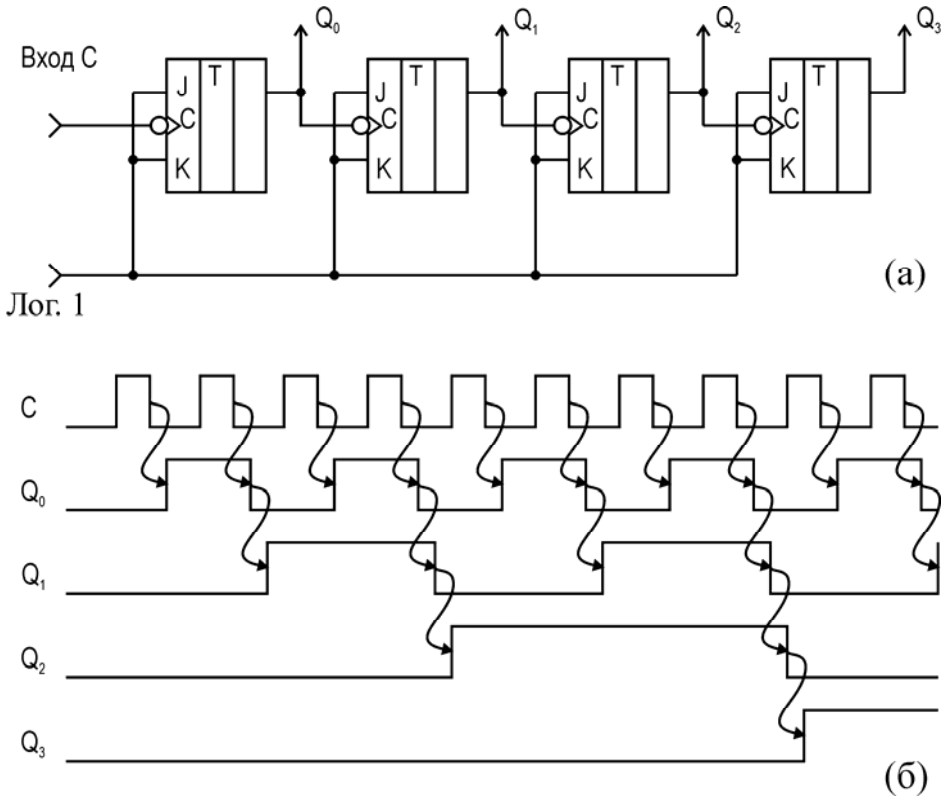


Рис. 10.4. Схема двоичного счетчика (*а*) и временные диаграммы его работы (*б*)

Таблица 10.1. Состояние выхода двоичного счетчика в зависимости от числа импульсов, поступивших на вход

Число входных импульсов	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

Запоминающие устройства

Каждой компьютерной системе крайне необходимо хранение и извлечение цифровой информации. Можно перечислить достаточно много физических сред, которые используются в компьютерах для запоминания информации: на ферритовых сердечниках, на магнитной ленте или другом магнитном слое, на основе оптоэлектроники, на транзисторах или твердотельных элементах интегральных микросхем и др.

По способу доступа запоминающие устройства (ЗУ) разделяют на:

- Память с произвольным доступом (RAM, Random Access Memory). Такое название связано с тем, что запоминаемые в этом устройстве данные могут быть использованы в порядке, отличном от порядка записи. Другое название этого вида памяти – ОЗУ (оперативное запоминающее устройство).
- Память, которая работает только на считывание информации. Это ROM (Read Only Memory) или ПЗУ (постоянное запоминающее устройство).
- Память с последовательным доступом к данным.
- Память с ассоциативным доступом (ассоциативное запоминающее устройство, АЗУ). Под АЗУ понимается ЗУ, обращение к информации в котором производится не по физическим адресам, а по содержанию произвольного количества разрядов чисел, хранящихся в ЗУ. При поиске информации происходит параллельное сравнение разрядов всех чисел данного ЗУ с некоторой внешней информацией (признаком), в результате которого за одно обращение обнаруживаются все числа, соответствующие заданному признаку опроса.

Иерархия в построении запоминающих устройств вычислительной системы связана с существенным различием в скорости и стоимости различных типов устройств. Как правило, объем памяти устройства обратно пропорционален его быстродействию и прямо пропорционален стоимости.

Скорость работы всей вычислительной системы существенно зависит от скорости обмена информацией запоминающих устройств и других узлов вычислительной системы. Поэтому при организации всей системы памяти цель проектировщика состоит в оптимизации расходов на построение вычислительной системы при достижении максимальной производительности (т. е. времени доступа к памяти). Основной принцип достижения этой цели состоит в том, что наиболее часто и активно используемые данные помещаются в наиболее быстро работающие системы памяти.

В связи с этим иерархия запоминающих устройств представляет собой несколько уровней, которые различаются принципом действия, объемом памяти и стоимостью.

Первый уровень иерархии – это внутренние регистры АЛУ и устройства управления. К этим регистрам осуществляется самый быстрый доступ, и по своему назначению они являются неотъемлемой частью логической схемы центрального процессора. К ним относятся: счетчик команд, регистры арифметического устройства, регистры прерываний. Не во всех машинах эти регистры доступны для программиста, но сами по себе они играют очень важную роль в функционировании компьютера.

К этому же уровню иерархии можно отнести регистры общего назначения (РОН). В большинстве машин они расположены в непосредственной близости от устройства управления и арифметико-логического устройства. Эти регистры служат для хранения промежуточных результатов вычислений и адресной информации. В принципе они служат для решения той же задачи, что и кэш, но в отличие от кэша, размещение информации в РОН может быть произведено программным путем. Особенно хорошо это заметно при программировании на ассемблере, в котором возможно прямое обращение к этим регистрам. В разных процессорах объем этой памяти разный, но всегда очень ограниченный. Например, в процессоре Intel 80286 всего 13 регистров, имеющих разрядность 16 бит, и 9 однобитных флажков.

Второй уровень – это кэш-память или сверхоперативная память. В процессорах многих типов кэш состоит из двух частей – внутренней и внешней, причем объем внешней кэш-памяти во много раз больше объема внутренней, в то время как по быстродействию она занимает промежуточное положение между внутренним кэшем и оперативной памятью. Смысл введения этого типа памяти состоит в сглаживании разницы между высоким быстродействием процессора и низким быстродействием основной памяти.

В простейшем случае алгоритм работы кэш-памяти может быть описан следующими основными положениями.

1. При появлении из устройства управления адреса обращения к основной оперативной памяти в схеме управления кэш-памятью производится проверка, есть ли данные с этим адресом в регистрах кэш-памяти.
2. Если данные с нужным адресом есть, то они считываются из кэш-памяти или записываются в нее (в зависимости от того, какое действие необходимо процессору).
3. Если в кэш-памяти нет данных с запрашиваемым адресом, то производится обращение к основной памяти. При считывании одновременно с доставкой данных в процессор они размещаются в свободном месте кэш-памяти. При проведении операции записи данные отправляются в основную память и записываются в свободное место кэш-памяти.
4. Если в кэш-памяти нет свободного места, то схема управления производит анализ данных и переписывает устаревшие данные в основную память, освобождая место для новых данных.

Здесь описаны только основные принципы работы кэш-памяти. В реальных устройствах кэш-память организована более сложно, поскольку при ее использовании необходимо решить ряд принципиальных и технических проблем. Главная проблема состоит в необходимости строгого соответствия информации в основной памяти и кэш-памяти. Из приведенного алгоритма ясно, что в некоторые моменты времени данные с одинаковыми адресами в кэше и основной памяти могут различаться. Например, при записи вычисленных данных они записываются в кэш, и только с некоторой задержкой – в основную память. При работе в многопроцессорной системе другой процессор может взять из основной памяти устаревшую информацию, что может привести к неверному решению общей задачи, решаемой несколькими процессорами. Проблема сохранения соответствия информации в основной памяти и кэш-памяти является очень важной и получила название проблемы когерентности кэш-памяти в многопроцессорных вычислительных системах.

Третий уровень – это основная оперативная память. Как правило, она имеет значительно больший объем, чем память предыдущих уровней, и расположена на отдельной плате, вне кристалла микропроцессора. В связи с этим при обмене информацией между процессором и памятью возникают задержки, приводящие к увеличению времени доступа. Время доступа к основной памяти обычно в несколько раз превышает время доступа к внутренним регистрам и кэшу.

На четвертый уровень можно отнести внешние запоминающие устройства. К ним можно отнести жесткий диск (винчестер), оптические диски (CD, DVD), магнитооптические диски, стримеры (накопители на магнитной ленте) и множество других типов памяти.

Дисковые накопители могут быть оптическими, магнитными и смешанными. К магнитным относят накопители на жестких или гибких магнитных дисках. Винчестеры, НЖМД – это несъемные жесткие магнитные диски. Емкость современных винчестеров от единиц до нескольких сотен гигабайт. На современных компьютерах это основной вид внешней памяти. Накопители на гибких магнитных дисках (флоппи-дисководы, НГМД) – устройства для записи и считывания информации с небольших съемных магнитных дисков (дискет), упакованные в пластиковый конверт (гибкий – у 5.25 дюймовых дискет и жесткий у 3.5 дюймовых). Максимальная емкость 5.25 дюймовой дискеты – 1.2 Мбайт, а 3.5 дюймовой дискеты – 1.44 Мбайт. В настоящее время 5.25 дюймовые дискеты морально устарели и не используются. Дискеты диаметром 3.5 дюйма также выходят из употребления (в первую очередь в силу ненадежности хранения).

Оптические диски (CD-ROM – Compact Disk Read Only Memory) – компьютерные устройства для чтения с компакт-дисков. CD-ROM диски получили распространение вслед за аудио-компакт-дисками. Это

пластиковые диски с напылением тонкого слоя светоотражающего материала, на поверхности которых информация записана с помощью лазерного луча. Лазерные диски являются наиболее популярными съемными носителями информации. При размерах 12 см в диаметре их емкость достигает 700 Мб. В настоящее время все более популярным становится формат компакт-дисков DVD-ROM, позволяющий при тех же размерах носителя разместить информацию объемом 4.3 Гб. Более современные технологии дают возможность удвоить этот объем.

К электронным внешним запоминающим устройствам можно отнести флэш-память – особый вид энергонезависимой перезаписываемой полупроводниковой памяти. Энергонезависимость означает, что не требуется дополнительной энергии для хранения данных (энергия требуется только для записи). Этот вид памяти стал особенно популярным в последнее время, поскольку очень удобен в применении и не боится механических воздействий. Рассмотрим более подробно принцип работы флэш-памяти. Справедливости ради следует сказать, что эта память может быть также и внутренней. В этом качестве флэш-память может быть использована для хранения BIOS. С другой стороны, по объему флэш-память превосходит ОЗУ и вполне может быть использована в качестве внешнего накопителя.

В зависимости от объема памяти и уровня ее иерархии в вычислительной системе могут варьироваться способы создания запоминающих устройств. Для хранения небольшого объема информации (как это необходимо во внутренних регистрах или регистрах общего назначения) можно использовать регистры данных на D-триггерах или других бистабильных устройствах.

Для хранения большого объема информации изготавливают большое число элементов памяти на одном кристалле и обеспечивают возможность доступа к каждому из них для записи и чтения информации. Обычно это относится к основной оперативной памяти.

Различают статическую и динамическую память. Статическое запоминающее устройство представляет собой набор триггеров, содержимое которых остается неизменным до тех пор, пока подключено питающее напряжение. В ячейках динамической памяти для запоминания информации используется заряд конденсатора. В связи с тем, что токи утечки являются существенными для такой схемы, информацию в ней нужно регулярно обновлять (поддерживать заряд конденсатора).

Рассмотрим работу блока статической памяти (см. рис. 10.5). Каждой ячейке присвоен индивидуальный адрес. На общий вход данных приходят данные, но для записи их в ячейку памяти должны выполниться два условия: первое – должен быть установлен сигнал от дешифратора адреса, а второе – должен быть подан сигнал разрешения записи. В этом случае на верхнем входе элемента И в левой части рисунка – логическая единица, следовательно, когда сигнал разрешения

записи поднимается в единицу, на выходе элемента И происходит переход из логического 0 в логическую 1, он подается на строб-вход триггера, и информация со входа записывается в триггер. При этом в остальных ячейках изменений не происходит, поскольку на соответствующих им выходах дешифратора – логические нули. Аналогично происходит считывание информации. При этом работает правый элемент И.

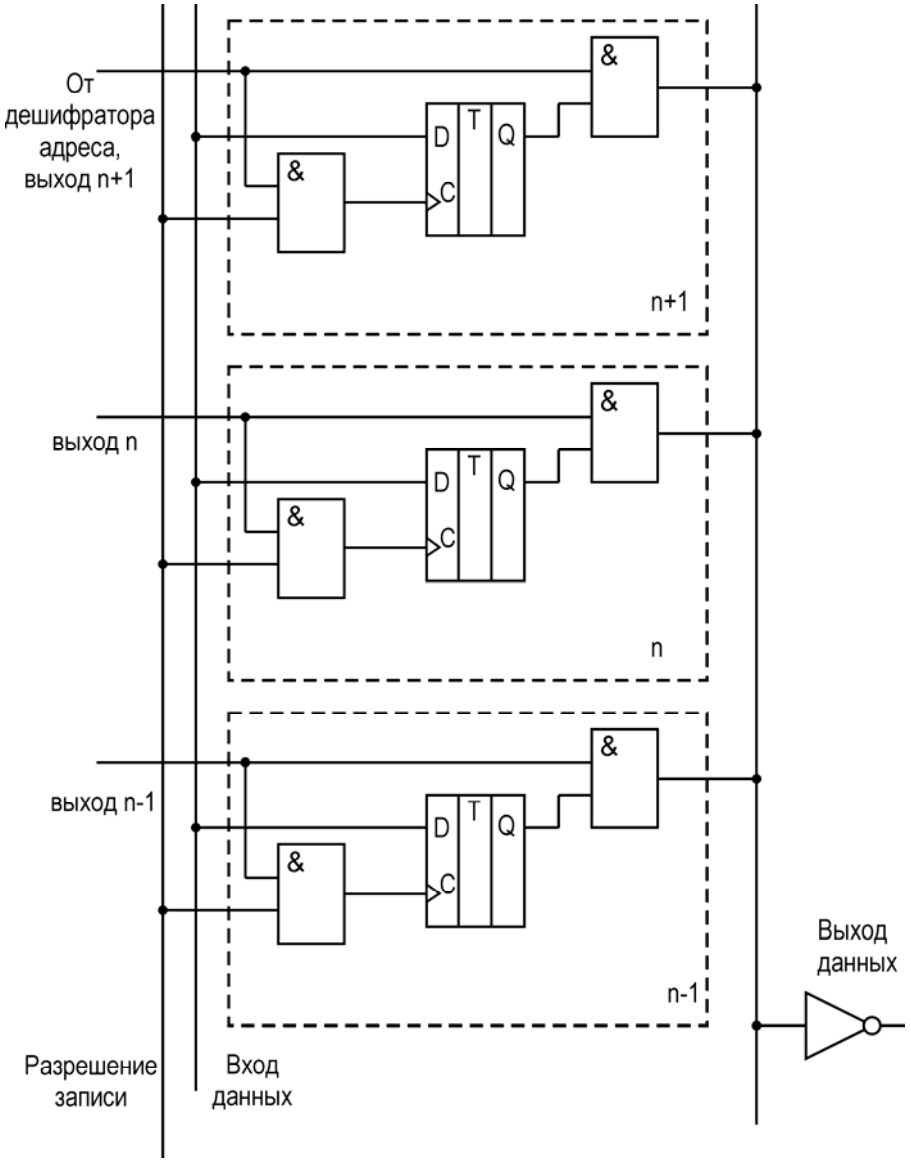


Рис. 10.5. Организация статической памяти

Наиболее распространенной в настоящее время является *динамическая* память. Рассмотрим более подробно принцип ее работы. В этой разновидности оперативной памяти информация хранится в виде заряда на конденсаторе, подключенном к затвору МОП-транзистора. Для создания одной ячейки памяти в этом случае требуется один конденсатор и один МОП-транзистор, поэтому реальная динамическая память получается более компактной, чем статическая. Работу транзисторов в ключевом режиме мы обсуждали ранее в лекции 5. Принцип работы ячейки

динамической памяти поясняется на рис. 10.6. Когда конденсатор заряжен до напряжения, равного логической 1, транзистор открыт, по выходной цепи течет ток и напряжение V_{out} близко к 0 (логический 0). Если конденсатор разряжен (напряжение на конденсаторе равно логическому 0), то транзистор закрыт и напряжение на выходе соответствует логической 1. Следует отметить, что конденсатор может сохранять заряд в течение длительного времени, и таким образом, можно сказать, что информация хранится в заряде конденсатора.

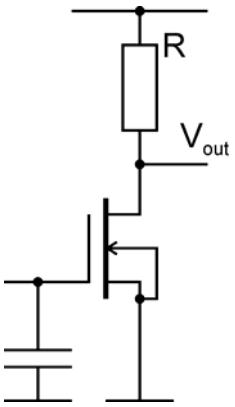


Рис. 10.6. Ячейка динамической памяти

Динамическое запоминающее устройство большой емкости может быть построено с использованием принципа адресации, как это было сделано для статического запоминающего устройства на рис. 10.5. Пример построения схемы динамического запоминающего устройства представлен на рис. 10.7.

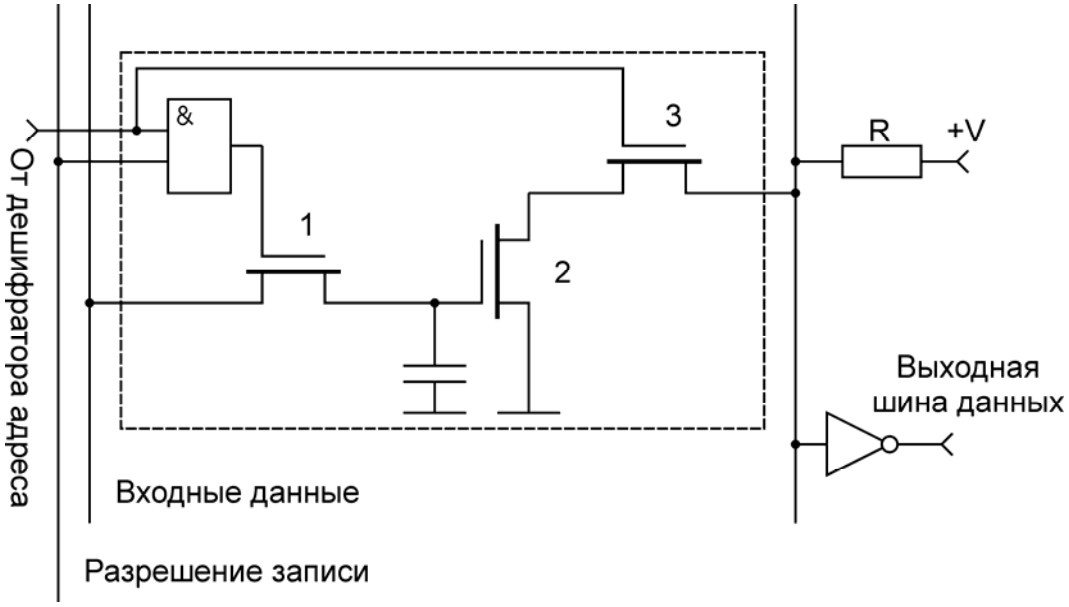


Рис. 10.7. Динамическое запоминающее устройство
(показана одна ячейка)

При поступлении сигнала от дешифратора адреса открывается ключ 3, и информация может быть считана из запоминающего устройства. В случае, когда выставлен еще и бит разрешения записи, ключи 1 и 3 открыты одновременно и входные данные записываются в данную ячейку памяти.

К недостаткам этого типа памяти можно отнести то свойство, что заряд на емкости не хранится бесконечно долго, поскольку у нее есть время саморазряда и дополнительно она разряжается через подключенные элементы. В связи с этим необходимо примерно каждые 1–2 мсек прочитать данные из памяти и снова записать в нее эту же информацию. Этот процесс называется регенерацией и требует некоторых временных затрат. Именно поэтому динамические ОЗУ медленнее статических. С другой стороны, затраты на его изготовление меньше, оно занимает меньшую площадь при той же информационной емкости. Поэтому обычно динамическое ОЗУ применяют в качестве RAM, а статическое – как кэш-память и регистры данных.

Контрольные вопросы и задания

1. Для чего служат регистры данных и как они устроены?
2. Что такое регистр сдвига? Как строится регистр сдвига и для чего его можно использовать?
3. Опишите принцип работы двоичного счетчика.
4. Чем различаются ячейки статической и динамической памяти?

Список литературы к лекциям 5–10

1. *Шоломов Л.А*. Основы теории дискретных логических и вычислительных устройств. – М. : Наука, Гл. ред. физ.-мат. лит., 1980. – 400 с.
2. *Фрике К*. Вводный курс цифровой электроники. М. : Техносфера, 2003. – 432 с.
3. *Угрюмов Е.П*. Цифровая схемотехника. – СПб. : БХВ-Петербург, 2004. – 528 с. : ил.

Лекция 11. ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Простейшие ПЗУ

В русском языке термин Постоянное Запоминающее Устройство (ПЗУ) используется как эквивалент английского термина ROM (Read Only Memory) – память только для чтения.

Строго говоря, к данному виду памяти можно отнести только две разновидности: Mask-ROM (Масочные ПЗУ) и PROM (Programmable ROM), или однократно Программируемые ПЗУ.

Другие типы памяти – EPROM, EEPROM, Flash-ROM – только исторически произошли от полупроводникового ROM. Однако эти виды памяти никак не могут быть ROM, поскольку ROM переводится как «память только для чтения» и ни о какой возможности перезаписи в ROM речи быть не может. Тем не менее эти типы памяти классифицируются как ROM. Небольшая неточность не обращала на себя внимания, однако с развитием технологий, когда флэш-память стала выдерживать до 1 миллиона циклов перезаписи и стала использоваться как накопитель общего назначения, этот недочет в классификации начал бросаться в глаза. По своим функциональным характеристикам память EPROM, EEPROM и Flash относятся к классу энергонезависимой перезаписываемой памяти (английский эквивалент – NonVolatile Read-Write Memory или NVRWM).

ROM память устроена в виде матрицы, каждая ячейка которой имеет свой адрес и может кодировать один бит информации. Данные на масочные ПЗУ записывались во время производства путем нанесения по маске (отсюда и название) алюминиевых соединительных дорожек литографическим способом. Наличие или отсутствие в соответствующем месте такой дорожки кодировало «0» или «1». Содержимое Mask-ROM после того, как она произведена, изменить невозможно. Этот подход к созданию схем является индивидуальным и для каждой новой задачи потребуется изготовление нового ПЗУ. Тем не менее преимущества такой технологии очевидны – это низкая стоимость готовой микросхемы (правда, это проявляется только при больших объемах производства), высокая скорость доступа к ячейкам памяти, а также высокая надежность готовой микросхемы и устойчивость к электромагнитным полям.

PROM (Programmable ROM), или однократно Программируемые ПЗУ. В качестве ячеек памяти в данном типе памяти использовались

плавкие перемычки. В отличие от Mask-ROM, в PROM появилась возможность кодировать («пережигать») перемычки при наличии специального программатора, что дало возможность пользователям записывать в ПЗУ любую информацию. В каждой отдельной ячейке PROM плавкая перемычка либо разрушалась (запись логического 0) путем подачи достаточно высокого напряжения (значительно больше стандартного напряжения питания), либо оставлялась без изменений (логическая 1). Возможность самостоятельной записи информации в PROM сделало их пригодными для штучного и мелкосерийного производства. Тем не менее PROM практически полностью вышли из употребления в конце 80-х годов XX века.

Схема простейшего программируемого ПЗУ состоит из дешифратора $n \times 2^n$ и схемы ИЛИ, к входам которой через плавкие перемычки подключены выходы дешифратора

$$DO = \sum_{i=0}^{2^n-1} a_i K_i(v),\ K_i(v) = \prod_{p=1}^{n} x_p^{e_p},\ v = (x_n ... x_1),\ K_i(v) - \text{минтерм},\ i = e_n ... e_1.$$

Функция *DO* по форме совпадает с СДНФ, т. е. с помощью ПЗУ, имеющего *n* адресных входов x_p, можно реализовать любую функцию *n* переменных.

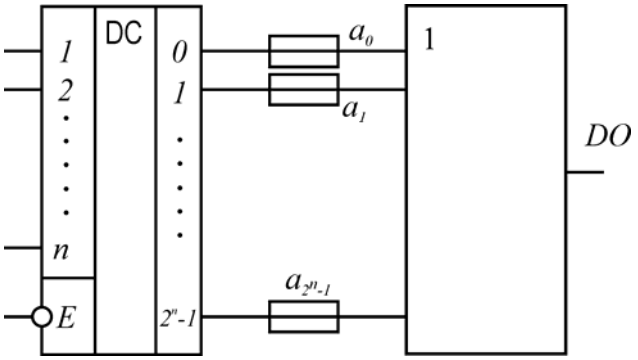


Рис. 11.1. Постоянное запоминающее устройство
с плавкими перемычками

EPROM

В различных источниках эта аббревиатура расшифровывается поразному. В одних случаях ее называют как Erasable Programmable ROM (стираемые программируемые ПЗУ), в других – как Electrically Programmable ROM (электрически программируемые ПЗУ). Обе этих расшифровки отражают качественные отличия EEPROM. Этот тип памяти программируется электрическим сигналом, но для того, чтобы записать новую информацию, необходимо произвести стирание. Стирание ячеек EPROM выполняется сразу для всей микросхемы посредством облучения чипа ультрафиолетовыми или рентгеновскими лучами в течение нескольких минут. Микросхемы, стирание которых производится путем засвечивания ультрафиолетом, были разработаны Intel

в 1971 году, и носят название UV-EPROM (приставка UV (Ultraviolet) – ультрафиолет). Они содержат окошки из кварцевого стекла, которые по окончании процесса стирания заклеивают.

EPROM от Intel была основана на МОП-транзисторах с лавинной инжекцией заряда (FAMOS – Floating Gate Avalanche injection Metal Oxide Semiconductor, русский эквивалент – ЛИЗМОП). В первом приближении такой транзистор представляет собой конденсатор с очень малой утечкой заряда. Позднее, в 1973 году, компания Toshiba разработала ячейки на основе SAMOS (Stacked gate Avalanche injection MOS, по другой версии – Silicon and Aluminum MOS) для EPROM памяти, а в 1977 году Intel разработала свой вариант SAMOS.

В EPROM стирание приводит все биты стираемой области в одно состояние (обычно во все единицы, реже – во все нули). Запись на EPROM, как и в PROM, осуществляется при помощи специальных программаторов (они, однако, отличаются от программаторов для PROM). Появление этого типа памяти значительно расширило возможности разработчиков. Тем не менее EPROM свойственны некоторые недостатки – небольшое количество циклов перезаписи, невозможность стирания части хранимых данных. Кроме того, обращаться с ними было непросто – с одной стороны, существует высокая вероятность не до конца удалить имеющиеся данные, что в конечном итоге приведет к сбоям, а с другой стороны, есть вероятность передержать микросхему под ультрафиолетом, что может уменьшить срок службы микросхемы и даже привести к ее полной негодности. В настоящее время EPROM практически полностью вытеснена с рынка EEPROM и Flash.

EEPROM

(Electronically EPROM) – электрически стираемые ППЗУ были разработаны в 1979 году в той же Intel. В 1983 году вышел первый 16 Кбит образец, изготовленный на основе FLOTOX-транзисторов (Floating Gate Tunnel-OXide – «плавающий» затвор с туннелированием в окисле). Флэш-память является разновидностью EEPROM, но в ней используется несколько отличный от EEPROM тип ячейки-транзистора.

Главной отличительной особенностью EEPROM (в т. ч. Flash) от ранее рассмотренных нами типов энергонезависимой памяти является возможность перепрограммирования при подключении к стандартной системной шине микропроцессорного устройства. В EEPROM появилась возможность производить стирание отдельной ячейки при помощи электрического тока. Для EEPROM стирание каждой ячейки выполняется автоматически при записи в нее новой информации, т. е. можно изменить данные в любой ячейке, не затрагивая остальные. Процедура стирания обычно существенно более длительная, чем процедура записи.

Флэш-память

Этот тип памяти можно выделить в отдельный класс. В отличие от многих других типов полупроводниковой памяти, ячейка флэш-памяти не содержит конденсаторов – типичная ячейка флэш-памяти состоит всего-навсего из одного транзистора особой архитектуры. Ячейка флэш-памяти прекрасно масштабируется, что достигается не только благодаря успехам в миниатюризации размеров транзисторов, но и благодаря конструктивным находкам, позволяющим в одной ячейке флэш-памяти хранить несколько бит информации.

Флэш-память исторически происходит от ROM-памяти, но функционирует подобно RAM. Данные флэш хранит в ячейках памяти, похожих на ячейки в динамической RAM. В отличие от RAM, при отключении питания данные из флэш-памяти не пропадают. Это выгодно отличает флэш-память, однако замены памяти RAM не происходит из-за двух особенностей флэш-памяти: флэш работает существенно медленнее и имеет ограничение по количеству циклов перезаписи (от 10 тыс. до 1 млн для разных типов).

Информация, записанная на флэш-память, может храниться очень длительное время (от 20 до 100 лет) и способна выдерживать значительные механические нагрузки (в 5–10 раз превышающие предельно допустимые для обычных жестких дисков). Основное преимущество флэш-памяти перед жесткими дисками и носителями CD-ROM состоит в том, что флэш-память потребляет значительно (примерно в 10–20 и более раз) меньше энергии во время работы. В устройствах CD-ROM, жестких дисках, кассетах и других механических носителях информации бóльшая часть энергии уходит на приведение в движение механики этих устройств. Кроме того, флэш-память компактнее большинства других механических носителей.

Благодаря низкому энергопотреблению, компактности, долговечности и относительно высокому быстродействию, флэш-память идеально подходит для использования в качестве накопителя в таких портативных устройствах, как цифровые фото- и видеокамеры, сотовые телефоны, портативные компьютеры, MP3-плееры, цифровые диктофоны и т. п.

Ее полное историческое название – Flash Erase EEPROM. Изобретение флэш-памяти зачастую незаслуженно приписывают Intel, называя при этом 1988 год. На самом деле память впервые была разработана компанией Toshiba в 1984 году, и уже на следующий год было начато производство 256 Кбит микросхем flash-памяти в промышленных масштабах. В 1988 году Intel разработала собственный вариант флэш-памяти.

Основное отличие флэш-памяти от обычной EEPROM заключается в том, что стирание содержимого ячеек выполняется либо для всей микросхемы, либо для определенного блока (кластера, кадра или стра-

ницы). Обычный размер такого блока составляет 256 или 512 байт, однако в некоторых видах флэш-памяти объем блока может достигать 256 КБ. Следует заметить, что существуют микросхемы, позволяющие работать с блоками разных размеров (для оптимизации быстродействия). Стирать можно как блок, так и содержимое всей микросхемы сразу. Таким образом, в общем случае, для того, чтобы изменить один байт, сначала в буфер считывается весь блок, где содержится подлежащий изменению байт, стирается содержимое блока, изменяется значение байта в буфере, после чего производится запись измененного в буфере блока. Такая схема существенно снижает скорость записи небольших объемов данных в произвольные области памяти, однако значительно увеличивает быстродействие при последовательной записи данных большими порциями.

Организация flash-памяти

Ячейки флэш-памяти бывают как на одном, так и на двух транзисторах.

В простейшем случае каждая ячейка хранит один бит информации и состоит из одного полевого транзистора со специальной электрически изолированной областью («плавающим» затвором – floating gate), способной хранить заряд многие годы. Наличие или отсутствие заряда кодирует один бит информации. Принцип конструирования ячеек флэш-памяти представлен на рис. 11.2.

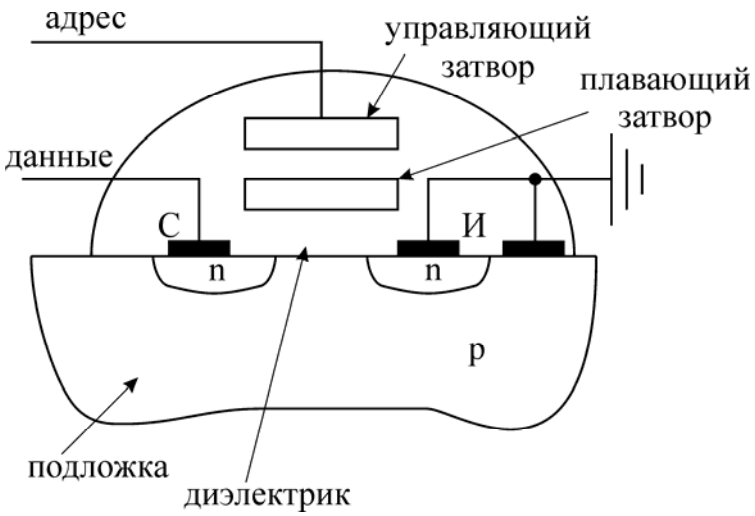


Рис. 11.2. Ячейка флэш-памяти

При записи заряд помещается на плавающий затвор одним из двух способов (зависит от типа ячейки): методом инжекции «горячих» электронов или методом туннелирования электронов. Стирание содержимого ячейки (снятие заряда с «плавающего» затвора) производится методом туннелирования.

Как правило, наличие заряда на транзисторе понимается как логический «0», а его отсутствие – как логическая «1».

Рассмотрим простейшую ячейку флэш-памяти на одном n-p-n транзисторе. Ячейки подобного типа чаще всего применялись в накопителях flash-памяти, а также в микросхемах EPROM.

Поведение транзистора зависит от количества электронов на «плавающем» затворе. «Плавающий» затвор играет ту же роль, что и конденсатор в DRAM, т. е. хранит запрограммированное значение.

Помещение заряда на «плавающий» затвор в такой ячейке производится методом инжекции «горячих» электронов (CHE – channel hot electrons), а снятие заряда осуществляется методом квантовомеханического туннелирования Фаулера-Нордхейма (Fowler-Nordheim, FN).

При чтении, в отсутствие заряда на «плавающем» затворе, под воздействием положительного поля на управляющем затворе образуется *n*-канал в подложке между истоком и стоком, и возникает ток.

Наличие заряда на «плавающем» затворе меняет вольт-амперные характеристики транзистора таким образом, что при обычном для чтения напряжении канал не появляется, и тока между истоком и стоком не возникает.

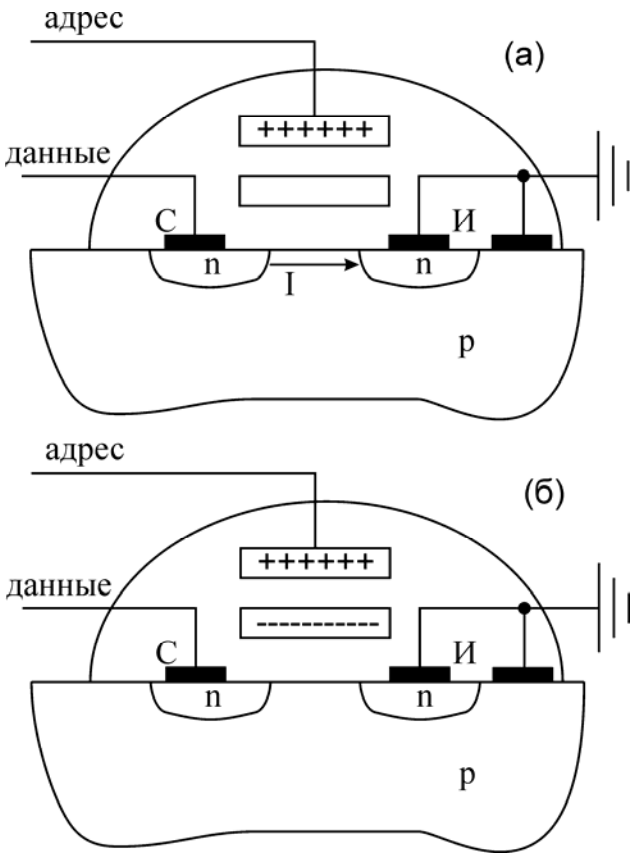


Рис. 11.3. Принцип считывания информации в ячейке флэш-памяти

При программировании на сток и управляющий затвор подается высокое напряжение (причем на управляющий затвор напряжение подается приблизительно в два раза выше). «Горячие» электроны из канала инжектируются на плавающий затвор и изменяют вольт-амперные

характеристики транзистора. Такие электроны называют «горячими», поскольку они обладают высокой энергией, достаточной для преодоления потенциального барьера, создаваемого тонкой пленкой диэлектрика.

При стирании высокое напряжение подается на исток. На управляющий затвор (опционально) подается высокое отрицательное напряжение. Электроны туннелируют на исток.

Эффект туннелирования – один из эффектов, использующих волновые свойства электрона. Сам эффект заключается в преодолении электроном потенциального барьера малой «толщины». Для наглядности представим себе структуру, состоящую из двух проводящих областей, разделенных тонким слоем диэлектрика (обедненная область). Преодолеть этот слой обычным способом электрон не может – не хватает энергии. Но при создании определенных условий (соответствующее напряжение и т. п.) электрон проскакивает слой диэлектрика (туннелирует сквозь него), создавая ток.

Различия методов туннелирования Фаулера – Нордхейма (FN) и метода инжекции «горячих» электронов:

- Channel FN tunneling – не требует большого напряжения. Ячейки, использующие FN, могут быть меньше ячеек, использующих CHE.
- CHE injection (CHEI) – требует более высокого напряжения по сравнению с FN. Таким образом, для работы памяти требуется поддержка двойного питания.
- Программирование методом CHE осуществляется быстрее, чем методом FN.

Следует заметить, что кроме FN и CHE существуют другие методы программирования и стирания ячейки, которые успешно используются на практике, однако два описанных нами применяются чаще всего.

Процедуры стирания и записи сильно изнашивают ячейку флэш-памяти, поэтому в устройствах флэш-памяти помимо самой микросхемы памяти дополнительно используют специальную микросхему-контроллер, которая управляет процессом стирания-записи и обеспечивает равномерное использование различных ячеек памяти. Права, в последнее время все чаще производители устройств флэш-памяти такой контроллер не используют, что приводит к снижению надежности.

Кроме наиболее часто встречающихся ячеек с «плавающим» затвором существуют также ячейки на основе SONOS-транзисторов, которые не содержат плавающего затвора. SONOS-транзистор напоминает обычный МНОП (MNOS) транзистор. В SONOS-ячейках функцию «плавающего» затвора и окружающего его изолятора выполняет композитный диэлектрик ONO. Расшифровывается SONOS (Semiconductor Oxide Nitride Oxide Semiconductor) как Полупроводник-Диэлектрик-

Нитрид-Диэлектрик-Полупроводник. Вместо давшего название этому типу ячейки нитрида в будущем планируется использовать поликристаллический кремний.

Многоуровневые ячейки

В последнее время многие компании проводят исследования микросхем флэш-памяти, в которых одна ячейка хранит не один, а два и более бит. Такие ячейки называются многоуровневыми (в английском варианте MLC – Multi Level Cell). Флэш-память, в которой используются двухбитовые ячейки, уже анонсированы. Кроме того, уже известно, что в лабораторных условиях получены прототипы, хранящие 4 бита в одной ячейке. В настоящее время активно ведутся исследования, связанные с поиском предельного числа бит, которое способна хранить многоуровневая ячейка.

В технологии MLC используется аналоговая природа ячейки памяти. Как известно, обычная однобитная ячейка памяти может принимать два состояния – «0» или «1». Во флэш-памяти эти два состояния различаются по величине заряда, помещенного на «плавающий» затвор транзистора. В отличие от «обычной» флэш-памяти, MLC способна различать более двух величин зарядов, помещенных на «плавающий» затвор, и, соответственно, большее число состояний. При этом каждому состоянию в соответствие ставится определенная комбинация значений бит. Например, для того, чтобы емкость ячейки была равна двум битам, необходимо различать четыре уровня заряда.

Во время записи на «плавающий» затвор помещается количество заряда, соответствующее необходимому состоянию. От величины заряда на «плавающем» затворе зависит пороговое напряжение транзистора. Пороговое напряжение транзистора можно измерить при чтении и определить по нему записанное состояние, а значит и записанную последовательность бит.

Идеология построения MLC микросхем определяет их основное преимущество по сравнению с обычными микросхемами флэш-памяти: при равном размере микросхем и одинаковом техпроцессе «обычной» и MLC-памяти, последняя способна хранить больше информации (размер ячейки тот же, а количество хранимых в ней бит – больше).

К основным недостаткам MLC можно отнести следующие:

- снижается помехоустойчивость этих ячеек и, соответственно, надежность хранения по сравнению с однобитными ячейками. Это приводит к необходимости встраивать более сложный механизм коррекции ошибок, что в свою очередь приводит к усложнению схемы;
- быстродействие микросхем на основе MLC зачастую ниже, чем у микросхем на основе однобитных ячеек;

- поскольку компьютерная система целиком работает с двоичными цифровыми сигналами, необходимо разработать и встроить в устройство памяти специфические схемы чтения/записи многоуровневых ячеек.

Контрольные вопросы и задания

1. Какие различные виды постоянных запоминающих устройств существуют?
2. Как устроена ячейка флэш-памяти?
3. В чем достоинства и недостатки флэш-памяти?

Лекция 12. ОСОБЕННОСТИ АРХИТЕКТУРЫ СОВРЕМЕННЫХ ЭВМ

Современная вычислительная техника может состоять не только из отдельных вычислительных машин. Существуют целые мультипроцессорные и многомашинные вычислительные комплексы, которые позволяют проводить параллельную обработку данных. К вычислительным машинам относят и небольшие наладонные компьютеры, и персональные компьютеры, и суперкомпьютеры, потребляющие огромное количество энергии. Наиболее распространенными в настоящее время являются персональные компьютеры, которые в основном являются однопроцессорными. Сфера их использования простирается от решения игровых задач до профессиональных математических расчетов и управления измерительными приборами. Рассмотрим структуру типовой однопроцессорной вычислительной машины.

Блок-схема современной однопроцессорной ЭВМ

Однопроцессорную ЭВМ без периферийных узлов можно условно представить в виде четырех частей: память, устройство управления, арифметико-логическое устройство, устройство сопряжения (предназначенное для управления устройствами ввода-вывода). Эта структура в целом соответствует машине фон Неймана (лекция 2).

Независимо от того, для какой цели применяется компьютер, принцип его действия один и тот же: он выполняет последовательность действий по передаче данных в соответствующее место в надлежащий момент времени. Эта последовательность действий определяется программой, которая представляет собой последовательность кодов инструкций, исполняемых компьютером в определенном порядке.

Структура типовой однопроцессорной ЭВМ представлена на рис. 12.1. Четыре составных части компьютера показаны здесь более детально. Память можно разделить на оперативную (память с произвольным доступом, RAM) и постоянную (ROM). Устройство сопряжения включает в себя входные и выходные регистры и служит для организации обмена с внешними устройствами. Центральный процессор выполнен, как правило, в виде одной интегральной микросхемы, называемой микропроцессором и содержащей арифметико-логическое устройство (АЛУ), несколько регистров общего назначения, кэш-память, блок операций с плавающей точкой и устройство управления. В состав устройства управления входят программный счетчик, регистр команд и дешифратор.

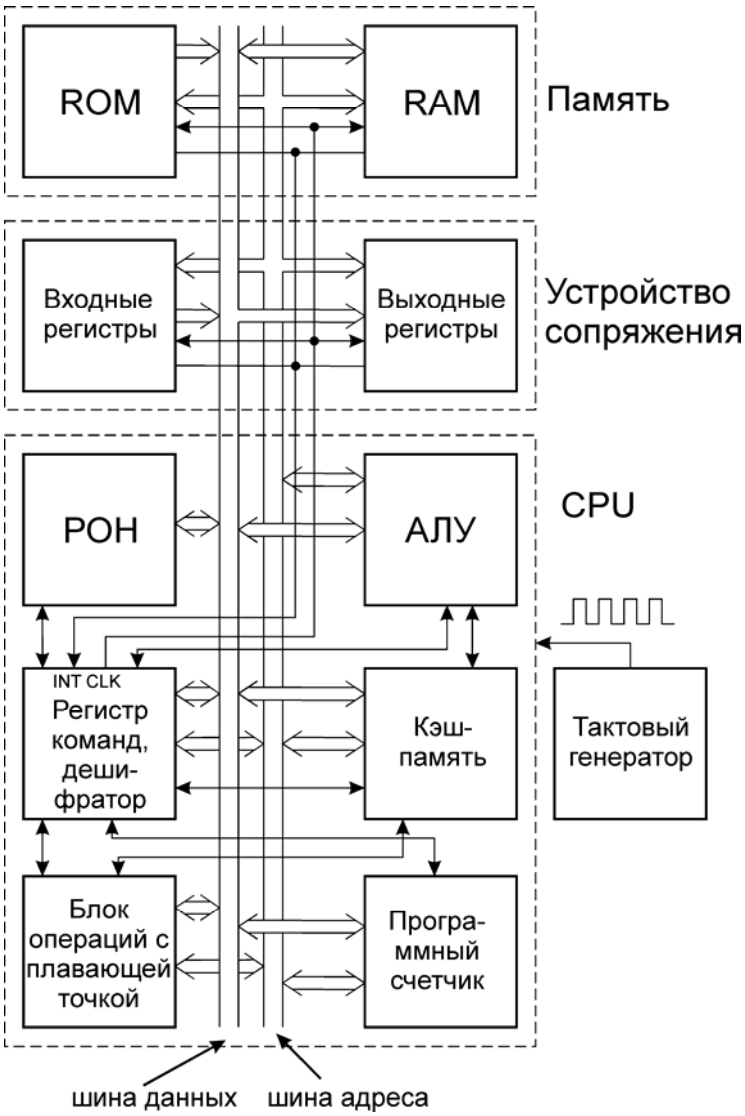


Рис. 12.1. Блок-схема типовой однопроцессорной ЭВМ

Рассмотрим логическое взаимодействие отдельных узлов при работе компьютера. Тактовый генератор необходим для синхронизации работы всех узлов компьютера в целом. Он вырабатывает тактовые импульсы, частота которых определяет скорость вычислений всей системы. Устройство управления выставляет подходящие адреса на адресной шине, выбирает команды, производит их дешифрацию и осуществляет взаимодействие между АЛУ, нужными регистрами и ячейками памяти. Программный счетчик по мере выполнения команд изменяет свое состояние в соответствии с алгоритмом и таким образом управляет последовательностью выполнения команд. Кроме того, устройство управления непосредственно синхронизирует (сигнал CLK, Clock или часы) работу отдельных узлов ЭВМ, используя сигнал тактового генератора. Кроме того, устройство управления реагирует на сигналы прерываний (INT, Interrupt), приходящие от различных узлов компьютера. Сигналы прерываний в ЭВМ передаются в устройство управления от внешних

по отношению к процессору устройств. Обработка сигналов прерываний позволяет процессору обратить внимание на особые условия, возникающие в нем самом или в другом оборудовании и переключиться на выполнение другой программы. Такая организация позволяет выполнять задания в соответствии с их важностью. Не дожидаясь окончания работы программы, процессор выполнит вывод информации на дисплей или принтер, ввод информации с клавиатуры, регенерацию памяти и другие необходимые задачи.

Постоянное запоминающее устройство обеспечивает начальную загрузку компьютера, оперативная память хранит исполняемую программу и служит для хранения промежуточных данных. Устройство сопряжения передает в нужном формате данные к периферийным устройствам, принимает от них данные, преобразуя в свою очередь в нужный формат и в своих регистрах хранит промежуточные данные.

Следует отметить, что организация конкретных компьютеров и вычислительных устройств может отличаться. Обычно для сборки компьютера необходимо использовать не только процессор, но и отдельные микросхемы памяти, интерфейсы, буферные усилители и ряд других специализированных микросхем. Тем не менее в некоторых приложениях, где нет необходимости выполнять объемные программы, требующие больших объемов памяти и вычислительных ресурсов, очень полезны специальные микросхемы. Это однокристальные микроконтроллеры, содержащие в одной микросхеме память, устройство сопряжения, CPU, тактовый генератор, и они могут работать самостоятельно.

Большое значение имеет концепция взаимосвязи отдельных частей ЭВМ, структура или (как это принято называть в информатике) архитектура вычислительной системы. Архитектура ЭВМ значительно влияет на производительность и эффективность использования вычислительных машин.

Развитие архитектуры вычислительных машин

ЭВМ первого поколения представляли собой огромные сооружения с тысячами радиоламп и были совершенно ненадежными. Поэтому главной задачей проектировщиков было создание максимально простой системы с минимальными удобствами программирования. Эти ЭВМ программировались в машинных кодах. После появления алфавитно-цифровых устройств ввода-вывода начался интенсивный процесс разработки средств автоматизации программирования. Появились языки символьного кодирования, предшественники современного ассемблера.

Что же изменилось за время развития вычислительной техники? Прежде всего, появляются новые технологии изготовления электронных приборов, и как следствие, уменьшается потребление энергии и размеры, увеличивается быстродействие. С другой стороны, претерпевает изменение и структура ЭВМ, совершенствуются принципы

взаимодействия ее основных узлов. Современные сложные системы невозможно программировать в машинных кодах, поэтому растет необходимость в разработке языков высокого уровня, прикладных пакетов программ, методов передачи данных и других современных способов обработки возрастающих объемов информации.

В машинах второго поколения появились элементы страничной организации памяти и сочетание операций с фиксированной и плавающей точкой. Эти новые элементы были, в частности, применены в машине российского производства М-2 (конструктор М.А. Карцев). На базе модернизированного ее варианта М4-2М были построены первые кластеры, т. е. многомашинные вычислительные системы. Развиваются машинно-независимые языки (первым был Фортран, который использовался еще на ламповых машинах). Появляются такие специализированные трансляторы, как КОБОЛ (предназначенный для программирования экономических задач) и ЛИСП (для обработки списков).

В машинах третьего поколения при переходе элементной базы от транзисторов к интегральным схемам заметно возросло быстродействие процессора. Поэтому при выводе информации на медленное устройство (или вводе с медленного устройства) процессор фактически простаивал, что снижало его общую производительность. Для решения этой проблемы попробовали создать для управления внешними устройствами специальные схемы и освободить центральный процессор от «рутинной» работы и занять его вычислениями. Схемы для управления внешними устройствами также содержат свои элементы управления и памяти и, таким образом, имеют структуру специализированного компьютера, что позволяет им самостоятельно работать с внешними устройствами. В машинах серии IBM 370 (в социалистических странах их аналогами были ЕС ЭВМ) внешние устройства подключались при помощи специальных каналов (мультиплексный канал использовался для подключения медленных устройств, селекторные каналы – для более быстрых).

При этом центральный процессор подключается к работе в тех случаях, когда его вмешательство необходимо. Для того чтобы работа проходила корректно, была предложена система прерываний. Прерывание – это сигнал управления процессором, который сигнализирует о происхождении какого-либо события. Их можно разделить на внутренние и внешние (по отношению к процессору). Внешние прерывания формируются внешними устройствами, а внутренние генерирует сам процессор, если при выполнении последовательности команд произошло нечто неожиданное (например, деление на ноль). Для управления сигналами прерываний и регулирования работы процессора и периферийных устройств был введен контроллер прерываний.

При переходе к третьему поколению существенные изменения претерпела структура памяти. Минимальной единицей считываемой

или записываемой информации становится 1 байт (8 бит). Реализована идея конвейеризации обработки информации, высказанная впервые советским конструктором, академиком Лебедевым еще в 1957 году. Более подробно рассмотрим ее несколько позднее.

К этому же периоду следует отнести возникновение понятия базовых архитектур, связанных с фирмами – производителями компьютеров. На основе базовой архитектуры фирмы проектировали целые семейства процессоров различной производительности.

Проходило дальнейшее развитие языков программирования. Появились операционные системы.

Для машин четвертого поколения характерно разделение средств вычислительной техники на две ветви – персональные компьютеры и супер-ЭВМ, выпускаемые большими партиями. В силу своей массовости персональные компьютеры потребовали качественно отличающегося подхода к программному обеспечению. Необходимо было значительно упростить взаимодействие человека с компьютером, придать ему дружественный интерактивный характер. Появились операционные системы для персональных компьютеров (MS-DOS), затем графические Windows-оболочки.

К машинам пятого поколения можно отнести современные персональные компьютеры и большие многопроцессорные системы. В этих машинах качественно упрощен процесс общения с человеком, изображение и звук становятся неотъемлемым элементом интерактивного режима. Идет процесс объединения компьютеров в единое информационное пространство, создание распределенных баз данных и знаний. При этом полным ходом идет унификация элементной базы – большие суперкомпьютеры и персональные ЭВМ используют одни и те же микропроцессоры в качестве основного вычислительного блока.

Тем не менее следует отметить, что, несмотря на существенные изменения в быстродействии, элементной базе и программном обеспечении, современные компьютеры состоят из тех же основных блоков и работают на тех же принципах, которые были предложены фон Нейманом в 1946 году. Изменения в архитектуре связаны с оптимизацией взаимодействия основных узлов ЭВМ в связи с необходимостью решения задач, требующих возрастающей вычислительной мощности. Рассмотрим сущность основных изменений в архитектуре компьютера.

Отличия современного компьютера

К наиболее крупным изменениям структуры в современных компьютерах можно отнести следующие:

- наличие специального вида памяти – кэш-памяти;
- арифметический сопроцессор;
- наличие более одного вычислительного ядра;
- наличие общей шины;

- конвейерная обработка информации;
- наличие прерываний.

Современные ЭВМ содержат те же узлы, что и первые ЭВМ, но дополнены новыми узлами, а также по-новому, более эффективно, организуют взаимодействие составных частей.

Рассмотрим сущность некоторых из этих нововведений.

Кэш-память (или просто кэш) – это специальное запоминающее устройство, скрытое от пользователя и являющееся промежуточным между процессором и основной оперативной памятью. Функция кэш состоит в минимизации времени обращения к основной оперативной памяти. Это устройство обладает большим быстродействием, но значительно меньшим объемом по сравнению с оперативной памятью (подробнее см. лекцию 10).

В компьютерах четвертого поколения появилось также арифметические сопроцессоры – устройства, которые выполняют вычисления с плавающей запятой. На начальном этапе сопроцессор выполнялся в виде отдельной микросхемы, а сейчас он включен в состав процессора.

Многоядерность позволяет ускорить выполнение задач процессором. При наличии даже двух ядер эксплуатационные характеристики компьютера значительно улучшаются, поскольку появляется возможность выполнять одновременно два приложения. Процессору нет необходимости переключаться с одного приложения на другое, поскольку он вполне может выполнять их одновременно. Кроме того, в некоторых случаях многоядерная организация позволяет увеличивать скорость выполнения единственной задачи (по сравнению с аналогичным одноядерным процессором).

Обмен данными – очень важная функция компьютера, и на нее обращают большое внимание. Идея создания отдельных каналов в компьютерах третьего поколения была развита в компьютерах четвертого поколении, для которых характерно наличие общей шины. Общая шина – это сложное устройство, к которому подключаются процессоры, оперативная память, внешние запоминающие устройства, устройства ввода-вывода. Шина обеспечивает взаимный обмен информацией между различными устройствами. В процессоре также имеется внутренняя шина, соединяющая блоки самого процессора. По внутренней шине (или шинам) передаются команды, данные, адреса между различными блоками процессора (см. лекцию 8).

Конвейерная обработка данных

Рассмотрим более подробно конвейерную обработку данных. Первоначальная идея конвейера, давным-давно предложенная Генри Фордом, состоит в том, что производительность цепочки последовательных действий определяется не сложностью этой цепочки, а лишь длительностью самой сложной операции. Иными словами, совершенно неважно,

сколько человек занимаются производством автомобиля и как долго длится его изготовление в целом, – важно то, что если каждый человек в цепочке тратит, скажем, на свою операцию одну минуту, то с конвейера будет сходить один автомобиль в минуту, ни больше и ни меньше; независимо от того, сколько операций нужно совершить с отдельным автомобилем и сколько заняла бы его сборка одним человеком. Применительно к процессорам принцип конвейера означает, что если мы сумеем разбить выполнение машинной инструкции на несколько этапов, то скорость, с которой процессор забирает данные на исполнение и выдает результаты будет обратно пропорциональна времени выполнения самого медленного этапа. Если это время удастся сделать достаточно малым (а чем больше этапов на конвейере, тем они короче), то мы сумеем резко повысить производительность процессора.

Для иллюстрации принципа работы конвейера рассмотрим простейший пример.

Разобьем команду компьютера на 5 этапов:

- ВК – выборка команды из памяти;
- ДК –декодирование команды;
- ВО – выборка операнда;
- РО – реализация операции;
- ЗР – запоминание результата.

Из табл. 12.1 видно, что конвейер начинает работать в полную силу по мере его заполнения. Теоретически, выполняя последовательные команды, он ускоряет работу в 5 раз. На самом деле все не так хорошо, но тем не менее ускорение имеет место.

Таблица 12.1. Последовательность выполнения команд при наличии конвейера

Номер команды	**Номер такта**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n	ВК	ДК	ВО	РО	ЗР				
n+1		ВК	ДК	ВО	РО	ЗР			
n+2			ВК	ДК	ВО	РО	ЗР		
n+3				ВК	ДК	ВО	РО	ЗР	
n+4					ВК	ДК	ВО	РО	ЗР

Конвейеризация потенциально применима к любой процессорной архитектуре, независимо от набора команд и положенных в ее основу принципов. Уже в процессоре Intel 8086 был реализован примитивный «двухстадийный конвейер» – выборка новых инструкций и их исполнение осуществлялись в нем независимо друг от друга.

Недостатки конвейера неочевидны, но, как обычно и бывает, из-за нескольких «мелочей» грамотно организовать конвейер совсем не просто. Основных проблем три.

1. Необходимость наличия блокировок конвейера. Дело в том, что время исполнения большинства инструкций может очень сильно варьироваться. Скажем, умножение (и тем более деление) чисел требуют на стадии выполнения нескольких тактов, а сложение или побитовые операции – одного такта; а для других операций это время может быть различным. Соответственно, должен быть какой-то механизм, который бы «притормаживал» выборку и декодирование новых инструкций до тех пор, пока не будут завершены старые. Методов решения этой проблемы много, но их развитие приводит к одному – в процессорах прямо перед исполнительными устройствами появляются специальные блоки-диспетчеры, которые накапливают подготовленные к исполнению инструкции, отслеживают выполнение ранее запущенных инструкций и по мере освобождения исполнительных устройств отправляют на них новые инструкции. Даже если исполнение займет много тактов – внутренняя очередь диспетчера позволит в большинстве случаев не останавливать подготавливающий все новые и новые инструкции конвейер.

2. Необходимость наличия системы сброса процессора. Поскольку операции выборки и выполнения команды всегда выделены в отдельные стадии конвейера, то в тех случаях, когда в программном коде происходит разветвление (условный переход), зачастую оказывается, что по какой из веток пойти – пока неизвестно: инструкция, вычисляющая код условия, еще не выполнена. В результате процессор вынужден либо приостанавливать выборку новых инструкций до тех пор, пока не будет вычислен код условия (а это может занять очень много времени и в типичном цикле сильно затормозит процессор), либо, руководствуясь соображениями блока предсказания переходов, «угадывать», какой из переходов скорее всего окажется правильным.

3. Наконец, конвейер обычно требует наличия специального планировщика, призванного решать конфликты по данным. Если в программе идет зависимая цепочка инструкций (когда инструкция 2, следующая за инструкцией 1, использует для своих вычислений данные, только что вычисленные инструкцией 1), а время исполнения одной инструкции (от момента запуска на стадию выполнения и до записи полученных результатов в регистры) превосходит один такт, то мы вынуждены придержать выполнение очередной инструкции до тех пор, пока не будет полностью выполнена предыдущая команда.

Таким образом, идея конвейера в процессоре очень красива на словах и в теории, однако реализовать ее даже в простом варианте чрезвычайно трудно. Но выгода от конвейеризации столь велика и несомненна, что приходится с этими трудностями мириться, и ничего лучшего до сих пор не придумано.

В процессорах марки Pentium впервые в семействе процессоров Intel конвейер получил качественное развитие. По сравнению с более ранними процессорами фирмы Intel он усложнен тем, что содержит два

параллельных канала. Они могут функционировать одновременно. При благоприятных обстоятельствах такая структура увеличивает производительность вдвое. В Pentium Pro количество каналов еще больше увеличено. Структура конвейера в процессоре Pentium приведена на рис. 12.2. Эта архитектура получила название суперскалярной. При исполнении программы процессор проверяет две очередные инструкции программы на совместимость, и если они совместимы, то запускаются обе ветви конвейера. В противном случае первая инструкция запускается в U-конвейер, а V-конвейер простаивает.

Для уменьшения потерь времени от пустого конвейера в процессоре Pentium существует дополнительно блок предсказания ветвлений. Опираясь на вероятность переходов и статистику работы команд в предшествующих случаях, он может прогнозировать адрес перехода, с которого необходимо выбрать очередную команду. Более чем в половине случаев он угадывает.

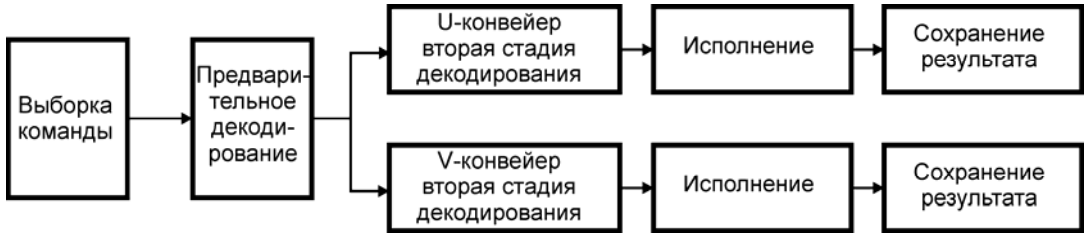


Рис. 12.2. Конвейер в процессоре Pentium

Таким образом, совершенствование компьютеров связано не только с увеличением тактовой частоты, объема постоянной и оперативной памяти, количества элементов, но и с алгоритмами их работы. Появление новых технологических возможностей и разработка новых алгоритмов позволяет оптимизировать работу процессора и в конечном счете добиться повышения его производительности.

Организация прерываний

По мере совершенствования технологии изготовления комплектующих для ЭВМ происходит совершенствование алгоритмов обработки. Значительное увеличение скорости работы центрального процессора привело к тому, что при обслуживании медленных устройств процессор непродуктивно расходовал ценное время. Для устранения таких простоев, а также для обработки программных ошибок была разработана система прерываний. Система прерываний позволяет с помощью определенного сигнала прервать выполнение текущей программы и начать выполнение программы с более высоким приоритетом. После окончания программы с более высоким приоритетом процессор возвращается к выполнению прерванной программы.

Таким образом, прерывание можно охарактеризовать как автоматическое изменение в программе расчета, вызванное определенным событием в какой-либо части вычислительной системы или необычными условиями в программе. Такими сигналами или условиями могут быть:

- сигналы от устройств ввода-вывода о готовности или об окончании операций ввода-вывода;
- сигнал, оповещающий об ошибке или ненормальности условий работы. Например, нарушение четности при чтении или записи информации, ошибки ввода-вывода;
- сигналы особых или аварийных условий. К ним относятся сигналы, сообщающие об аварии (или достижении какого-либо особого условия) внешнего устройства;
- необычное условие при выполнении программы – неприемлемая команда, деление на ноль и другие программные ошибки.

Механизм прерывания обеспечивается соответствующими аппаратно-программными средствами компьютера.

Любая особая ситуация, вызывающая прерывание, сопровождается сигналом, называемым запросом прерывания (ЗП). Запросы прерываний от внешних устройств поступают в процессор по специальным линиям, а запросы, возникающие в процессе выполнения программы, поступают непосредственно изнутри микропроцессора. Механизмы обработки прерываний обоих типов схожи. Рассмотрим функционирование компьютера при появлении сигнала запроса прерывания, опираясь в основном на обработку аппаратных прерываний (рис. 12.3). После появления сигнала *запроса прерывания* ЭВМ переходит к выполнению программы – обработчика *прерывания*. Обработчик выполняет те действия, которые необходимы в связи с возникшей особой ситуацией. Например, такой ситуацией может быть нажатие клавиши на клавиатуре компьютера. Тогда обработчик должен передать код нажатой клавиши из контроллера клавиатуры в процессор и, возможно, проанализировать этот код. По окончании работы обработчика управление передается прерванной программе.

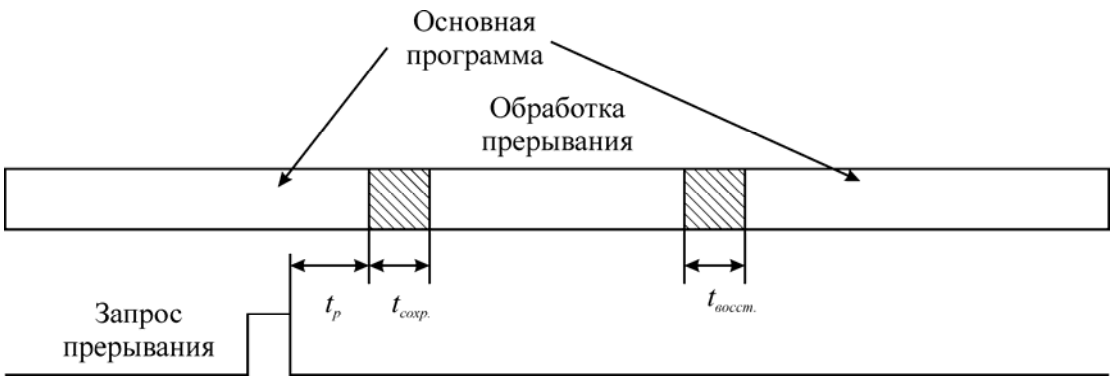


Рис. 12.3. Выполнение прерывания в компьютере

Время реакции t_p – это время между появлением сигнала *запроса прерывания* и началом выполнения прерывающей программы (обработчика *прерывания*) в том случае, если данное прерывание разрешено к обслуживанию.

Время реакции зависит от момента, когда процессор определяет факт наличия *запроса прерывания.* Опрос *запросов прерываний* может проводиться либо по окончании выполнения очередного этапа команды (например, считывание команды, считывание первого операнда и т. д.), либо после завершения каждой команды программы. Первый подход обеспечивает более быструю реакцию, но при этом необходимо при переходе к обработчику прерывания сохранять большой объем информации о прерываемой программе, включающей состояние буферных регистров процессора, номера завершившегося этапа и т. д. При возврате из обработчика также необходимо выполнить большой объем работы по восстановлению состояния процессора. Во втором случае время реакции может быть достаточно большим. Однако при переходе к обработчику прерывания требуется запоминание минимального контекста прерываемой программы (обычно это счетчик команд и регистр флагов). В настоящее время в компьютерах чаще используется распознавание запроса прерывания после завершения очередной команды. На рис. 12.3 время $t_{сохр.}$ – это время сохранения состояния прерываемой программы и вызова обработчика прерывания; $t_{восст.}$ – время *восстановления* прерванной программы.

Программа обработки прерывания в свою очередь также может быть прервана, если приоритет запроса выше, чем приоритет программы, выполняемой в данный момент. Таким образом, можно ввести понятие глубины прерывания. Глубина прерывания – максимальное число программ, которые могут прерывать друг друга. Глубина прерывания обычно совпадает с числом уровней приоритетов, распознаваемых системой прерываний.

Одна из возможных организаций системы прерываний в компьютере представлена на рис. 12.4. Если на входе «запрос 3» присутствует сигнал высокого уровня (есть запрос), то на выходе двухвходового логического элемента И-НЕ формируется логический 0, который блокирует прохождение сигнала опроса далее по цепочке и подается на вход 4-входового логического элемента И-НЕ. На его выходе формируется общий сигнал наличия запроса прерывания. Если «запрос 3» = 0, то анализируется сигнал «запрос 2», и т. д. На выходе приоритетного шифратора формируется номер поступившего запроса прерывания. Высшим приоритетом обладает запрос с большим номером.

Задача программиста состоит в том, чтобы составить программу для обработки прерывания, которая выполняла бы действия, связанные с появлением запроса данного типа, и поместить адрес начала этой программы в специальной таблице адресов *прерываний*. Программа-

обработчик, как правило, должна начинаться с сохранения состояния тех регистров процессора, которые будут ею изменяться, и заканчиваться восстановлением состояния этих регистров.

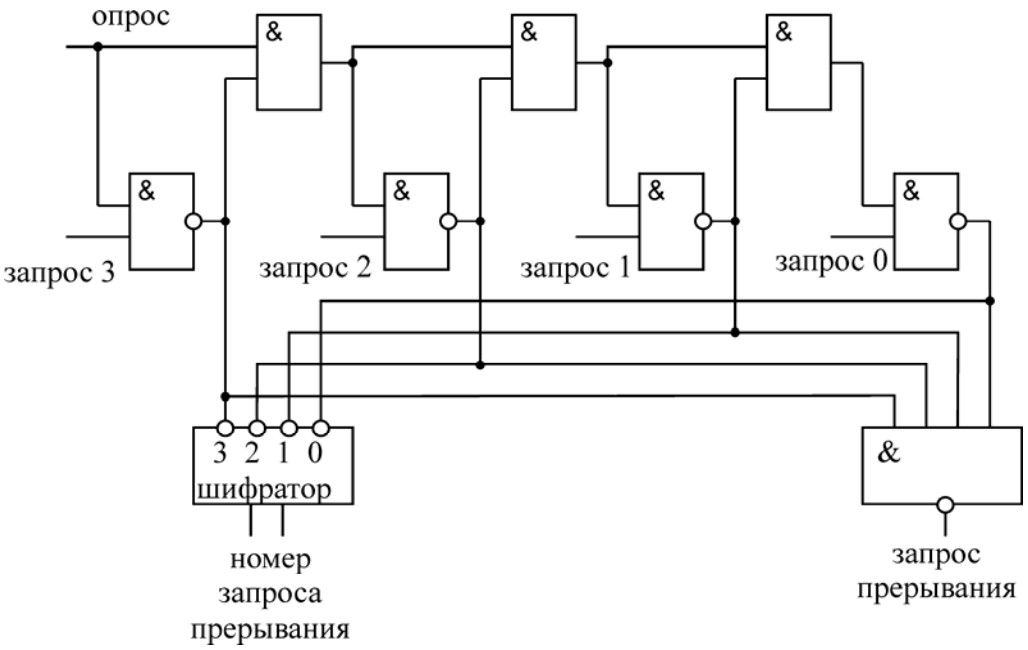


Рис. 12.4. Схема определения приоритетного номера запроса прерывания

Новые технологии, предназначенные для повышения производительности работы вычислительных систем, постоянно разрабатываются изготовителями микропроцессоров. Рассмотрим некоторые из них.

Hyper Threading Technology

Hyper Threading Technology (в пер. с англ. – технология гиперпоточности) – это технология повышения эффективности использования вычислительных блоков процессора. Для его обозначения используется аббревиатура HTT. Дело в том, что в обычном процессоре используется одновременно не более 40 % исполнительных блоков. Для того, чтобы создать более ощутимую нагрузку на процессор, компания Intel создала кардинальное решение: один физический процессор представлялся системе как два логических. Первый использовал основные ресурсы процессора, а второй – оставшиеся незадействованными ресурсы. Процессор Pentium 4 с включенным HTT система распознает как два процессора вместо одного.

Для эффективного применения данной технологии необходима разработка специального программного обеспечения, учитывающего архитектуру такого процессора. Подобные программы так и не появились, и поэтому такая технология использовалась в некоторых процессорах Pentium 4. Далее были разработаны двухъядерные процессоры,

и поначалу в них тоже использовалась технология HTT, но сейчас от нее отказались. Кроме проблем с поддержкой программ, была еще одна – второй процессор хоть и использовал «остатки» основного, все же создавал ему помехи в работе, что очень часто приводило не к увеличению, а даже к некоторому снижению производительности по сравнению с отключенным режимом HT.

MMX

MMX (Multimedia Extensions, мультимедийное расширение) – коммерческое название дополнительного набора инструкций, выполняющих характерные для процессов кодирования-декодирования потоковых аудио-видеоданных действия за одну машинную инструкцию. Впервые появился в процессорах Pentium MMX. Разработан в лаборатории Intel в первой половине 1990-х.

SSE

SSE (английское соокращение «Streaming SIMD Extensions», потоковое SIMD-расширение процессора). SIMD – это принцип компьютерных вычислений, характерный для персонального компьютера (англ. Single Instruction, Multiple Data, Одна инструкция – множество данных). Такой принцип позволяет при обработке информации обеспечить параллелизм на уровне данных. SSE – это набор инструкций, разработанный Intel и впервые представленный в процессорах серии Pentium III как ответ на аналогичный набор инструкций 3DNow! от AMD, который был представлен годом раньше. Первоначально названием этих инструкций было KNI, что расшифровывалось как Katmai New Instructions (Katmai – название первой версии ядра процессора Pentium III).

Технология SSE позволяла преодолеть две основных проблемы MMX – при использовании MMX невозможно было одновременно использовать инструкции сопроцессора, так как его регистры использовались для MMX и работы с вещественными числами.

SSE включает в архитектуру процессора восемь 128-битных регистров (от xmm0 до xmm7), каждый из которых трактуется как 4 последовательных значения с плавающей точкой одинарной точности. Преимущество в производительности достигается в том случае, когда необходимо произвести одну и ту же последовательность действий над разными данными. Реализация блоков SIMD осуществляется распараллеливанием вычислительного процесса между данными. В этом случае через один блок проходит поочередно множество потоков данных.

SSE2 (англ. Streaming SIMD Extensions 2, потоковое SIMD-расширение процессора) – это SIMD набор инструкций, разработанный Intel, и впервые представленный в процессорах серии Pentium 4.

SSE2 использует восемь 128-битных регистров (xmm0 – xmm7), включенных в архитектуру x86 с вводом расширения SSE, каждый из

которых трактуется как два последовательных значения с плавающей точкой двойной точности. SSE2 включает в себя набор инструкций, который производит операции со скалярными и упакованными типами данных. Также SSE2 содержит инструкции для потоковой обработки целочисленных данных в тех же 128-битных xmm-регистрах, что делает это расширение более предпочтительным для целочисленных вычислений, нежели MMX, появившийся гораздо ранее.

Преимущество в производительности достигается в том случае, когда необходимо произвести одну и ту же последовательность действий над большим набором однотипных данных.

SSE3 (PNI – Prescott New Instruction) – третья версия SIMD-расширения Intel, потомок SSE, SSE2 и x87. Впервые представлен 2 февраля 2004 года в ядре Prescott процессора Pentium 4. В 2005-м AMD предложила свою реализацию SSE3 для процессоров Athlon 64 (ядра Venice и San Diego).

SSE4 – это новый набор команд Intel Core микроархитектуры, впервые реализованный в процессорах серии Penryn. Он был анонсирован 27 сентября 2006 года, однако детальное описание стало доступно только весной 2007-го, свежее описание для программистов можно найти здесь на сайте Intel. SSE4 состоит из 54 инструкций, 47 из них относят к SSE4.1. Ожидается, что полный набор команд (SSE4.1 и SSE4.2, т. е. 47 + оставшиеся 7 команд) будет доступен в процессорах Nehalem. Ни одна из SSE4 инструкций не работает с 64-битными mmx-регистрами (только с 128-битными xmm0-15).

Фирма Intel уже занимается поддержкой этого нового набора команд. Компилятор языка Си от Intel начиная с версии 10 будет генерировать инструкции SSE4 при задании опции -QxS.

Supplemental Streaming SIMD Extension 3 (SSSE3) – это обозначение, данное Intel 4-му расширению системы команд. Предыдущее имело обозначение SSE3 и Intel добавил еще один символ «S» вместо того, чтобы увеличить номер расширения, возможно потому, что они посчитали SSSE3 простым дополнением к SSE3. Часто, до того как стало использоваться официальное обозначение SSSE3, эти новые команды назывались SSE4. Также их называли кодовыми именами Tejas New Instructions (TNI) и Merom New Instructions (MNI) по названию процессоров, где впервые Intel намеревалась поддержать эти новые команды. Появившись в Intel Core Microarchitecture, SSSE3 доступно в сериях процессоров Xeon 5100 (Server и Workstation версии), а также в процессорах Intel Core 2 (Notebook и Desktop версии).

Новыми в SSSE3, по сравнению с SSE3, являются 16 уникальных команд, работающих с упакованными целыми. Каждая из них может работать как с 64-битными (MMX), так и с 128-битными (XMM) регистрами, поэтому Intel в своих материалах ссылается на 32 новые команды.

Другие технологии

Новые технологии, предназначенные ускорить работу компьютеров и повысить надежность работы, непрерывно совершенствуются.

Например, уделяется внимание разработке функций защиты системы от вирусов, троянских коней и прочих вредоносных программ (технология NT), поддержка нескольких операционных систем на одном компьютере (Intel Virtualization Technology), ряд энергосберегающих технологий (Intel SpeedStep, Intel VRT, AMD PowerNow!, Transmeta Long Run, Transmeta LongRun2, VIA LongHaul).

В процессоре Core 2 Duo, например, применен целый ряд новейших технологий, позволяющих еще больше повысить эффективность процессоров этого типа. К ним относятся:

- Intel Wide Dynamic Execution – технология выполнения большего количества команд за каждый такт, повышающая эффективность выполнения приложений и сокращающая энергопотребление. Каждое ядро процессора может выполнять до четырех инструкций одновременно с помощью 14-стадийного конвейера;
- Intel Intelligent Power Capability – технология, с помощью которой для исполнения задач активируется работа отдельных узлов чипа по мере необходимости, что значительно снижает энергопотребление системы в целом;
- Intel Advanced Smart Cache – технология использования общей для всех ядер кэш-памяти L2, что снижает общее энергопотребление и повышает производительность, при этом, по мере необходимости, одно из ядер процессора может использовать весь объем кэш-памяти при динамическом отключении другого ядра;
- Intel Smart Memory Access – технология оптимизации работы подсистемы памяти, сокращающая время отклика и повышающая пропускную способность подсистемы памяти;
- Intel Advanced Digital Media Boost – технология обработки 128-разрядных команд SSE, SSE2 и SSE3, широко используемых в мультимедийных и графических приложениях, за один такт.

Принципы работы новых технологий можно посмотреть на http://ru.wikipedia.org/wiki/.

Контрольные вопросы

1. Назовите основные составные части современного однопроцессорного компьютера.
2. Какие основные отличия современного компьютера от первых ЭВМ?

Лекция 13. ИНТЕРФЕЙСЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Компьютеры используются в комбинации с различными периферийными устройствами. Типичными примерами таких устройств являются клавиатура, мышь, дисплей, принтер и другие устройства. К компьютеру эти устройства подсоединяются при помощи интерфейсов (специализированных устройств для обеспечения взаимодействия периферийного устройства и основного компьютера). Для того чтобы взаимодействие было возможно, необходимо обеспечить согласование электрических и логических параметров устройств. Для этого разрабатывают специальные стандарты.

При обмене информацией необходимо обеспечить согласование на логическом, электрическом и механическом (конструктивном) уровнях.

В качестве логических условий должны быть заданы виды сигналов (адресные, информационные и управляющие) и их количество, система кодирования и форма передачи данных, функции адресных и управляющих сигналов и т. п.

В качестве электрических условий обеспечения совместимости задаются значение напряжений (логических) двоичных сигналов, временные параметры этих сигналов, нагрузочная способность по входу и выходу сопрягаемых цепей и т. д.

К конструктивным условиям обеспечения совместимости относятся конструктив соединения (тип разъема), распределение контактов в разъемном соединении, допустимые типы кабеля и др.

В отечественной практике для описания совокупности схемотехнических средств, обеспечивающих непосредственное взаимодействие составных элементов систем обработки данных (ЭВМ, сетей ЭВМ, систем передачи данных), подсистем периферийного оборудования, используются понятия «интерфейс», «стык» и «протокол».

Под стандартным интерфейсом понимается совокупность унифицированных аппаратурных, программных и конструктивных средств, необходимых для реализации взаимодействия различных функциональных элементов. В техническом обиходе стандартный интерфейс – это система обмена данными между узлами системы или различными системами, описанная стандартом.

Стык – место соединения устройств передачи сигналов данных, входящих в систему передачи данных. Это понятие используется вместо понятия интерфейса для описания функций и средств сопряжения элементов средств связи и систем передачи данных (СПД).

Под протоколом понимается строго заданная процедура или совокупность правил, регламентирующая способ выполнения определенного класса функций. Взаимосвязь понятий интерфейса и протокола не всегда однозначна, так как практически любой интерфейс содержит в большей или меньшей степени элементы протокола, определяемые процедурами и функциональными характеристиками интерфейса.

Основное назначение интерфейсов, стыков и протоколов – унификация обмена информацией как внутри компьютера, так и для обмена с периферийными устройствами.

Различают несколько видов интерфейсов:

- системные (внутрисистемные), которые являются базовой частью архитектуры ЭВМ и представляют собой совокупность унифицированной магистрали, электронных схем, управляющих прохождением сигналов по шинам, и т. п.;
- периферийного оборудования, включающие универсальные (параллельный и последовательный) и специализированные интерфейсы (для жестких и гибких дисков, накопителей на магнитной ленте и т. п.);
- программируемых приборов, служащие для подключения нестандартной аппаратуры, измерительных и управляющих систем;
- магистрально-модульных, микропроцессорных систем;
- локальных вычислительных систем и т. п.

В современном персональном компьютере в качестве системных интерфейсов используются PCI (универсальный), PCI-E (для подключения видеоадаптеров), периферийные интерфейсы IDE, SCSI, SATA, USB, COM, IEEE1394, и множество других. Рассмотрим основные принципы их работы и некоторые примеры.

Организация ввода-вывода информации в ЭВМ

В передаче данных в самом общем случае участвуют следующие компоненты (см. рис. 13.1): приемник, передатчик и канал связи. Если речь идет о подключении внешнего устройства к компьютеру, то он организован с использованием контроллеров внешних устройств. Контроллеры подключаются к устройству сопряжения или непосредственно к шине компьютера и обеспечивают прием, передачу и преобразование сигналов, принятых в компьютере и во внешнем устройстве, и наоборот.

Варианты обмена данными между устройствами можно разделить по разрядности приемопередающих устройств на параллельные и последовательные и по способу обмена информацией на синхронные и асинхронные. Параллельный тип обмена характеризуется тем, что слово в линию связи посылается целиком, а последовательный – когда биты слова пересылаются по очереди друг за другом по одной линии. Синхронность означает, что каждый передаваемый элемент информации сопровождается импульсом синхронизации, а при асинхронном

обмене такого импульса нет. В соответствии с этим можно выделить четыре способа обмена информацией: последовательный синхронный, последовательный асинхронный, параллельный синхронный и параллельный асинхронный.

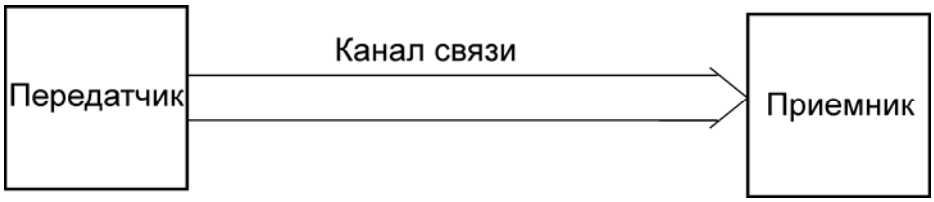


Рис. 13.1. Схема передачи данных

Синхронная последовательная передача информации требует линию связи минимум в три провода: для данных, для импульсов синхронизации и заземляющий провод. Такая передача начинается с пересылки в приемник одного или двух импульсов синхронизации. Получив такой символ, приемник начинает прием данных и их преобразование из последовательного формата в параллельный.

Асинхронная последовательная передача данных означает, что у передатчика и приемника нет общего генератора синхроимпульсов и синхронизирующий сигнал не передается вместе с данными. Поэтому при асинхронной передаче важно, чтобы приемник и передатчик были настроены на одну частоту. Аналогично можно охарактеризовать параллельный синхронный и асинхронный способы обмена информацией.

Организация последовательной передачи информации

Рассмотрим более подробно возможную реализацию принципа последовательной синхронной передачи данных на примере устройства для осуществления этого принципа – контроллера синхронной последовательной передачи данных (см. рис. 13.2).

Буферный регистр контроллера последовательной передачи данных служит для временного хранения байта данных до его загрузки в регистр сдвига. Запись байта данных производится при наличии единицы на выходе регистра состояния. Эта единица указывает на готовность контроллера принять очередной байт в буферный регистр. Выход регистра состояния учитывается логикой управления при формировании сигнала готовности контроллера, передаваемого в компьютер. При записи очередного байта данных в буферный регистр обнуляется регистр состояния (сигнал записи в буфер одновременно подается на сброс регистра состояния).

Последовательная линия связи контроллера с внешним устройством подключается к выходу младшего разряда регистра сдвига. По очередному тактовому импульсу генератора содержимое регистра сдвига сдвигается на один двоичный разряд вправо, и в линию связи с внешним

устройством выдается значение очередного разряда. Одновременно с битом данных в линию связи передается тактовый импульс «синхронизация». Таким образом, каждый бит данных сопровождается импульсом синхронизации, что обеспечивает его однозначное восприятие в приемнике.

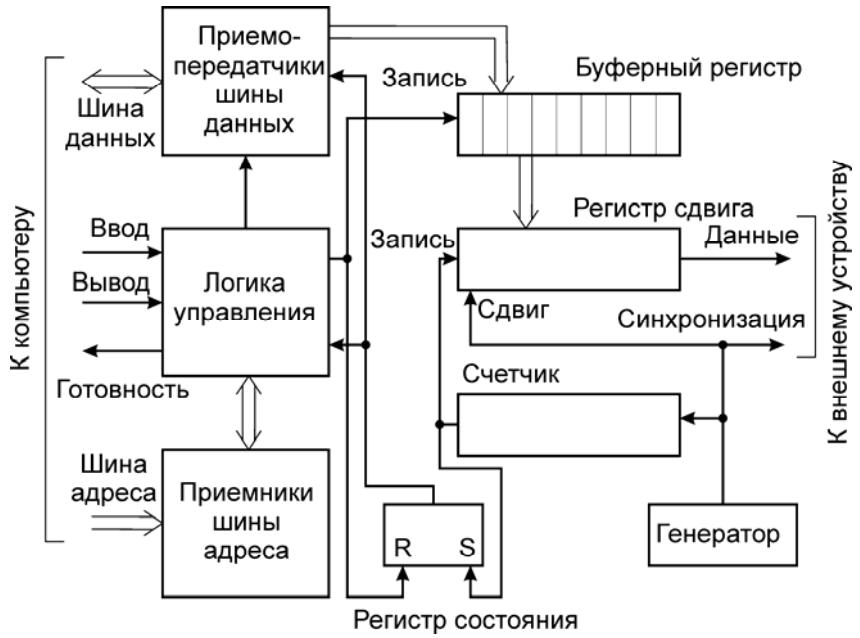


Рис. 13.2. Блок-схема контроллера последовательной синхронной передачи данных

Количество передаваемых битов информации подсчитывается счетчиком. Как только содержимое счетчика становится равным длине передаваемого слова, формируется управляющий сигнал загрузки регистра сдвига, и из буферного регистра в регистр сдвига загружается новая порция данных. Этим же управляющим сигналом устанавливается регистр состояния. Когда же он установлен, в компьютер отправляется сигнал о готовности контроллера принять очередную порцию данных, и цикл передачи снова повторяется.

Контроллер последовательного синхронного приема данных содержит те же элементы, за исключением, может быть, генератора. В этом случае тактовые импульсы поступают от передающего устройства.

Стандарт RS232C

Стандартный интерфейс RS-232C был первоначально разработан для сопряжения компьютера (терминала) с модемом (устройством передачи данных). Этот интерфейс использовался также для подключения мыши, в некоторых случаях – текстового дисплея, для соединения компьютеров между собой. В настоящее время используется все реже, поскольку ему на смену приходит USB. Этому стандарту соответствует 25-штырьковый двухрядный разъем (или 9-штырьковый в сокращенном варианте).

Уровень напряжения от +5 до +15 В считается логическим «0», а от –5 до –15 В – логической «1». Кроме того, могут быть существенными и другие ограничения, поэтому электрические характеристики подробно описаны в стандарте.

Полный интерфейс содержит 13 соединительных проводов, хотя обычно используется 9 (см. рис. 13.3). Стрелками показано направление передачи информации по линии. Обсудим назначение этих соединительных проводов.

Рис. 13.3. Стандарт RS232C

Назначение выводов интерфейса следующее:

Вывод 1. Защитное заземление (PG) соединяет корпуса приборов для предотвращения накопления статического заряда и удаления сетевых наводок.

Вывод 2. Передаваемые данные (TxD) передаются из компьютера в информационный терминал (модем, принтер). Передача данных не осуществляется, пока все управляющие сигналы не будут установлены в активное состояние.

Вывод 3. Принимаемые данные (RxD) – данные, полученные компьютером из модема.

Вывод 4. Запрос на передачу (RTS) – возбуждается компьютером, когда он должен передать данные по линии. Этот вывод должен сохранять активное состояние до конца передачи.

Вывод 5. Разрешение передачи (CTS) – используется модемом для информирования о том, готов ли он к приему передаваемых данных. Этот вывод также должен быть активным на протяжении всей передачи. Если этот сигнал переходит в неактивное состояние во время сеанса передачи данных, компьютер прекращает передачу и формирует сообщение об ошибке.

Вывод 6. Готовность модема (DSR) сигнализирует о том, что источник питания модема включен и устройство находится в рабочем состоянии.

Вывод 7. Сигнальное заземление (SG) – общий провод для информационных сигналов.

Вывод 8. Обнаружение несущей (DCD, CD) используется модемом для информирования передатчика о том, что каналом передачи можно пользоваться, и обычно активизируется в случае, когда уже выдан сигнал «Запрос передатчика».

Вывод 20. Готовность (DTR) – сигнализирует о том, что устройство включено и с ним возможна связь.

В компьютерных системах до сих пор используется COM-порт, соответствующий стандарту RS-232. Его уже редко используют для подключения мыши, но некоторые внешние модемы, устройства программирования микроконтроллеров, ключи аппаратной защиты подключаются именно к нему. И хотя для всех компьютеров новый интерфейс USB уже стал стандартным, COM-порт все еще активно применяется.

COM-порт обеспечивает асинхронную последовательную передачу данных. На рис. 13.4 представлена временная диаграмма передачи байта данных. Когда линия свободна, на ней присутствует уровень логической «1». Перед байтом данных передается «стартовый бит» – передатчик устанавливает в линии уровень логического «0». Когда приемник его обнаруживает, начинается прием байта данных. После байта данных передается бит четности, затем один или несколько стоповых битов. Для того чтобы передача велась корректно, приемник и передатчик должны соответствовать друг другу по параметрам. Основные параметры – это частота передачи и количество стоповых битов. После передачи одного байта может следовать пауза неопределенной длительности или передача нового байта.

Рис. 13.4. Временная диаграмма передачи байта в COM-интерфейсе

Структура интерфейса принтера (стандарт Centronics)

Печать символов на принтере осуществляется путем параллельного вывода 8-разрядных данных. Так как буквы латинского алфавита и цифры представлены кодом ASCII, то печать каждого символа производится

путем вывода байта данных. В текстах на японском языке каждый символ представлен 2 байтами, но и в этом случае осуществляется побайтовый вывод данных. Центральным узлом интерфейса является 8-разрядный порт вывода, хотя кроме него для работы принтера требуется еще несколько управляющих сигналов.

Однако в основном средства интерфейсов совпадают, причем часто используется интерфейс, предложенный фирмой Centronics (США). Этот интерфейс предусматривает использование уровней ТТЛ-сигналов и в качестве соединителя имеет со стороны принтера 36-выводной двухрядный разъем, а со стороны компьютера – 25-штырьковый. Этот интерфейс пока еще используется, но все чаще для связи спринтером его заменяют на USB.

Поскольку это параллельный интерфейс, данные в нем передаются побайтно. Отдельные биты обозначены *D*7 – *D*0. Кроме того, существуют служебные сигналы. Самые важные из них:

1. BUSY (занято).
2. $\overline{\text{DATASTROBE}}$ (строб данных).
3. $\overline{\text{ACKNOWLEDGE}}$ (подтверждение).

Они используются для обмена служебными сигналами при установлении связи.

Управление работой принтера со стороны компьютера осуществляется следующим образом (см. рис. 13.5):

1. Подтверждается, что сигнал BUSY имеет низкий уровень.
2. Осуществляется вывод данных из 8-разрядного порта вывода.
3. Передается сигнал $\overline{\text{DATASTROBE}}$, управляющий началом печати.

Принтер выполняет следующие операции:

1. При наличии стробирующего сигнала уровень сигнала BUSY становится высоким, начинается печать.
2. Когда печать завершена, передается сигнал $\overline{\text{ACKNOWLEDGE}}$ и уровень сигнала BUSY становится низким.

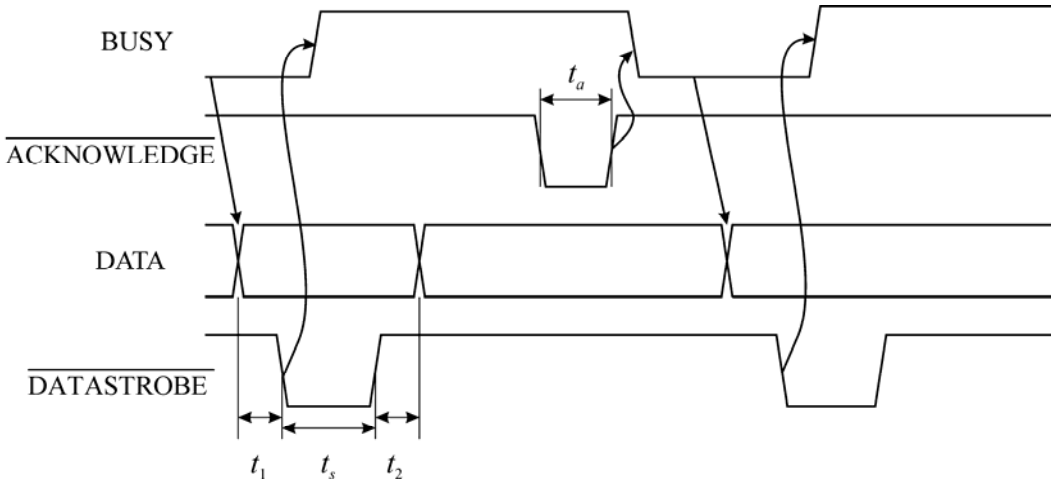


Рис. 13.5. Обмен сигналами между принтером и компьютером

Временная диаграмма работы принтера (см. рис. 13.5) должна устанавливаться в соответствии с техническими требованиями на принтер. Так, устанавливаются минимальные значения интервалов:

t_1 – время от установки данных до передачи стробирующего сигнала;

t_s – длительность строб-импульса;

t_2 – время удержания данных после окончания передачи стробирующего сигнала.

Обычно t_s, t_1 и t_2 составляют 500 нс – 1 мкс, t_a равно 5–10 мкс. Кроме того, в принтере существуют некоторые служебные сигналы:

- PE – отсутствие бумаги;
- $\overline{\text{FAULT}}$ – состояние, в котором печать невозможна, включая также отсутствие бумаги;
- SELECT – принтер находится в рабочем состоянии;
- $\overline{\text{INTI}}$ – сигнал сброса контроллера;
- $\overline{\text{AUTOFEED}}$ – сигнал, который управляет автоматическим переводом строки при появлении кода возврата каретки.

Стандарт USB

USB (Universal Serial Bus – универсальная последовательная шина) появилась не так давно (версия первого утвержденного варианта стандарта датирована 15 января 1996 года). В разработке стандарта приняли участие такие лидеры компьютерной и телекоммуникационной промышленности, как Microsoft, Intel, DEC, IBM, NEC, Northen Telecom и Compaq.

Основная цель стандарта, поставленная перед его разработчиками, – обеспечить пользователям реальную возможность работы в режиме Plug&Play с периферийными устройствами. Это означает, что должно быть предусмотрено подключение устройства к работающему компьютеру, автоматическое распознавание его немедленно после подключения и последующей установки соответствующих драйверов (если это необходимо). Кроме того, желательно обеспечить подачу питания для маломощных устройств с самой шины. Скорость шины должна быть достаточной для подавляющего большинства низкоскоростных периферийных устройств. При этом контроллер USB занимает только одно прерывание независимо от количества подключенных к шине устройств.

Практически все поставленные задачи были решены в стандарте USB 1.1, и уже весной 1997 года появились компьютеры, оборудованные разъемами для подключения USB-устройств. Однако периферия с подключением по USB стала реальностью лишь спустя год, в середине 1998-го. По мере развития стандарт USB получил широкую популярность, и количество устройств, использующих USB, стало быстро расти.

В настоящее время USB активно используется многими производителями компьютерной периферии. Рассмотрим некоторые характеристики USB 1.1:

- Высокая максимальная скорость обмена – до 12 Мбит/с.
- Максимальная длина кабеля – 4,5–5 м.
- Максимальное количество подключенных устройств (включая концентраторы) – до 127.
- Возможно подключение устройств с различными скоростями обмена.
- Не требуется использование дополнительных устройств и терминаторов.
- По шине может подаваться напряжение питания 5 В для периферийных устройств.
- Максимальный ток потребления на одно устройство – 500 мА.

При разработке USB 2.0 группе требовалось решить две основные задачи: во-первых, необходимо было сохранить совместимость со старым стандартом, во-вторых, следовало добиться скорости, в 10–20 раз превышающей скорость, которую обеспечивает существующий стандарт (USB 1.1). Первую задачу группа выполнила: современные ПК, периферийные устройства и кабели, поддерживающие USB, смогут бесконфликтно работать с новым стандартом. Что касается второй цели, то тут результат превзошел самые смелые ожидания: скорость возросла в 40 раз!

С появлением стандарта USB 2.0 далеко не все проблемы, стоящие перед разработчиками различных электронных устройств, нашли свое решение. Большим недостатком USB являлось то, что устройства, подключенные к одной шине, не могли работать друг с другом напрямую. Связь была возможна только при участии HOST-контроллера, который, как правило, установлен на материнской плате персонального компьютера. Таким образом, в некоторых случаях компьютер оказывается «третьим лишним». Например, у вас есть карманный компьютер и принтер, подключенные к вашему настольному компьютеру при помощи USB, и вы хотите распечатать файл, находящийся в памяти карманного компьютера. Очевидно, что для этого необходимо скопировать файл на настольный компьютер, а уже затем его распечатать. Оказалось, что это можно сделать напрямую, то есть подключить карманный компьютер непосредственно к принтеру при помощи USB-соединителя и передавать данные для печати. В декабре 2001 года отраслевой консорциум Universal Serial Bus Implementers Forum, созданный с целью дальнейшего развития стандарта USB, выпустил дополнение к этому стандарту: USB On-The-Go (сокращенно OTG). Это дополнение позволило соединять устройства не через ПК, а непосредственно друг с другом. Например, музыкальные файлы можно переписать с MP3-плеера прямо на другой плеер, а цифровую камеру подключить прямо к принтеру.

Подключение устройств USB1.1

Для подключения USB-устройств к хосту (компьютеру) или концентратору (хабу) USB используется стандартный разъем с четырьмя контактами (два контакта – дифференциальные данные, другие два – питание +5В). Внешний вид одной из разновидностей USB-разъемов (тип А) представлен на рис. 13.6. Кроме представленных на рисунке, существует еще несколько видов разъемов для подключения периферийных устройств (фотоаппаратов, принтеров, MP3-плееров и др.)

Рис. 13.6. USB-разъемы на флэш-карте и USB-удлинителе

Низкоскоростное (Low Speed, LS) соединение USB происходит по кабелю из неэкранированной невитой пары с максимальной длиной в 3 метра. Длительность фронта и спада сигналов на этом кабеле должна быть в пределах от 75 нс до 300 нс. С одной стороны, это ограничивает излучение, а с другой – возникают задержки синхронизации и перекосы передачи сигналов и искажения. Драйвер (усилитель) должен достигать определенных статических уровней сигнала с гладким нарастанием и спадом во времени, с минимальными отражениями сигнала при передаче по кабелю. Неэкранированный кабель используется только в сегментах сети между низкоскоростными устройствами и портами, с которыми они соединены. Подключение показано на рис. 13.7.

Рис. 13.7. Кабель низкоскоростного (LS) устройства
и подсоединение резисторов

Полноскоростное (Full Speed, FS) соединение USB происходит по кабелю из экранированной витой пары с характеристикой импеданса (Z_0) $90\Omega \pm 15$ % и максимальной длиной в 5 метров. Импеданс каждого из драйверов должен быть между 29 и 44 Ω. Длительности фронта и спада на линии данных должны находиться в диапазоне от 4 до 20 нс, а сами изменения должны быть монотонными, гладкими и хорошо согласованными. Это позволяет минимизировать излучение и перекосы в сигнале. Подключение показано на рис. 13.8.

USB-устройство при подключении к компьютеру указывает свою скорость, поскольку в нем имеется резистор R2, подтягивающий одну из линий данных к напряжению питания. Этот резистор указывает компьютеру, что к компьютеру подключено новое устройство, а в зависимости от того, к какой линии данных подключен этот резистор, компьютер понимает, какую скорость он может обеспечить.

Рис. 13.8. Кабель полноскоростного (FS) устройства и подсоединение резисторов

Передача сигналов данных при обмене информацией по USB1.1

Для обмена информацией используется пакетная передача данных, то есть передатчик формирует пакет (набор) данных, сообщает о начале передачи пакета приемнику, а в конце формирует сигнал окончания пакета. Данные внутри пакета передаются дифференциальными сигналами.

Приемник видит дифференциальную 1, если на шине D+ по крайней мере на 200 mV больше, чем на D– , и видит дифференциальный 0, если на D–, по крайней мере на 200 mV больше, чем на D+. Точка пересечение сигнала должна быть между 1.3 и 2.0 В.

Начало пакета (Start Of Packet, SOP) сообщает о появлении порта, переводя линии D+ и D– от неактивного состояния к активному. Далее идут дифференциальные данные. Состояние асимметричного 0 используется, чтобы сообщить о конце пакета (End Of Packet, EOP). Это состояние фиксируется при нахождении сразу двух сигналов D+ и D– ниже 0.8 В в течение двух времен передачи бита. Далее линия перево-

дится в остановленное состояние в течение одного времени передачи бита. Остановленное состояние удерживается в течение одного времени передачи бита, затем и D+ и D– выходные драйверы переводятся в состояние высокого импеданса. Согласующие резисторы шины удерживают шину в неактивном состоянии. Рис. 13.9 показывает передачу сигналов для начала и конца пакета.

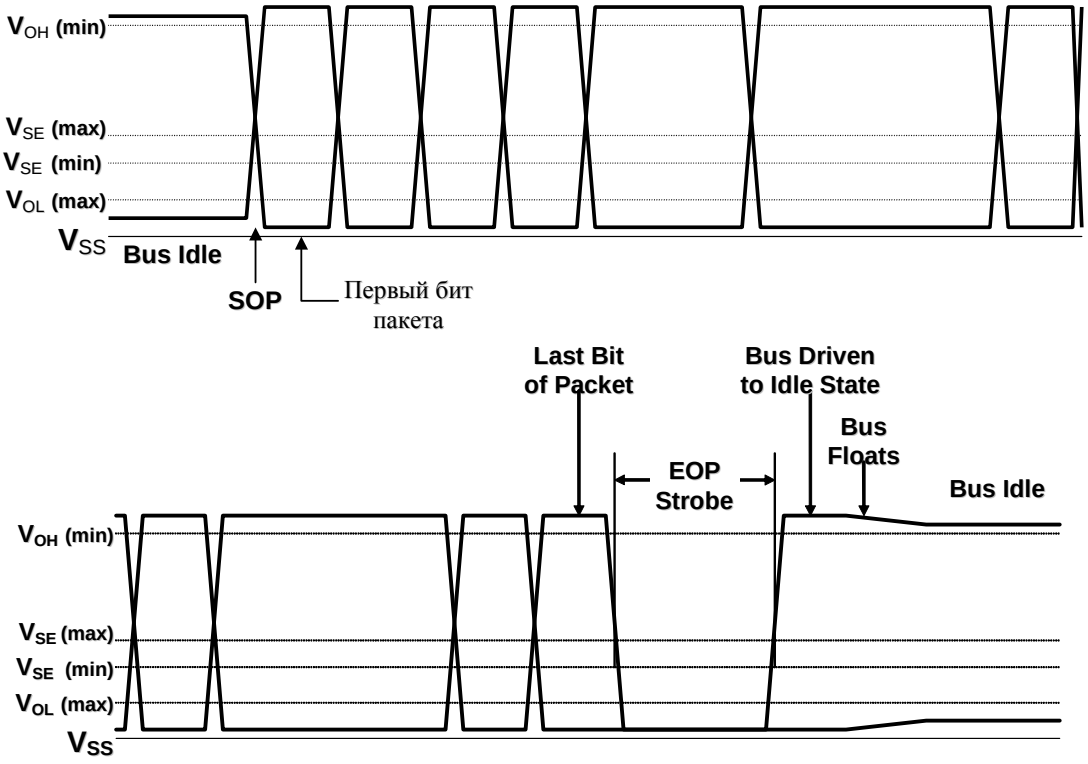


Рис. 13.9. Передача сигналов данных в пакете по шине USB

По последовательной линии USB передаются последовательности битов, кодированные при помощи специального метода NRZI with bit stuffing (Non Return to Zero Invert, метод возврата к нулю с инвертированием единиц).

При этом кодируются не логические уровни, а их изменения. Логический 0 определяется как изменение напряжения, а логическая 1 – как неизменение напряжения. Таким образом, потенциал, используемый для кодирования текущего бита, зависит от потенциала, который использовался для кодирования предыдущего бита. Если текущий бит имеет значение 0, то текущий потенциал представляет собой инверсию потенциала предыдущего бита, вне зависимости от того, каково было его значение. Если же текущий бит имеет значение 1, то текущий потенциал повторяет предыдущий (см. рис. 13.10).

Такой алгоритм кодирования повышает надежность передачи данных. В то же время видно, что если в байте подряд идет большое количество нулей, то приемнику и передатчику достаточно легко поддерживать синхронизацию, поскольку уровень сигнала постоянно изменяется.

С другой стороны, если в байте большое количество записанных подряд единиц, то уровень сигнала не будет изменяться и возможна рассинхронизация. В этом случае для повышения надежности передачи данных используется специальный прием, называемый стаффингом (bit stuffing). Это действие заключается в том, что после каждых 6 единиц, переданных подряд, автоматически добавляется ноль.

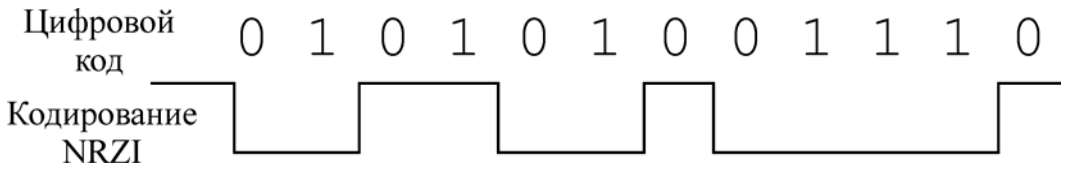


Рис. 13.10. Кодирование данных методом NRZI

Использование USB в настоящее время представляется весьма перспективным. Скорость обмена данными возросла настолько, что в его рамках можно передавать потоковое видео. Единственным недостатком этого интерфейса является короткий соединительный кабель, но поскольку эта шина разрабатывалась для домашнего применения, то дальние подключения в нее не закладывались даже изначально.

Контрольные вопросы и задания

1. По каким признакам можно проклассифицировать различные интерфейсы?
2. Какие основные узлы используются в системе последовательной передачи данных?
3. Какие современные интерфейсы вам известны?
4. Каков принцип передачи информации с использованием интерфейса RS-232? Centronics? USB?

Список литературы к лекциям 11–13

1. *Королев Л.Н.* Архитектура электронных вычислительных машин. – М. : Научный мир, 2005. – 272 с. : ил.
2. *Лапин А*. Интерфейсы. Выбор и реализация. М. : Техносфера, 2005. 168 с. : ил.
3. *Агуров П.В*. Интерфейсы USB. Практика использования и программирования. – СПб. : БХВ-Петербург, 2004. – 576 с. : ил.

Лекция 14. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

Периферийными называются устройства, входящие в состав внешнего оборудования компьютера и предназначенные для ввода или вывода данных, а также организации хранения данных и передачи их к другим узлам – потребителям информации.

Компьютер обменивается информацией с внешним миром с помощью периферийных устройств. Только благодаря периферийным устройствам человек может взаимодействовать с компьютером, а также со всеми подключенными к нему устройствами. Любое подключенное периферийное устройство в каждый момент времени может быть или занято выполнением порученной ему работы или пребывать в ожидании нового задания. Влияние скорости работы периферийных устройств на эффективность работы с компьютером не меньше, чем скорость работы его центрального процессора. Скорость работы внешних устройств от быстродействия процессора не зависит.

Периферийные устройства можно разделить на следующие основные классы:

1. Устройства ввода и вывода информации (к ним относятся также интерактивные устройства, предназначенные одновременно для ввода и вывода). Устройства ввода-вывода необходимы для организации связи с пользователем компьютера.
2. Устройства для хранения информации (жесткие диски, флэш-память и др.).
3. Устройства управления другими объектами и получения связи от других объектов (различные датчики, АЦП и ЦАП). Их отличием от устройств ввода-вывода является то, что связь осуществляется не с человеком, а с другими объектами.
4. Средства приема-передачи данных, или телекоммуникационные системы. Они предназначены для передачи данных на значительные расстояния.

В двух следующих лекциях будут рассмотрены некоторые устройства ввода и вывода информации.

Устройства ввода

Устройства ввода преобразуют информацию в цифровую форму, понятную машине, после чего компьютер может ее обрабатывать и запоминать. Наиболее распространенные устройства ввода предназначены для ввода информации с участием человека (клавиатура, мышь и т. д.). Другой большой класс устройств ввода относится к элементам автоматики (преобразователи любых физических величин в цифровой код).

Самым известным устройством ввода информации является клавиатура (keyboard) – это стандартное устройство, предназначенное для ручного ввода информации. Работой клавиатуры управляет контроллер клавиатуры, расположенный на материнской плате и подключаемый к ней через разъем на задней панели компьютера. При нажатии пользователем клавиши на клавиатуре, контроллер клавиатуры преобразует код нажатой клавиши в соответствующую последовательность битов и передает их компьютеру. Отображение символов, набранных на клавиатуре, на экране компьютера называется *эхом*. Обычная современная клавиатура имеет, как правило, 101–104 клавиши, среди которых выделяют алфавитно-цифровые клавиши, необходимые для ввода текста, клавиши управления курсором и ряд специальных и управляющих клавиш. Существуют беспроводные модели клавиатуры, в них связь клавиатуры с компьютером осуществляется посредством инфракрасных лучей.

Наиболее важными характеристиками клавиатуры являются чувствительность ее клавиш к нажатию, мягкость хода клавиш и расстояние между клавишами. Долговечность клавиатуры определяется количеством нажатий, которые она рассчитана выдержать. Клавиатура проектируется таким образом, чтобы каждая клавиша выдерживала 30–50 миллионов нажатий.

К манипуляторам относят устройства, преобразующие движения руки пользователя в управляющую информацию для компьютера. Среди манипуляторов выделяют мыши, трекболы, тачпады, джойстики.

Мышь предназначена для перемещения курсора (специального указателя), а также для выбора и перемещения графических объектов на экране монитора компьютера. Мышь позволяет существенно сократить работу человека с клавиатурой при управлении курсором и вводе команд. Особенно эффективно мышь используется при работе с графическими редакторами, издательскими системами, играми. Современные операционные системы также активно используют мышь для управляющих команд. У мыши есть клавиши, две из которых относятся к основным. Между двумя клавишами часто располагают колесо прокрутки, которое позволяет осуществлять прокрутку документов вверх-вниз и другие дополнительные функции.

Мышь состоит из пластикового корпуса, сверху находятся кнопки, соединенные с микропереключателями. Внутри корпуса находится обрезиненный металлический шарик, нижняя часть которого соприкасается с поверхностью стола или специального коврика для мыши, который увеличивает сцепление шарика с поверхностью. При движении манипулятора шарик вращается и переедает движение на соединенные с ним датчики продольного и поперечного перемещения. Датчики преобразуют движения шарика в соответствующие импульсы, которые передаются по проводам мыши в системный блок на управляющий контроллер. Контроллер передает обработанные сигналы операционной

системе, которая перемещает графический указатель по экрану. В более современных оптических мышах красный светодиод освещает поверхность, на которой находится мышь, а функции датчика движения выполняют светодиоды, анализирующие перемещение мыши. В беспроводной мыши данные передаются с помощью инфракрасных лучей.

Трекбол по функциям близок мыши, но шарик в нем больших размеров, и перемещение указателя осуществляется вращением этого шарика руками. Трекбол удобен тем, что его не требуется перемещать по поверхности стола, которого может не быть в наличии. Поэтому, по сравнению с мышью, он занимает на столе меньше места. Переносные компьютеры оснащаются часто встроенным трекболом (в последнее время более распространенным устройством в ноутбуках является тачпад). Трекбол функционально представляет собой перевернутую мышь. Шар находится сверху или сбоку, и пользователь может вращать его ладонью или пальцами, при этом не перемещая корпус устройства. Несмотря на внешние различия, трекбол и мышь конструктивно похожи – при движении шар приводит во вращение пару валиков или, в более современном варианте, его сканируют оптические датчики перемещения (как в оптической мыши).

Тачпад (от англ. touchpad – сенсорная площадка) – специальная сенсорная панель, применяемая вместо мыши, чаще всего, в ноутбуках. Как и мышь, тачпад обычно используется для управления «мышиным курсором», перемещением пальца по поверхности устройства. Тачпады имеют различные размеры, но обычно их площадь не превосходит 50 $см^2$.

Работа тачпадов основана на измерении емкости пальца или измерении емкости между сенсорами. Емкостные сенсоры расположены вдоль вертикальной и горизонтальной осей тачпада, что позволяет определить положение пальца с нужной точностью.

Поскольку работа устройства основана на измерении емкости, тачпад не будет работать, если водить по нему каким-либо непроводящим предметом, например, основанием карандаша. В случае использования проводящих предметов тачпад будет работать только при достаточной площади соприкосновения.

Джойстик (от англ. Joystick = Joy + Stick) – устройство управления, преимущественно необходимое в компьютерных играх. Представляет собой рычаг на подставке, который можно отклонять в двух плоскостях. На рычаге могут быть разного рода гашетки и переключатели. Также словом «джойстик» в обиходе называют рычажок управления, например, в мобильном телефоне. Внутри джойстика расположены датчики, преобразующие угол и направление наклона рукоятки в соответствующие сигналы, передаваемые операционной системе. В соответствии с этими сигналами осуществляется перемещение и управление графических объектов на экране.

Дигитайзер, или графический планшет, – это устройство для ввода графических данных, таких как чертежи, схемы, планы и т. п. Он состоит из планшета, соединенного с ним визира или специального карандаша. Перемещая карандаш по планшету, пользователь рисует изображение, которое выводится на экран.

В современных планшетах основной рабочей частью также является сеть из проводов (или печатных проводников). Эта сетка имеет достаточно большой шаг (3–6 мм), но механизм регистрации положения пера позволяет получить шаг считывания информации намного меньше шага сетки (до 100 линий на мм). По принципу работы и технологии есть разные типы планшетов. В электростатических планшетах регистрируется локальное изменение электрического потенциала сетки под пером. В электромагнитных планшетах перо излучает электромагнитные волны, а сетка служит приемником. В обоих случаях на перо должно быть подано питание. Фирма Wacom создала технологию на основе электромагнитного резонанса, когда сетка и излучает, и принимает сигнал, а перо лишь отражает его. Поэтому в таком устройстве перо не требует никакого питания. Также есть планшеты, в которых нажим пера улавливается за счет пьезоэлектрического эффекта. При нажатии пера в пределах рабочей поверхности планшета, под которой проложена сетка из тончайших проводников, на пластине пьезоэлектрика возникает разность потенциалов, что позволяет определять координаты нужной точки. Такие планшеты вообще не требуют специального пера и позволяют чертить на рабочей поверхности планшета как на обычной чертежной доске.

Кроме координат пера, в современных графических планшетах также могут определяться давление пера на рабочую поверхность, наклон, направление и сила сжатия пера рукой.

Сканер – устройство ввода графических изображений в компьютер. В сканер закладывается лист бумаги с изображением. Устройство считывает его и пересылает компьютеру в цифровом виде. Во время сканирования вдоль листа с изображением плавно перемещается мощная лампа и линейка с множеством расположенных на ней в ряд светочувствительных элементов. Обычно в качестве светочувствительных элементов используют фотодиоды. Каждый светочувствительный элемент вырабатывает сигнал, пропорциональный яркости отраженного света от участка бумаги, расположенного напротив него. Яркость отраженного луча меняется из-за того, что светлые места сканируемого изображения отражают гораздо лучше, чем темные, покрытые краской. В цветных сканерах расположено три группы светочувствительных элементов, обрабатывающих соответственно красные, зеленые и синие цвета. Таким образом, каждая точка изображения кодируется как сочетание сигналов, вырабатываемых светочувствительными элементами красной, зеленой и синей групп. Закодированный таким образом сигнал передается на контроллер сканера в системный блок.

Более подробно принципы работы и устройство сканера будут описаны в следующей лекции.

Устройства вывода

Устройства вывода переводят информацию из машинного представления в образы, понятные человеку (или автомату, для которого эта информация предназначена).

После ввода пользователем исходных данных компьютер должен их обработать в соответствии с заданной программой и вывести результаты в форме, удобной для восприятия пользователем или для использования другими автоматическими устройствам посредством устройств вывода. При выводе данных, предназначенных для восприятия человеком, необходимо ориентироваться на его органы чувств. Таким образом, можно разделить устройства вывода на пять групп – воздействующие на зрение, слух, обоняние, осязание и вкус. Наиболее распространена группа устройств вывода, воздействующих на зрение (примерами являются мониторы, принтеры и плоттеры). Информация может также воспроизводиться в виде звуков с помощью акустических колонок или головных телефонов, регистрироваться в виде тактильных ощущений, запахов и вкусов в технологии виртуальной реальности.

Рассмотрим устройства, выводящие графическую информацию, предназначенную затем для визуального изучения. Для этого используются мониторы, принтеры или плоттеры.

Монитор (дисплей) является основным устройством вывода графической информации. По размеру диагонали экрана мониторы варьируют от 15 до 24 дюймов. Чем больше диагональ монитора, тем он дороже. Мониторы бывают монохромные (в настоящее время практически не используются) и цветные. Любое изображение на экране монитора образуется из светящихся разными цветами точек, называемых пикселями (это название происходит от PICture CELL – элемент картинки). Пиксель – это самый мелкий элемент, который может быть отображен на экране. Чем качественнее монитор, тем меньше размер пикселей, тем четче и контрастнее изображение, тем легче прочесть самый мелкий текст, а значит, и меньше напряжение глаз. Наиболее распространены жидкокристаллические мониторы (Liquid Crystal Display – LCD). Всего несколько лет назад были распространены мониторы с электронно-лучевой трубкой (Catode Ray Tube – CRT). Существуют также плазменные мониторы (на основе плазменной панели) и проекционные мониторы (видеопроектор и экран, размещенные отдельно или объединенные в одном корпусе).

Мониторы с электронно-лучевой трубкой имеют значительные размеры, потребляют много энергии и являются источниками вредного для человека излучения. Изображение на экране монитора формируется с помощью зерен люминофора – вещества, которое светится под

воздействием электронного луча. Различают три типа люминофоров в соответствии с цветами их свечения: красный, зеленый и синий. Цвет каждой точки экрана определяется смешением свечения трех разноцветных точек (триады), отвечающих за данный пиксель. Яркость соответствующего цвета меняется в зависимости от мощности электронного пучка, попавшего в соответствующую точку. Электронный пучок формируется с помощью электронной пушки. Электронная пушка состоит из нагреваемого при прохождении электрического тока проводника с высоким удельным электрическим сопротивлением, эмитирующего электроны покрытия, фокусирующей и отклоняющей системы.

При прохождении электрического тока через нагревательный элемент электронной пушки эмитирующее покрытие, нагреваясь, начинает испускать электроны. Под действием ускоряющего напряжения электроны разгоняются и достигают поверхности экрана, покрытой люминофором, который начинает светиться. Управление пучком электронов осуществляется отклоняющей и фокусирующей системой, которые состоят из набора катушек и пластин, воздействующих на электронный пучок с помощью магнитного и электрического полей. В соответствии с сигналами развертки, подаваемыми на электронную пушку, электронный луч побегает по каждой строчке экрана, последовательно высвечивая соответствующие точки люминофора. Дойдя до последней точки, луч возвращается к началу экрана. Таким образом, в течение определенного периода времени изображение перерисовывается. Частоту смены изображений определяет частота горизонтальной синхронизации. Это один из наиболее важных параметров монитора, определяющих степень его вредного воздействия на глаза. В настоящее время гигиенически допустимый минимум частоты горизонтальной синхронизации составляет 80 Гц, у профессиональных мониторов она составляет 150 Гц.

Жидкокристаллические мониторы имеют меньшие размеры, потребляют меньше электроэнергии, обеспечивают более четкое статическое изображение. В них отсутствуют типичные для мониторов с электронно-лучевой трубкой искажения. Принцип отображения на жидкокристаллических мониторах основан на поляризации света. Источником излучения здесь служат лампы подсветки, расположенные по краям жидкокристаллической матрицы. Свет от источника света однородным потоком проходит через слой жидких кристаллов. В зависимости от того, в каком состоянии находится кристалл, проходящий луч света либо поляризуется, либо не поляризуется. Далее свет проходит через специальное покрытие, которое пропускает свет только определенной поляризации. Там же происходит окраска лучей в нужную цветовую палитру. Жидкокристаллические мониторы практически не производят вредного для человека излучения.

Для получения копий изображения на бумаге применяют ***принтеры***, которые классифицируются:

- по способу получения изображения: литерные, матричные, струйные, лазерные и термические;
- по способу формирования изображения: последовательные, строчные, страничные;
- по цветности: черно-белые, цветные.

Наиболее распространены принтеры матричные, лазерные и струйные. Матричные принтеры схожи по принципу действия с печатной машинкой. Печатающая головка перемещается в поперечном направлении и формирует изображение из множества точек, ударяя иголками по красящей ленте. Красящая лента перемещается через печатающую головку с помощью микроэлектродвигателя. Соответствующие точки в месте удара иголок отпечатываются на бумаге, расположенной под красящей лентой. Бумага перемещается в продольном направлении после формирования каждой строчки изображения. Полиграфическое качество изображения, получаемого с помощью матричных принтеров, низкое, и они шумны во время работы. Основное достоинство матричных принтеров – низкая цена расходных материалов и невысокие требования к качеству бумаги.

Струйный принтер относится к безударным принтерам. Изображение в нем формируется с помощью чернил, которые распыляются через капилляры печатающей головки.

Лазерный принтер также относится к безударным принтерам. Он формирует изображение постранично. Первоначально изображение создается на фотобарабане, который предварительно электризуется статическим электричеством. Луч лазера в соответствии с изображением снимает статический заряд на белых участках рисунка. Затем на барабан наносится специальное красящее вещество – тонер, который прилипает к фотобарабану на участках с неснятым статическим зарядом. Затем тонер переносится на бумагу и нагревается. Частицы тонера плавятся и прилипают к бумаге.

Для ускорения работы принтеры имеют собственную память, в которой они хранят образ информации, подготовленной к печати.

К основным характеристикам принтеров можно отнести следующие:

- ширина каретки, которая обычно соответствую бумажному формату А3 или А4;
- скорость печати, измеряемая количеством листов, печатаемых в минуту;
- качество печати, определяемое разрешающей способностью принтера – количеством точек на дюйм линейного изображения. Чем разрешение выше, тем лучше качество печати;
- расход материалов: лазерным принтером – порошка, струйным принтером – чернил, матричным принтером – красящих лент.

Плоттер (графопостроитель) – это устройство для отображения векторных изображений на бумаге, кальке, пленке и других подобных

материалах. Плоттеры снабжаются сменными пишущими узлами, которые могут перемещаться вдоль бумаги в продольном и поперечном направлениях. В пишущий узел могут вставляться цветные перья или ножи для резки бумаги. Графопостроители могут быть миниатюрными и могут быть настолько большими, что на них можно вычертить кузов автомобиля или деталь самолета в натуральную величину.

Формирование цветного изображения на экране монитора или бумаге при печати происходит за счет смешения различных цветов. Для понимания принципа получения различных оттенков цвета на экране монитора или на бумаге необходимо иметь представление о формировании ощущений цвета у человека.

Представления о цвете и цветовом зрении

Видимый свет – это электромагнитные волны в диапазоне, видимом человеческим глазом (длины волн примерно от 380 до 760 нм). Обычно попадающий в наши глаза солнечный свет состоит из сравнительно однородной смеси лучей с различными длинами волн. Такую смесь называют белым светом. Монохроматический свет – это электромагнитные колебания на одной частоте. Близкий к монохроматическому свет можно получить, если отфильтровать белый свет в узкой полосе частот. На рис. 14.1 представлены цвета от красного (длинноволновая часть спектра) до фиолетового (коротковолновая часть спектра), отражающие восприятие человеком света с изменением его частоты. В табл. 14.1 приведены основные цвета видимого спектра и соответствующие им длины волн.

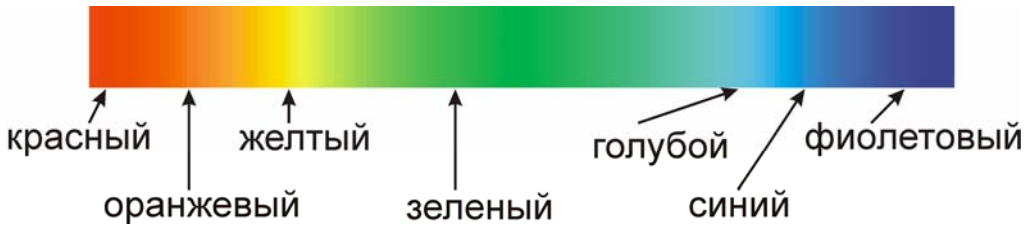


Рис. 14.1. Цвет в зависимости от длины волны излучения (от 760 нм в левой части до 380 нм в правой)

Таблица 14.1. Цвета видимого спектра и соответствующие им длины волн (нм)

Цвет спектра	**Соответствующая длина волны λ (нм)**
Красный	760–620
Оранжевый	620–590
Желтый	590–560
Зеленый	560–500
Голубой	500–480
Синий	480–450
Фиолетовый	450–380

Когда свет падает на некоторый объект, может происходить одно из трех событий: свет может поглощаться, а энергия его превращаться в тепло, как это бывает, когда что-то нагревается на солнце; он может проходить сквозь объект, если, например, на пути солнечных лучей окажется вода или стекло; либо он может отражаться, как в случае зеркала или любого светлого предмета, например куска мела. Обычно происходят в большей или меньшей степени все три события; например, одна часть света может поглотиться, другая часть – отразиться, а третья – пройти сквозь объект. Для многих объектов относительное количество поглощенного и отраженного света зависит от длины волны. Зеленый лист растения поглощает длинно- и коротковолновый свет и отражает свет промежуточной области спектра, так что при освещении листа солнечными лучами отраженный свет будет иметь выраженный широкий максимум на средних длинах волн (в области зеленого цвета). Красный объект будет иметь свой максимум, тоже широкий, в области длинных волн.

Вещество, которое поглощает часть падающего на него света и отражает остальную часть, называют пигментом. Если какие-то спектральные компоненты в диапазоне видимого света поглощаются лучше, чем другие, пигмент представляется нам окрашенным. Ощущение цвета зависит не только от длины волн, но также от распределения энергии между разными участками спектра и от свойств нашей зрительной системы. Здесь важны как физические свойства света, так и физиологические процессы, проходящие в процессе восприятия глазом информации.

Рассмотрим процесс восприятия света человеком. Свет поступает в глаз через отверстие в радужной оболочке (зрачок) и, проходя через хрусталик (где фокусируется) и стекловидное тело, попадает на сетчатку.

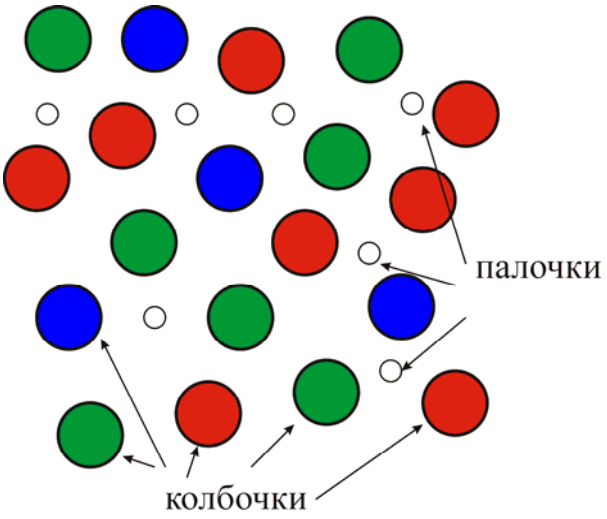


Рис. 14.2. Рецепторы сетчатки образуют мозаику, состоящую из палочек и трех типов колбочек (красные, зеленые, синие). Данная схема могла бы отображать участок сетчатки в нескольких градусах от центральной ямки, где колбочек больше, чем палочек

Сетчатка содержит своего рода мозаику из рецепторов четырех типов – палочек и трех типов колбочек (рис. 14.2). Каждый тип рецепторов содержит свой особый пигмент. Разные пигменты отличаются друг от друга в химическом отношении, а в связи с этим и по способности поглощать свет с различной длиной волн. Палочки ответственны за нашу способность видеть при слабом свете, т. е. за сравнительно грубую разновидность зрения, не позволяющую различать цвета. Палочковый пигмент родопсин обладает наибольшей чувствительностью в области около 510 нм, в зеленой части спектра. Палочки отличаются от колбочек во многих отношениях: они меньше и имеют несколько иное строение, по-иному распределены в разных частях сетчатки и имеют свои особенности в системе связей, образуемых с последующими уровнями зрительного пути. И, наконец, по содержащимся в них светочувствительным пигментам три типа колбочек отличаются как друг от друга, так и от палочек.

Зрительный пигмент обладает особым свойством: при поглощении им светового фотона он изменяет свою молекулярную форму и при этом высвобождает энергию, запуская цепь химических реакций, которые в конце концов приводят к появлению электрического сигнала и к выделению химического медиатора в синапсе. Пигментная молекула в своей новой форме, как правило, поглощает свет значительно хуже, чем в исходной. Затем сложный химический механизм глаза восстанавливает первоначальную конфигурацию пигмента; в противном случае его запас быстро истощился бы.

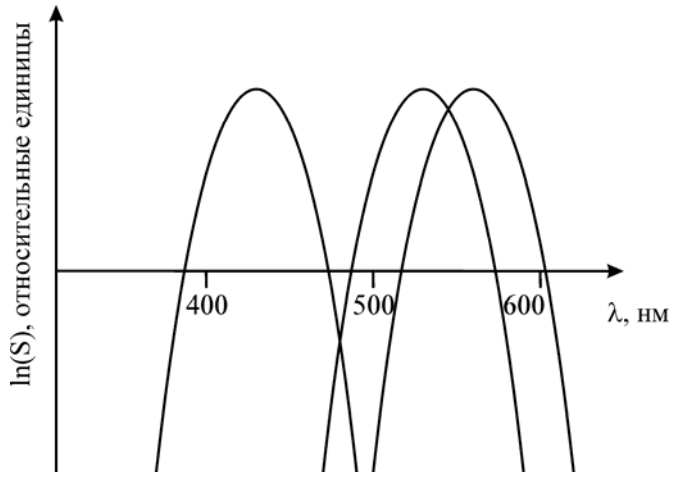


Рис. 14.3. Зависимости чувствительности *S* у колбочек трех типов от длины волны

Пигменты колбочек трех типов имеют пики поглощения в области 430, 530 и 560 нм (рис. 14.3); поэтому разные колбочки несколько неточно называют соответственно «синими», «зелеными» и «красными». Неточность состоит в том, что: 1) эти названия отражают максимумы чувствительности (которые в свою очередь зависят от светопоглощающей

способности), а не то, как эти пигменты выглядели бы, если бы на них можно было посмотреть; 2) монохроматический свет с длинами волн 430, 530 и 560 нм будет не синим, зеленым и красным, а фиолетовым, зеленым и желтым; 3) если бы можно было стимулировать колбочки только одного типа, мы видели бы не синий, зеленый и красный цвета, а, вероятно, фиолетовый, зеленый и желтый.

Однако приведенные выше названия колбочек широко распространены, а попытки изменить укоренившуюся терминологию обычно оканчиваются неудачей. Более корректными были бы названия «длинноволновые», «средневолновые» и «коротковолновые», но они затрудняли бы понимание для тех, кто не слишком хорошо знаком со спектром.

Имея максимум поглощения в зеленой области, палочковый пигмент родопсин отражает синие и красные лучи и поэтому выглядит пурпурным. Поскольку в наших сетчатках он присутствует в количествах, достаточных для того, чтобы химики смогли его выделить и можно было на него посмотреть, он издавна получил название зрительного пурпура. Само по себе это нелогично, поскольку «зрительный пурпур» называют так по его видимому цвету, тогда как названия для колбочек («красные», «синие» и «зеленые») соответствуют их относительной чувствительности, т. е. способности поглощать свет. Во избежание путаницы об этом следует помнить.

Три типа колбочек имеют широкие зоны чувствительности со значительным перекрытием, особенно для красных и зеленых колбочек. Свет с длиной волны 600 нм вызовет наибольшую реакцию красных колбочек, пик чувствительности которых расположен при 560 нм; вероятно, он вызовет также некоторую, хотя и более слабую, реакцию колбочек двух других типов. Таким образом, «красная» колбочка реагирует не только на длинноволновый, т. е. красный, свет; она лишь реагирует на него лучше других колбочек. Сказанное относится и к колбочкам других типов.

Таким образом, в явлении цвета можно выделить три основных процесса: *физический* – излучение энергии; *физиологический* – действие лучистой энергии на глаз и преобразование ее в энергию возбуждения нервных клеток органа зрения; *психологический* – восприятие цвета.

Мы рассмотрели физические и физиологические аспекты цветового зрения: природу света и пигментов, свойства объектов, отражающих свет, особенности пигментов, преобразующих поглощенный свет в электрические сигналы. Интерпретировать эти исходные сигналы как различные цвета – это уже задача мозга. Теперь рассмотрим вкратце историю вопроса и наиболее распространенную теорию цветового зрения.

Теория цветового зрения

С древних времен ученые пытались объяснить природу цвета. Однако вплоть до 60-х годов XVII века существовали самые неправдоподобные теории этого явления.

Еще Аристотель (384–322 годы до н. э.) считал, что причиной возникновения цветов является смешение света с темнотой. Подобные теории выдвигались и значительно позднее такими учеными, как Рене Декарт (1596–1650), Иоганн Кеплер (1571–1630), Роберт Гук (1635–1703). Причину цвета многие ученые того времени связывали со свойствами самого света, а не с работой глаза.

В 1664–1668 годах *Исаак Ньютон* (1643–1727) провел серию опытов по изучению солнечного света и причин возникновения цветов. Результаты исследований были опубликованы в 1672 году под названием «Новая теория света и цветов». Этой работой Ньютон заложил основу современных научных представлений о цвете. Изобретательность, которую проявил Ньютон в своих экспериментах, трудно переоценить: в работе, посвященной цвету, он при помощи призмы расщеплял белый свет; воссоединял его компоненты второй призмой, вновь получая белый свет; изготовил волчок с цветовыми секторами, при вращении которого опять-таки получался белый цвет. Эти открытия привели к осознанию того, что обычный белый свет состоит из непрерывного ряда лучей с различными длинами волн.

Хотя с тех пор наука о цвете получила большое развитие, многие положения, установленные Ньютоном, не утратили своего значения до наших дней.

Впервые наиболее близко к объяснению трехцветной природы зрения подошел великий русский ученый *М.В. Ломоносов* (1711–1765) в своем сочинении «Слово о происхождении света, новую теорию о цветах представляющую» (1765).

Постепенно выяснилось, что всякий цвет можно получить путем смешения трех цветных компонентов в надлежащих пропорциях при условии, что длины их волн достаточно отличаются друг от друга. Представление о том, что любой цвет может быть «составлен» путем манипулирования тремя управляющими факторами (в данном случае путем изменения интенсивности трех различных лучей), получило название трихроматичности. В 1802 году Томас Юнг выдвинул четкую и простую теорию, объясняющую трихроматичность: он предположил, что в каждой точке сетчатки должны существовать по меньшей мере три «частицы» – крошечные структуры, чувствительные соответственно к красному, зеленому и синему.

Решающие эксперименты, прямо и недвусмысленно подтвердившие, наконец, идею Юнга о том, что цвет должен определяться мозаикой трех видов детекторов в сетчатке, были проведены в 1959 году: Джордж Уолд и Пол Браун в Гарварде и Эдвард Мак-Никол и Уильям Маркс в Университете Джонса Гопкинса изучали под микроскопом способность отдельных колбочек поглощать свет с различной длиной волны и обнаружили три и только три типа колбочек. До этого ученые прилагали все усилия, используя менее прямые методы, и за несколько

столетий фактически пришли к такому же результату, доказав теорию Юнга о необходимости именно трех типов колбочек и оценив их спектральную чувствительность. Применялись в основном психофизические методы: ученые выясняли, какие цветовые ощущения вызывают различные смеси монохроматических лучей, как влияет на цветовое зрение избирательное обесцвечивание рецепторов под действием монохроматического света, а также исследовали цветовую слепоту.

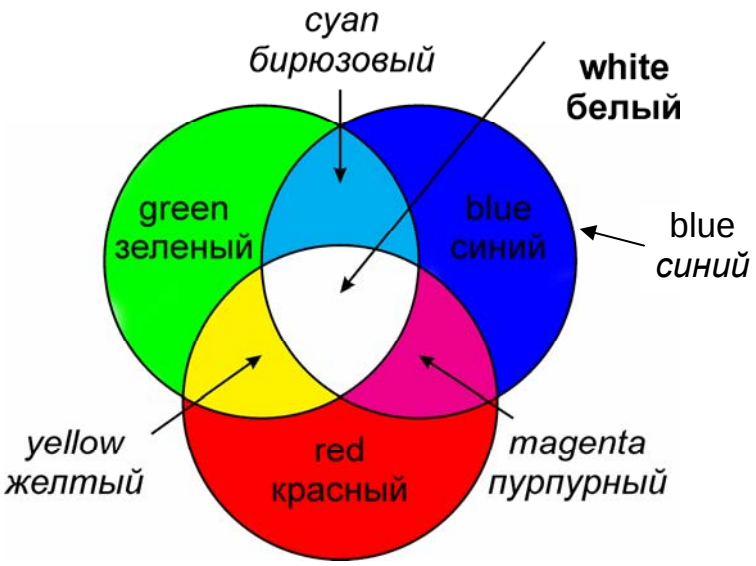


Рис. 14.4. С помощью трех диапроекторов и трех фильтров на экран проецируются три перекрывающихся пятна (красное, зеленое и синее). Красное и зеленое при наложении дают желтый цвет, синее и зеленое – бирюзовый, красное и синее – пурпурный, а все три вместе – белый цвет

Изучение эффектов смешения цветов необычайно интересно – настолько его результаты удивительны и противоречат интуиции. Никто без предварительного знания не угадал бы разнообразные явления, иллюстрируемые на рис. 14.4, – например, не мог бы предсказать, что два пятна, ярко-синее и ярко-желтое, при наложении друг на друга сольются в белый цвет, неотличимый на глаз от цвета мела, или что зеленый и красный спектральные цвета при их объединении дадут желтый, почти неотличимый от монохроматического желтого цвета.

Математически эквивалентная возможность определения цвета состоит в задании трех чисел, представляющих относительные влияния света на три типа колбочек. В любом случае необходимы три числа.

Представление цвета в пространстве RGB и отображение информации на экране монитора

В цветных устройствах визуального отображения информации используются принципы синтеза изображения, приближенные к физиологическим основам восприятия цвета. Так, если речь идет о приборах, излучающих свет, то в них используется представление оттенков красным,

зеленым и синим цветом (так называемая модель RGB). Это одна из наиболее распространенных моделей. Во всех устройствах отображения информации, независимо от принципа работы и технологии изготовления, общее цветное изображение складывается из большого числа элементов, каждый из которых представляет собой цветную триаду, т. е. три элемента красного (Red), зеленого (Green) и голубого (Blue) цветов. При смешении этих цветов с различной интенсивностью можно получить большое количество оттенков. При 24-битной глубине цвета интенсивность каждого из этих основных цветов может иметь 256 градаций (от 0 до 255), что соответствует одному байту. Следовательно, полный элемент изображения, состоящий из трех цветов (триада), занимает в памяти компьютера 3 байта и может воспроизвести 16 777 216 (т. е. 2^{24}) различных цветов. Например, чистый красный цвет характеризуется тремя байтами: 255, 0, 0. Это означает, что интенсивность красного (R) равна 255, а интенсивности зеленого (G) и синего (B) равны нулю. Чистому зеленому цвету соответствуют байты 0, 255, 0; чистому синему – байты 0, 0, 255. Другие цвета получаются смешением этих трех основных в различных пропорциях. Цветовая модель RGB относится к аддитивным моделям, поскольку цвет здесь образуется при помощи активных источников разных цветов (они добавляются друг к другу). Рис. 14.4 поясняет смешение трех основных цветов. Например, смешивание красного и синего цвета дает пурпурный, красного и зеленого дает желтый, а зеленого и синего – бирюзовый.

Размеры элемента изображения и количество таких элементов на поверхности устройства отображения информации характеризуют качество отображаемой визуальной информации. У современного 17-дюймового жидкокристаллического монитора разрешение составляет 1280 (по горизонтали) на 1024 (по вертикали) пикселов, то есть 1 310 720 элементов (триад) изображения.

Представление цвета при печати и цветной принтер

При печати на бумаге цвет получается не по аддитивной схеме (RGB), а по субтрактивной (вычитающей). Отличие заключается в том, что в первом случае цвет образуется при помощи красного, зеленого и синего лучей разной интенсивности. Если все они максимально яркие, то это белый цвет, а если у них нулевая яркость, то результирующим будет черный цвет. Для субтрактивной модели представления цвета – все наоборот. Когда интенсивность красок нулевая, то мы видим белый лист бумаги. При наложении всех красок получается черный цвет.

Дело в том, что при рассматривании напечатанного оттиска мы видим отраженные цвета, а не те, которые сами являются источниками света (как это было при синтезе цвета в мониторе). К отражаемым относятся цвета, которые сами не излучают, а используют белый свет, вычитая из него определенные цвета. Такие цвета называются субтрак-

тивными (вычитательными), поскольку они остаются после вычитания основных аддитивных: полиграфическая краска бирюзового цвета поглощает красный и отражает синий и зеленый цвета. Таким образом, в принтере используются краски, соответствующие так называемым дополнительным цветам (Cyan, Magenta, Yellow). Эти цвета составляют так называемую полиграфическую триаду. При печати они поглощают красную, зеленую и синюю составляющие белого цвета, и таким образом большая часть видимого цветового спектра может быть репродуцирована на бумаге. Каждому пикселу в таком изображении присваиваются значения, определяющие процентное соотношение триадных красок. При смешивании двух субтрактивных красок результирующий цвет затемняется, и можно получить один из основных цветов. Эффект смешения дополнительных красок также можно проиллюстрировать на рис. 14.4. Например, если взять бирюзовую краску (она пропускает зеленый и синий) и пурпурную (она пропускает красный и синий), то при их смешивании получим синий. Остальные лучи поглотятся слоями краски. При смешивании всех трех должен получиться черный цвет. При полном отсутствии красок остается белый цвет (белая бумага). Однако, несмотря на то, что теоретически при смешивании трех красок цвет должен быть черным, на практике так не получается. Это зависит от качества красок, спектров их отражения, которые не являются идеальными. Сложное смешивание цветов обычно приводит к тому, что совокупность трех красок дает темно-коричневый цвет. Поэтому для того, чтобы получить черный цвет, используется дополнительная черная краска. Цветные принтеры обычного качества используют картриджи CMYK (Cyan, Magenta, Yellow, blacK). В современных принтерах для получения более натурального цвета используется и большее количество красок (6–8).

Контрольные вопросы

1. Как классифицируются периферийные устройства?
2. Как формируется цветное изображение на экране компьютера?
3. Какими числами характеризуется пурпурный цвет в модели RGB?
4. Чем отличается принцип формирования цветного изображения на компьютере и в принтере?

Список литературы к лекции 14

1. *Мураховский В.И.* Железо ПК. Новые возможности. – СПб. : Питер, 2005. – 592 с. : ил.
2. *Хьюбел Д.* Глаз, мозг, зрение : пер. с англ. – М. : Мир, 1990. – 239 с. : ил.

Лекция 15. УСТРОЙСТВО И РАБОТА СКАНЕРА

Сканер является одним из широко распространенных устройств ввода графической информации в компьютер. Он предназначен для преобразования визуальной информации в цифровую форму, хранящуюся в компьютере. Сканеры используются для создания цифровых копий фотографий, рисунков, фотопленок, могут создавать файлы для дальнейшего использования в программах распознавания текста.

История сканера

Прообразом сканера принято считать прибор, изобретенный в 1857 году флорентийским аббатом Джованни Казелли, который был предназначен для передачи изображения на расстояние. Впоследствии этот прибор был назван пантелеграф. Передаваемая картинка наносилась на барабан токопроводящими чернилами, а роль считывающей головки в устройстве играла металлическая игла, с помощью которой происходило преобразование рисунка в электрический сигнал.

В 1902 году немецкий физик Артур Корн запатентовал технологию фотоэлектрического сканирования, получившую впоследствии название телефакс. Работа этого прибора была основана на принципиально новом технологическом решении с использованием фоточувствительного элемента. Передаваемое изображение закреплялось на прозрачном вращающемся барабане, луч света от лампы, перемещающейся вдоль оси барабана, проходил сквозь оригинал и через расположенные на оси барабана призму и объектив попадал на селеновый фотоприемник. Эта технология до сих пор применяется в наиболее дорогих (но и обеспечивающих наивысшее качество оцифровки) барабанных сканерах.

В дальнейшем, с развитием полупроводников и появлением матричных фотоприемников, был реализован планшетный способ сканирования, но сам принцип оцифровки изображения остается почти неизменным.

Принципиальное отличие сканера от родственных ему приспособлений заключено в судьбе преобразованного в электрическую форму изображения. В видеокамере электрический сигнал записывается на магнитную ленту или диск. В телефаксе сигнал передается посредством линий связи на большое расстояние, и изображение воспроизводится на специальном печатающем устройстве. В цифровых копировальных аппаратах изображение преобразуется в цифровую форму и печатается с помощью встроенного механизма лазерной печати. Сканер же после оцифровки передает изображение в компьютер.

Классификация сканеров

В современной технике сканером называется устройство, позволяющее вводить в компьютер образы изображений, представленных в виде текста, рисунков, слайдов, фотографий или другой графической информации. Сканер предназначен для преобразования изображений (текста, рисунков, фотографий и т. д.), представленных на бумаге или прозрачном носителе, в цифровые данные элементов изображения (пикселов). Термин «пиксел» (англ. pixel) произошел от объединения английских слов picture elements. Принцип работы сканера состоит в том, что он последовательно «просматривает» образец и генерирует соответствующие цифровые сигналы. Несмотря на обилие различных моделей сканеров, в первом приближении их классификацию можно провести по следующим признакам:

1. По степени прозрачности оригиналы можно условно разделить на две большие группы: прозрачные (слайды, негативная фотопленка, фотопластинки) и непрозрачные (рисунки на бумаге, фотографии, печатная продукция и т. д.). Для прозрачных оригиналов получение электронной копии изображения происходит путем просвечивания оригинала при помощи источника света. При сканировании непрозрачного оригинала необходимо анализировать отраженный от объекта световой поток.

2. По конструкции механизмы сканеров можно разделить на несколько групп в зависимости от области их применения[5]. *Планшетные* – наиболее распространенный вид сканеров, поскольку обеспечивает максимальное удобство для обычного пользователя – высокое качество и приемлемую скорость сканирования. Представляет собой планшет, внутри которого под прозрачным стеклом расположен механизм сканирования. *Ручные* – в них отсутствует двигатель, следовательно, объект приходится сканировать пользователю вручную, единственным его плюсом является дешевизна и мобильность, при этом он имеет массу недостатков – низкое разрешение, малую скорость работы, узкую полосу сканирования, возможны перекосы изображения, поскольку пользователю будет трудно перемещать сканер с постоянной скоростью. *Листопротяжные* сканеры относятся к настольным устройствам и отличаются компактными размерами. В устройствах этого типа страницы документа при считывании пропускаются через специальную щель с помощью направляющих роликов, как в факсимильных аппаратах. Именно поэтому такой сканер не может использоваться для работы с журналами и книгами, и пригоден только для сканирования отдельных страниц и рулонов чертежей. Многие модели имеют устройство автоматической подачи, что позволяет быстро сканировать большое количество документов. *Планетарные* (проекционные) сканеры – при-

[5] Интернет-ресурс. Режим доступа: http://ru.wikipedia.org.

меняются для сканирования книг или легко повреждающихся документов. При сканировании нет контакта со сканируемым объектом (как в планшетных сканерах). *Барабанные* сканеры применяются в полиграфии, имеют большое разрешение (около 10 тыс. точек на дюйм). Оригинал располагается на внутренней или внешней стенке прозрачного цилиндра (барабана). *Слайд-сканеры* – как ясно из названия, служат для сканирования пленочных слайдов, выпускаются как самостоятельные устройства, так и в виде дополнительных модулей к обычным сканерам. *Сканеры штрих-кода* – небольшие, компактные модели для сканирования штрих-кодов товаров.

3. По типу вводимого изображения сканеры делятся на черно-белые и цветные. Первые модели сканеров воспринимали только черный и белый цвет. По мере развития электроники сканеры научились получать картинки в градациях серого, а затем и в цветном режиме.

4. Аппаратный интерфейс связи с компьютером. Для связи с компьютером сканеры обычно используют стандартные интерфейсы, применяемые в IBM PC-совместимых компьютерах (последовательный и параллельный порты, интерфейс SCSI, интерфейс USB).

Программное обеспечение для сканирования

Для управления работой сканера (впрочем, как и иного устройства) необходима соответствующая программа – драйвер. В этом случае управление идет не на уровне «железа» (портов ввода-вывода), а через функции или точки входа драйвера. На первых порах каждый драйвер для сканера имел свой собственный интерфейс. Это было достаточно неудобно, поскольку для каждой модели сканера требовалась своя прикладная программа. Логичнее было бы наоборот, если бы с одной прикладной программой могли работать несколько моделей сканеров. Это стало возможным благодаря TWAIN –стандарту, согласно которому осуществляется обмен данными между прикладной программой и внешним устройством. TWAIN – это не аббревиатура, а просто название интерфейса, посредством которого сканер «общается» с компьютером. Слово TWAIN было взято из «Баллады о Востоке и Западе» Р. Киплинга: «...and never the twain shall meet...» (и двое никогда не встретятся), отражая существовавшую в то время сложность взаимодействия компьютера и сканера. После частого написания названия спецификации большими буквами сложилось предубеждение, что это аббревиатура, и были предложены такие варианты: Technology Without An Interesting Name (технология без интересного имени) или Toolkit Without Any Important Name (средство без какого-либо важного имени). Консорциум TWAIN был организован с участием представителей компаний Aldus, Caere, Eastman Kodak, Hewlett Packard и Logitech. Основной целью создания TWAIN-спецификации было решение проблемы совместимости, то есть легкого объединения различных устройств

ввода с любым программным обеспечением. Благодаря использованию TWAIN-интерфейса можно вводить изображение одновременно с работой в прикладной программе, поддерживающей TWAIN, например CorelDraw, Picture Publisher, PhotoFinish. Таким образом, любая TWAIN-совместимая программа будет работать с TWAIN-совместимым сканером.

Основные принципы работы сканера

Принцип работы планшетного (наиболее распространенного) сканера состоит в следующем. Сканирование выполняется при помощи светового луча. Источник света перемещается вдоль оригинала, освещая изображение. Примерная схема сканирования представлена на рис. 15.1.

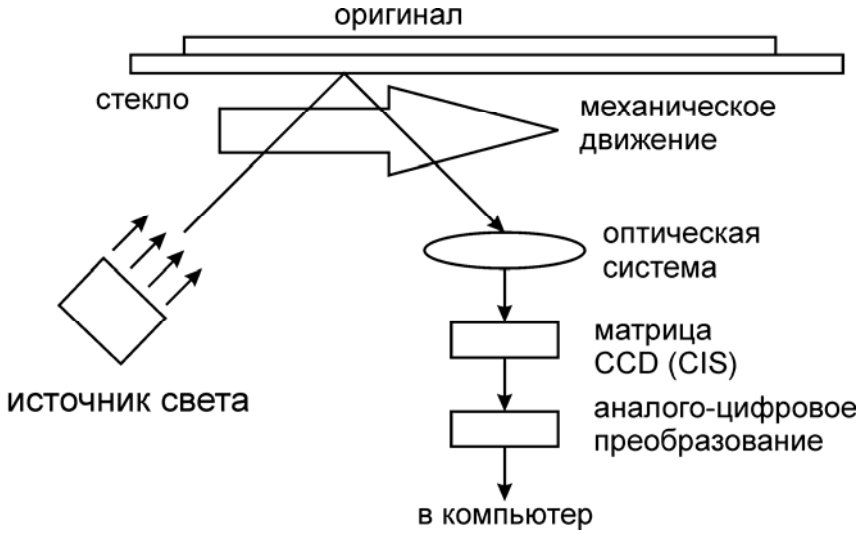


Рис. 15.1. Схема сканирования непрозрачного оригинала в планшетном сканере

Оригинал располагается на прозрачном неподвижном стекле, вдоль которого передвигается сканирующая каретка с источником света (если сканируется прозрачный оригинал, используется так называемый слайд-модуль – крышка, в которой находится вторая лампа). Оптическая система сканера (состоит из объектива и зеркал или призмы) проецирует световой поток от сканируемого оригинала на приемный элемент, осуществляющий разделение информации о цветах, – три параллельных линейки из равного числа отдельных светочувствительных элементов, принимающие информацию о содержании «своих» цветов. Приемный элемент преобразует уровень освещенности в уровень напряжения. Далее аналоговый сигнал поступает на аналого-цифровой преобразователь (АЦП), где он преобразуется в понятный для компьютера цифровой формат, и через контроллер сканера – в компьютер. В компьютере происходит программная обработка в драйвере сканера (TWAIN-модуль), после чего изображение может быть передано в прикладную программу.

Основной частью сканера является светочувствительная матрица. Большинство современных сканеров для дома и офиса базируются на матрицах двух типов: на CCD (Charge Coupled Device, в пер. с англ. – ПЗС (приборы с зарядовой связью)) или на CIS (Contact Image Sensor).

CIS-матрица состоит из светодиодной линейки, которая освещает поверхность сканируемого оригинала, микролинз и непосредственно самих сенсоров (фотодиодов или фототранзисторов). Конструкция матрицы очень компактна, таким образом, сканер, в котором используется контактный сенсор, всегда будет намного тоньше своего CCD-собрата. К тому же, такие аппараты славятся низким энергопотреблением; они практически нечувствительны к механическим воздействиям. Однако CIS-сканеры несколько ограничены в применении: аппараты, как правило, не приспособлены к работе со слайд-модулями и автоподатчиками документов. Из-за особенностей технологии CIS-матрица обладает сравнительно небольшой глубиной резкости. Для сравнения, у CCD-сканеров глубина резкости составляет ±30 мм, у CIS ±3 мм. У сканера с CCD-матрицей в конструкции предусмотрена система зеркал и фокусирующая линза. В свою очередь, именно достаточно громоздкая оптическая система и не позволяет CCD-сканеру достичь столь же компактных размеров, как у CIS-собрата. Однако, с другой стороны, именно оптика обеспечивает очевидный выигрыш в качестве.

Тем не менее CIS-технология также стремительно развивается. Сканеры с CIS-матрицей нашли свое применение там, где требуется оцифровывать не книги, а листовые оригиналы. Эти сканеры могут целиком получать питание по шине USB и не нуждаются в дополнительном источнике питания, что выгодно отличает его от более мощных и тяжелых сканеров с CCD-матрицей. Такие преимущества позволяют смириться с рядом недостатков контактного сенсора.

Физика приборов с зарядовой связью

Приборы с зарядовой связью (ПЗС) представляют собой систему взаимодействующих МДП-элементов, расположенных на общей полупроводниковой подложке. Взаимодействие обеспечивается не за счет соединительных проводов, а за счет малого расстояния между отдельными элементами.

На рис. 15.2, *а* схематично представлена структура прибора с зарядовой связью. В простейшей форме он представляет собой набор МДП-конденсаторов с единой подложкой. По аналогии с МДП-транзисторами металлические электроды называют затворами. На рисунке показан ПЗС с полупроводником, легированным примесями *n*-типа. Это означает, что электроны являются основными носителями, дырки – неосновными. Если к металлическим электродам приложить отрицательное напряжение *U*, то под затворами образуются обедненные

электронами слои, глубина l которых зависит от величины этого напряжения

$$l = \sqrt{\frac{2\varepsilon\varepsilon_0 U}{eN_d^+}},$$

где ε – диэлектрическая проницаемость диэлектрика; N_d^+ – концентрация ионизованных доноров.

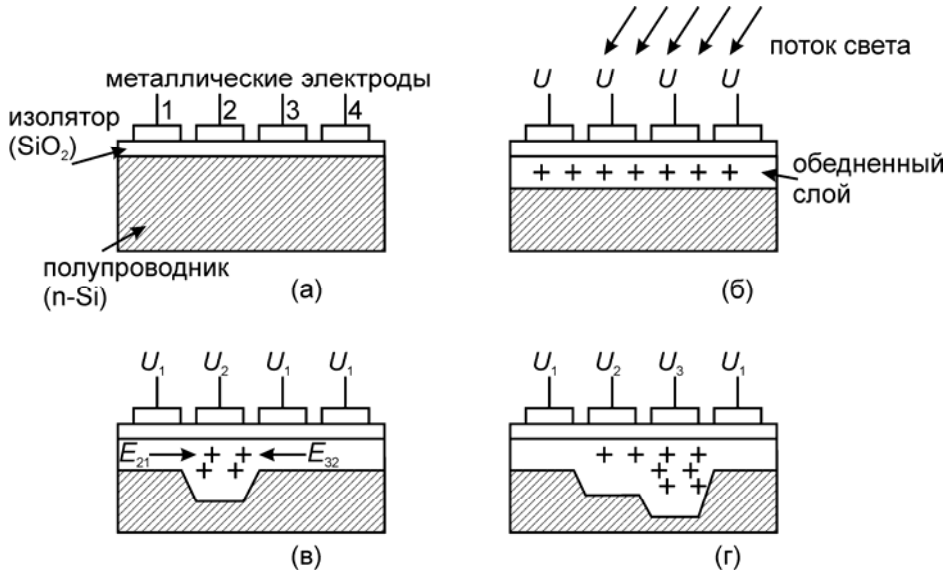


Рис. 15.2

а – структура ПЗС: металлические (возможно, прозрачные) электроды, напыленные на окись кремния (изолятор) на поверхности легированного полупроводника; *б* – режим покоя; *в* – режим хранения; *г* – режим записи

Поскольку расстояние между МДП-элементами небольшое, то их обедненные слои сливаются в единый слой. Если ко всем затворам приложено одинаковое отрицательное напряжение U, то обедненный слой вдоль всей поверхности имеет одну и ту же глубину (рис. 15.2, *б*). При попадании света в полупроводнике образуются электронно-дырочные пары, и дырки попадают в обедненный слой. Если же к какому-либо затвору, например, второму, приложено более отрицательное (большее по абсолютному значению) напряжение $|U_2| > |U_1|$, то под ним возникает более глубокий обедненный слой (рис. 15.2, *в*). При этом геометрическому изменению обедненного слоя соответствует изменение потенциального рельефа под затворами: в области увеличения обедненного слоя будет минимальное значение потенциала, т. е. возникает потенциальная яма для свободных дырок (по абсолютному значению максимальное значение потенциала).

Так как $|U_2| > |U_1|$, то на границах затвора 2 с затворами 1 и 3 электрические поля E_{21} и E_{23} препятствуют выходу положительных дырок из-под затвора 2 (рис. 15.2, *в*). Дырки, которые там оказались (или из-за инжектирования извне, или за счет освещения), попадают в потенци-

альную яму под затвором 2, и они могут находиться в этой области продолжительное время. Время их нахождения в потенциальной яме определяется не только тормозящими полями на ее границе, но и отсутствием свободных электронов, с которыми дырки могли бы рекомбинировать. Это состояние в ПЗС называется режимом хранения, а напряжение U_2 – напряжением хранения. Очевидно, что общий положительный заряд под затвором задается напряжением на затворе, поэтому инжекция дырок должна приводить к уменьшению ионизованных доноров в обедненном слое, т. е. к уменьшению глубины этого слоя. В этом случае тормозящие поля исчезнут и инжектированные дырки равномерно распределяться вдоль поверхности.

Максимальный заряд дырок под затвором равен

$$Q_{\max} = (U_2 - U_1)C_0 S_{затв},$$

где $S_{затв.}$ – площадь затвора, C_0 – удельная емкость диэлектрика, $C_0 = \varepsilon\varepsilon_0 / d$ (d – толщина диэлектрика).

Положительный пакет, хранящийся под затвором, на нашем рисунке под вторым, можно переместить под соседний затвор. Для реализации этого процесса перемещения дырок приложим к затвору 3 отрицательное напряжение $-U_3$, большее по абсолютному значению, чем напряжение $-U_2$ на втором затворе (рис. 15.2, *г*). Тогда на границе второго и третьего затворов возникнет электрическое поле, способствующее движению дырок к третьему затвору. Положительный пакет переместится под третий затвор и здесь останется, поскольку следующий, четвертый, затвор находится под напряжением $|U_1| < |U_3|$, и на границе третьего и четвертого затворов будет действовать тормозящее для дырок электрическое поле. Процесс перевода зарядового пакета от одного затвора к другому называют режимом записи информации, или режимом переноса, а напряжение U_3, обеспечивающее этот перевод, – напряжением записи.

Трехтактный регистр сдвига на ПЗС

Наиболее наглядно работу ПЗС можно рассмотреть на примере трехтактного сдвигового регистра, структура которого и схема его работы представлены на рис. 15.3. На этом же рисунке показан и один из способов ввода и вывода дырочного пакета с помощью *p*-*n*-переходов.

На рис. 15.3, *а* на шину, соединенную с затворами 1, 4, 7, 10, подается отрицательное напряжение U_1, на затворы 2, 5, 8, 11 – напряжение U_2, на затворы 3, 6, 9, 12 – напряжение U_3. Под затворами 3, 6, 9, 12 находится самая глубокая потенциальная яма для дырок неосновных носителей подложки из *n*-Si. Если в потенциальную яму под затвором 1 введен дырочный пакет, несущий какую-либо информацию, с помощью инжекции через *p*-*n*-переход со стороны входа или освещением данного участка, то дырки стекут в яму под затвором 3.

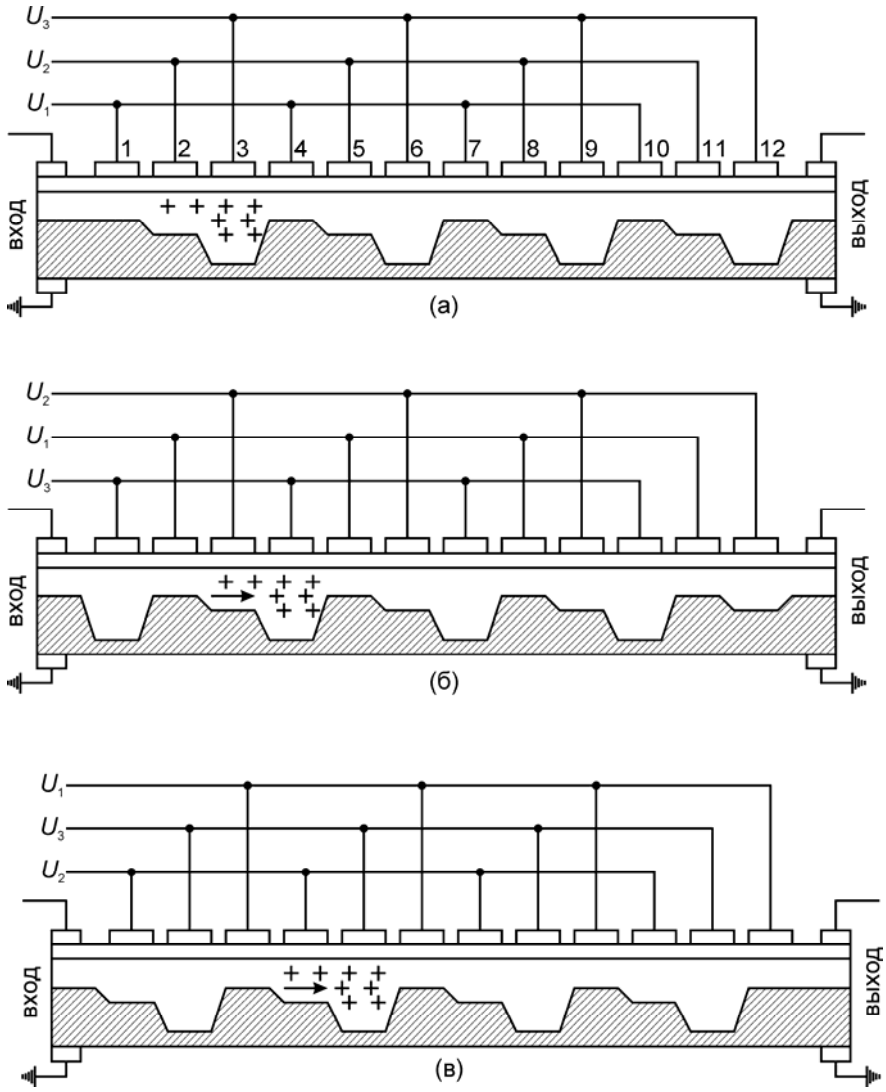


Рис. 15.3. Структурная схема и распределение потенциалов в трехтактном регистре сдвига в последовательные моменты времени (*а* – *в*).

После этого на шину, соединенную с затворами 1, 4, 7, 10, подается отрицательное напряжение U_3, на затворы 2, 5, 8, 11 – напряжение U_1, на затворы 3, 6, 9, 12 – напряжение U_2 (см. рис. 15.3, *б*). Зарядовый пакет перетекает под затвор 4. При следующем такте на шину, соединенную с затворами 1, 4, 7, 10, подается отрицательное напряжение U_2, на затворы 2, 5, 8, 11 – напряжение U_3, на затворы 3, 6, 9, 12 – напряжение U_1 (рис. 15.3, *в*). Пакет дырок переходит в потенциальную яму под затвором 5. Для того чтобы заряд, инжектированный p-n-переходом, прошел через всю структуру и оказался в области пространственного заряда выходного p-n-перехода, где его можно подать на выходное устройство, необходимо на шины подавать тактовые импульсы напряжения в такой последовательности:

- на нижнюю шину: U_1, U_3, U_2, U_1, U_3, U_2, U_1 …
- на среднюю шину: U_2, U_1, U_3, U_2, U_1, U_3, U_2 …
- на верхнюю шину: U_3, U_2, U_1, U_3, U_2, U_1, U_3 …

Эти шины управляющих сигналов часто называют *фазами* и соответственно весь прибор – *трехфазным* ПЗС.

Видно, что при таком способе питания управляющих шин зарядовые пакеты в приборе перемещаются вправо. Фазировкой управляющих импульсов можно добиться перемещения информации влево. Время передачи заряда от затвора к соседнему затвору определяется длиной затвора *L*:

$$t = \frac{L^2}{2.5D_p},$$

где *Dp* – коэффициент диффузии дырок. При длине затвора L = 20 мкм время передачи составляет 0.2 мкс. Естественно, что время передачи должно быть меньше времени существования потенциальных ям.

На практике для управления фазами $Ф_1$, $Ф_2$, $Ф_3$ трехфазными ПЗС используют последовательность импульсов трапецеидальной формы с затянутыми фронтами (рис. 15.4).

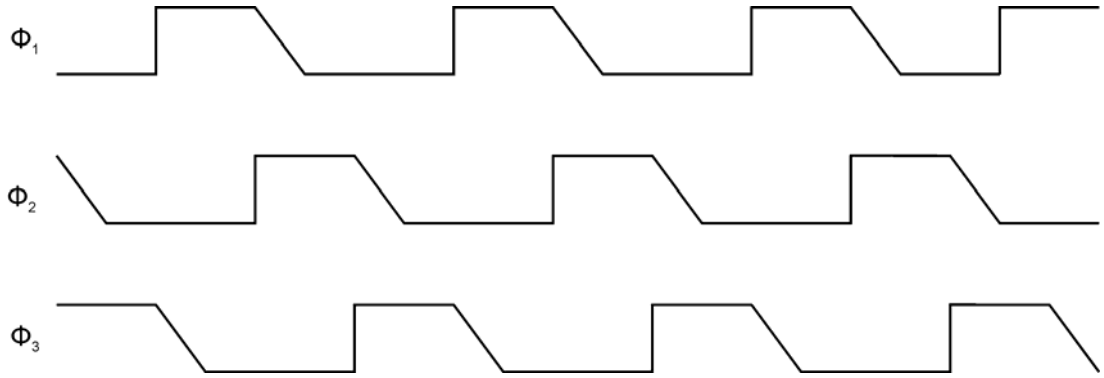


Рис. 15.4. Временная диаграмма управляющих импульсов трехфазного ПЗС

Со схемотехнической точки зрения создание такой управляющей схемы не очень удобно. Более простые управляющие сигналы (правда, за счет усложнения технологии изготовления) могут быть получены в двухфазных ПЗС.

Двухтактный регистр сдвига на ПЗС

Такой регистр представляет собой асимметричную конструкцию. Под каждым электродом для обеспечения асимметричного распределения поверхностного потенциала применен окисел различной толщины. Поскольку напряжение на электроде одно по всей поверхности, поверхностный потенциал под толстым окислом ниже, чем под тонким. Поэтому на участках с толстым окислом образуется потенциальный барьер, который не дает заряду, хранящемуся под тонким окислом, двигаться в противоположном направлении. Качественно двухтактный регистр сдвига на ПЗС представлен на рис. 15.5.

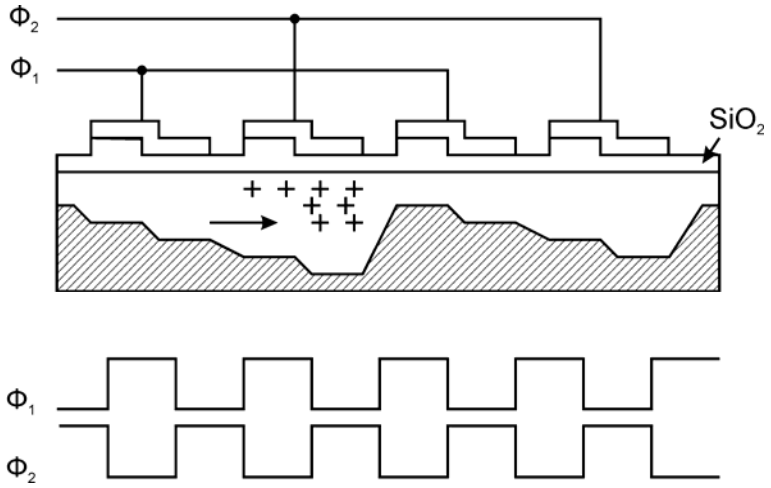


Рис. 15.5. Двухтактный ПЗС со ступенчатым окислом и соответствующая фазовая диаграмма

Технология изготовления двухфазного ПЗС дает возможность переносить информацию только в одном направлении. С другой стороны, такой прибор может быть более компактным, чем трехфазный.

Формирователь сигналов изображения при сканировании

Приборы с зарядовой связью являются замечательным преобразователем освещенности в пропорциональный электрический сигнал. Как уже было отмечено, они используются в большинстве сканеров, а также в огромном числе фотокамер и видеокамер. В отличие от электровакуумных приборов (видиконов), которые до недавнего времени использовались в качестве передающих трубок при телевизионном вещании и в видеокамерах, ПЗС обладает рядом преимуществ – масса, габариты и потребляемая мощность на порядок меньше, а прочность, надежность и продолжительность безотказной работы значительно выше.

Рассмотрим, как можно применить эту светочувствительную линейку в планшетном сканере. Внешне CCD-матрица представляет собой большую микросхему со стеклянным окошком. Именно сюда и фокусируется отраженный от оригинала свет. Матрица не прекращает работать все то время, пока сканирующая каретка, приводимая в движение шаговым электродвигателем, совершает путь от начала планшета до его конца. За один шаг матрица целиком захватывает горизонтальную линию планшета, которая называется линией растра. Если речь идет о черно-белом сканировании, то происходит засветка единственной линейки через объектив, а затем передача информации в компьютер, как это было показано на примере работы сдвиговых регистров на ПЗС. По истечении времени, достаточного для обработки одной такой линии, сканирующий блок перемещается на небольшой шаг, и наступает очередь для сканирования следующей линии, и т. д. Если сканер должен проводить полноцветное сканирование, то CCD-матрицу делают

состоящей из трех линеек, каждая из которых снабжена своим фильтром (красный, зеленый, голубой). Это позволяет получить цветовую информацию о сканируемом объекте.

Параметры сканеров

Существует множество параметров сканера, непосредственно зависящих от его целевого назначения. К ним относятся его технические характеристики и возможности проставляемого программного обеспечения. Перечислим основные параметры.

1. ***Разрешение***. Характеризует величину самых мелких деталей изображения, передаваемых при сканировании без искажений. Измеряется обычно в dpi – числе отдельно видимых точек на дюйм изображения (dot per inch). Существует несколько видов разрешения, указываемого производителем сканеров.

Оптическое разрешение определяется плотностью элементов в ПЗС-линейке и равно количеству элементов ПЗС-линейки, деленному на ее ширину. Оно является самым важным параметром сканера, определяющим детальность получаемых с его помощью изображений. В силу этого не всегда приводится в рекламной информации производителем или продавцом сканера, стремящимся завысить его реальные характеристики. В массовых моделях сканеров обычно оно бывает равно 100–600 dpi для ручных и листопротяжных сканеров и до 4800 dpi для планшетных сканеров.

Механическое разрешение определяет точность позиционирования каретки с ПЗС-линейкой при перемещении вдоль изображения. Механическое разрешение обычно в два раза больше оптического, что дает повод изготовителю сканера вводить в заблуждение покупателя тем, что сканер имеет «оптическое разрешение 1200×2400 dpi», хотя без интерполяции на таком сканере можно сканировать только с разрешением 1200 dpi.

Интерполяционным называется разрешение, полученное путем программного увеличения изображения. Оно не несет в себе абсолютно никакой дополнительной информации об изображении по сравнению с реальным разрешением, причем в специализированных пакетах операция масштабирования и интерполяции выполняется зачастую качественнее, чем драйвером сканера.

2. ***Глубина цвета***, или ***разрядность***. Характеризует количество бит, применяемых для хранения информации о цвете каждого пиксела. Черно-белые сканеры имеют один разряд, монохромные, как правило, 8 разрядов, а цветные сканеры, как минимум, 24 разряда (по 8 бит на хранение каждой из RGB-компонент цвета пиксела). Более совершенные сканеры могут иметь разрядность 30 или 36 (по 10 или 12 бит на каждый канал). При этом их внутренняя разрядность может быть выше внешней: «лишние» разряды используются для выполнения цветовой коррекции изображения до передачи в компьютер, хотя такая практика

в основном характерна для дешевых моделей. Профессиональные и полупрофессиональные сканеры имеют и внешнюю разрядность 30 или 36 бит (а некоторые модели и до 48 бит).

3. ***Диапазон оптических плотностей***. Это динамический диапазон сканера, который во многом определяется его разрядностью. Этот показатель применим к прозрачным оригиналам (негативная фотопленка, слайды) и отражает меру непрозрачности слоя вещества для световых лучей, то есть характеризует ослабление оптического излучения при прохождении через слой вещества. Вычисляется по формуле:

$$D = \lg(I_0 / I),$$

где I_0 – интенсивность излучения, падающего на поглощающую среду; I – интенсивность прошедшего излучения. $D = 0$ соответствует полной прозрачности, когда $I_0 = I$. Верхняя граница регистрируемых оптических плотностей зависит от разрядности: у 36-битного сканера он не превышает 3.6; у 30-битного 3.0.

Этот показатель можно также применять и к непрозрачным оригиналам, при этом принимается во внимание не прошедший поток света, а отраженный.

4. ***Размер области сканирования***. Для бытовых планшетных сканеров наиболее распространены форматы A4 и (существенно реже) A3, для рулонных сканеров – A4, а для ручных сканеров область сканирования составляет обычно полосу шириной 11 см.

5. ***Интерфейс***. Для подключения сканеров в настоящее время применяют следующие интерфейсы.

Собственный (Proprietary) интерфейс разработчика сканера, применявшийся в ранних моделях планшетных и ручных сканеров. Как правило, представлял собой специализированную плату на шине ISA, для работы которой требовался драйвер. После прекращения выпуска таких сканеров прекращался и выпуск новых драйверов для них, что делало невозможным использовать выпущенный в эпоху Windows 3.1 сканер под Windows NT, OS/2 или Linux. В настоящее время не применяется.

С параллельным портом EPP (LPT, или ECP) выпускались самые младшие модели в семействах планшетных сканеров различных производителей. Сканеры с таким интерфейсом имеют, как правило, посредственные характеристики и рассчитаны на выполнение несложных работ наподобие сканирования небольших фотографий или нескольких страниц текста. Использование параллельного порта совместно с принтером и дополнительными устройствами (например, Iomega Zip) часто приводит к трудноразрешимым аппаратным конфликтам и несовместимости. В настоящее время в силу того, что интерфейс LPT устарел, такие сканеры не выпускаются.

Интерфейс SCSI является стандартом для подключения высококачественных и высокопроизводительных устройств, обеспечивает межплатформенную совместимость сканера и его малую зависимость от

смены операционной системы. К SCSI-сканерам обычно прилагается SCSI-плата на шине ISA, хотя такой сканер можно подключать и к полнофункциональным SCSI-контроллерам на шине PCI.

Интерфейс USB – это новый, наиболее распространенный интерфейс для подключения сканеров, обеспечивающий универсальность подключения, безопасность (возможно горячее подключение) и высокую скорость обмена данными.

6. ***Качество драйвера***. Все современные сканеры обмениваются данными с прикладными программами под Windows при помощи программного интерфейса TWAIN, однако предоставляемый драйвером набор функций может быть разным, его обязательно следует уточнить при выборе сканера. Среди них наиболее важны:

- возможность предварительного просмотра изображения с выбором области сканирования, параметров разрешения и цветности;
- возможность регулировки яркости, контраста и нелинейной цветовой коррекции (обычно задаваемой в виде кривых);
- возможность подавления муара при сканировании изображений с печатным растром;
- возможность простейших преобразований изображения (инверсия, поворот и др.);
- возможность сетевого сканирования;
- возможность режимов автоматической коррекции контраста и цветопередачи;
- возможность работы сканера (в сочетании с принтером) в режиме копировального аппарата;
- возможности цветокалибровки как сканера, так и всей компьютерной системы;
- возможности пакетного сканирования;
- возможности тонкой настройки фильтров и параметров цветокоррекции.

7. ***Количество и качество прилагаемого к сканеру программного обеспечения***. Традиционно в комплекте со сканерами поставляются ПО обработки изображений (Adobe PhotoDeluxe или Photoshop LE, ULead Photo Impact и др.) и программа оптического распознавания текста (OCR – Optical Character Recognition). В комплект ПО обычно входят две таких программы: англоязычная (Xerox TextBridge или Caere OmniPage Pro) и предназначенная для распознавания русских текстов программа OCR отечественной разработки, обычно – одна из версий FineReader производства ABBY Software.

Контрольные вопросы

1. Для чего предназначен сканер?
2. Как устроены приборы с зарядовой связью?

Учебное издание

Пономаренко Владимир Иванович,
Лапшева Елена Евгеньевна

ИНФОРМАТИКА. ТЕХНИЧЕСКИЕ СРЕДСТВА

Подписано в печать 02.10.2009. Формат 60×84 $^{1}/_{16}$.
Усл. печ. л. 12,3. Тираж 100 экз.

Издательство «Научная книга»
410054, Саратов, ул. Б. Садовая, 127

Отпечатано с готового оригинал-макета
410005, Саратов, ул. Пугачевская, 161, оф. 320