КОГДА СДЕЛАНЫ УРОКИ

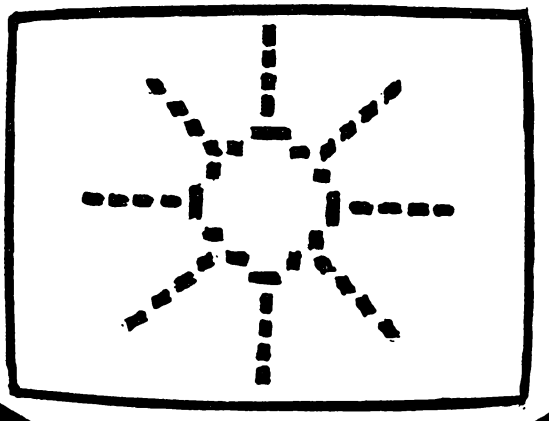
В. Г. ТИЩЕНКО, Г. В. ТИЩЕНКО

МИКРОЭВМ — СВОИМИ РУКАМИ

СЕРИЯ „КОГДА СДЕЛАНЫ УРОКИ“

В. Г. ТИЩЕНКО, Г. В. ТИЩЕНКО

МИКРОЭВМ — СВОИМИ РУКАМИ



Для старшего школьного возраста

КИЕВ „РАДЯНСЬКА ШКОЛА“ 1989

ББК 32.973.2
Т47

Рецензенты: *В. Н. Касаткин,* кандидат педагогических наук, доцент Симферопольского университета; *Я. Г. Господарский,* заведующий лабораториями микропроцессорной и вычислительной техники Тернопольского педагогического института.

Редакторы *О. П. Бондаренко, Л. Л. Розумова*

Художник-оформитель *К. И. Правдин*

Тищенко В. Г., Тищенко Г. В.

Т47 МикроЭВМ — своими руками: Для ст. шк. возраста.— К.: Рад. шк., 1989.— 208 с.— (Сер. «Когда сделаны уроки»)

ISBN 5—330—00747—X.

В книге занимательно и доступно излагаются сведения об элементной базе, структуре и принципах работы персонального компьютера — микроЭВМ. Даются развернутые советы по сборке, налаживанию и модернизации этого устройства, а также правила составления простейших программ. Прилагаются краткие справки о языках программирования и системе команд микропроцессора. Раскрывается смысл основных использованных в книге специальных терминов.

Для учащихся старших классов.

Т $\frac{4802020000—323}{M210(04)—89}$ 344—89

ББК 32.973.2

ISBN 5—330—00747—X

От авторов

Сегодня каждый, наверное, знает, что такое электронная вычислительная машина (ЭВМ), или, как ее еще нередко называют, компьютер. В самом деле, от года к году все больше и больше людей работают с ЭВМ, все больше специалистов зависят от сложнейших расчетов, выполняемых компьютерами. Компьютерам доверяют также управление производственными процессами, в их памяти можно хранить огромное количество данных. Компьютеры «обучаются» сочинять музыку, рисовать, переводить с иностранных языков, даже писать стихи...

Правда, до недавнего времени ЭВМ были громоздки, дороги, и, чтобы работать с ними, требовалось специальное образование.

В последние годы освоен выпуск компьютеров, которые пригодны практически для любого из нас. Теперь ЭВМ могут пользоваться не одни только специалисты в области электронной обработки информации. Называют такие ЭВМ по-разному: домашними, бытовыми, персональными, а иногда — микрокомпьютерами, или микроЭВМ.

Мы расскажем именно о них — о микроЭВМ. И еще о многом, что должен понимать любой грамотный современный человек, который хочет научиться работать с компьютером. Поэтому предлагаемая книга не только знакомит читателя с принципом работы микроЭВМ и правилами ее программирования, но и рассказывает о том, как самому (в домашних условиях или в радиотехническом кружке) собрать простейшую микроЭВМ и как ее использовать в учебных и практических целях.

Авторы надеются, что многие читатели, прочтя эту книгу, заинтересуются микропроцессорной техникой и у них возникнет желание углубить свои знания в этой области, усовершенствовать собранную микроЭВМ, попытаться превратить ее в современный персональный компьютер. В этом случае авторы будут полагать, что цель, поставленная ими при написании книги, достигнута.

Глава 1. ЧТО ТАКОЕ МИКРОПРОЦЕССОР

1.1. ПЕРВОЕ ЗНАКОМСТВО С МИКРОПРОЦЕССОРОМ

МикроЭВМ *, или **компьютер**,— сложное устройство, основной частью которого является **микропроцессор.** Поэтому, чтобы изучить микроЭВМ, необходимо прежде всего ознакомиться с микропроцессором, узнать принципы его действия и особенности применения.

Ознакомление начнем с разъяснения слова «микропроцессор» Первая часть его, приставка «микро», говорит о малых размерах этого устройства, а вторая — «процессор» — происходит от английского слова processor, означающего «обработчик», и указывает на обработку цифровых данных, буквенного текста, графических изображений и других видов сообщений или сведений, называемых **информацией.** При этом под обработкой понимают вычисление, сравнение, преобразование, пересылку и упорядочение поступающей информации.

Исходя из названия, можно коротко сказать: микропроцессор — это миниатюрное устройство, служащее для обработки информации. Однако такое определение неполное. Оно не отражает самого главного, характерного свойства микропроцессора — его способности обрабатывать информацию по программе, заложенной в специальное приспособление, называемое **памятью.** Меняя программу, один и тот же микропроцессор можно использовать для различных видов программной обработки, решая разнообразные задачи. Например, поместив в память данные о расположении шахматных фигур и программу их ходов, можно получить партнера для игры в шахматы. Если в эту же память заложить данные о соответствии русских слов — английским и программу поиска этого соответствия, то удается получить автоматический переводчик с русского языка на английский, и наоборот. Разместив в памяти сведения о температуре плавильной печи и составе компонентов, а также программу технологического процесса, получим возможность автоматически управлять режимом плавки металлов. Подобных примеров универсального использования одного и того же микропроцессора для решения различных задач можно привести очень много.

Помимо миниатюрности и программной обработки данных, микропроцессор имеет ряд других особенностей. Подобно радио-

---

* Ключевые слова, набранные полужирным шрифтом, поясняются в словаре, помещенном в Приложении (с. 187—199).

приемнику или телевизору, он состоит из электронных элементов — транзисторов. Разница только в их количестве и размерах. Если радиоприемник содержит несколько транзисторов, а телевизор — два-три десятка, то в микропроцессоре их насчитывается сотни тысяч. Транзисторы радиоприемника размещаются на пластине (называемой платой) величиной с папиросную коробку. Плата телевизора примерно в десять раз больше. А кристаллик микропроцессора с транзисторами имеет размер спичечной головки. Благодаря уникальной технологии, удается в столь крошечном объеме разместить огромное количество транзисторов и соединить их друг с другом по нужной схеме.

Характерной особенностью микропроцессора является также чрезвычайно высокое быстродействие, во много раз превышающее человеческие возможности. Микропроцессор может выполнять миллионы и даже миллиарды операций в секунду. Чтобы понять, как велика эта скорость, попробуйте, например, сосчитать в уме до миллиона. На это уйдет более трех месяцев ежедневного восьмичасового труда, а для счета до миллиарда не хватит и всей человеческой жизни. Или попытайтесь сложить в уме два числа 56 712 381 и 79 148 312. Не каждому удается это сделать. Сложение при помощи карандаша и бумаги займет примерно полминуты. А микропроцессор сложит эти числа мгновенно, за миллионную долю секунды. Такой молниеносный счет позволяет выполнять огромное количество различных вычислений за короткий промежуток времени, решая этим самым очень сложные ранее недоступные человеку задачи из различных областей науки и техники.

Помимо арифметических вычислений, микропроцессор может выполнять разнообразные логические действия в виде сопоставления обрабатываемых данных, поиска наилучшего значения какой-либо величины, определения аварийных состояний, создания сигналов управления типа: «открыть—закрыть», «ввести—вывести», «прочесть—записать» и многие другие.

Отличительной особенностью микропроцессора является также обработка информации, условно представляемой в виде чисел в двоичной системе счисления, которая, в отличие от привычной нам десятичной системы, имеет всего два символа: 0 и 1.

Микропроцессор имеет малую массу, относительно низкую стоимость, очень надежен в работе, прост в техническом обслуживании и потребляет мало электроэнергии. Благодаря этим качествам, он находит все большее применение во многих областях науки и техники, на производстве, в быту.

1.2. КОГДА И КАК ВОЗНИК МИКРОПРОЦЕССОР

Микропроцессор не является чьим-либо открытием или изобретением. Его появление — результат развития технологий производства радиоэлектронных элементов и совершенствования электронных вычислительных машин. Чтобы узнать, как был создан микропроцессор и как он работает, полезно проследить

путь развития радиоэлектроники и ЭВМ за последние десятилетия.

В 50-х годах XX столетия в радиоэлектронике беспредельно господствовала радиолампа. Вся радиотехническая аппаратура — радиоприемники, телевизоры, ЭВМ — собиралась на радиолампах. ЭВМ на лампах выглядела очень неуклюже. Одна из первых ЭВМ «ЭНИАК», насчитывающая 18 тыс. ламп, имела массу 30 тонн и занимала площадь 600 м^2. Наличие огромного количества ламп приводило к тому, что часто какая-либо из ламп перегорала и это затрудняло эксплуатацию ЭВМ. Лампы выделяли много тепла и требовались специальные охлаждающие установки. Для накала ламп нужна была собственная подстанция, мощности которой хватало для электроснабжения городского поселка. ЭВМ и другие электронные устройства, собранные на радиолампах, были названы электронной аппаратурой первого поколения.

В 60-х годах радиолампы уступили место полупроводниковому прибору — транзистору, который выполнял те же функции, что и радиолампа. По сравнению с лампой транзистор имел ряд преимуществ. Он был более долговечен, не требовал электрической мощности для накала, имел малые размеры и массу, в связи с чем всю электронную аппаратуру начали выпускать на транзисторах. Такая аппаратура получила название электронной аппаратуры второго поколения.

Транзисторов требовалось все больше и больше. Технология их производства непрерывно совершенствовалась и привела к тому, что транзисторы стали изготовлять «одним заходом», получая за один технологический цикл на одной кремниевой пластинке несколько десятков, а то и сотен транзисторов. После изготовления пластинку разрезали на отдельные транзисторы, упаковывали каждый в пластмассовый или металлический корпус и пускали в продажу. Приобретая транзисторы, их снова устанавливали на пластинке и соединяли друг с другом по заданной схеме. При этом невольно возникал вопрос: зачем транзисторы сперва разрезать, а потом снова соединять? Нельзя ли сразу, при изготовлении серии транзисторов на пластинке, соединять их по нужной схеме? Оказалось — можно. Ученые в содружестве с технологами разработали разные способы изготовления готовых электронных схем, состоящих из транзисторов и других электронных элементов. Чаще всего применяли многократное напыление нужного материала на кремниевую пластинку, называемую подложкой. Напыление производилось через отверстия специального трафарета. При этом распыляемый материал оседал в виде тончайшей пленки в нужных местах подложки, создавая определенный геометрический рисунок отдельных частей транзисторов, а также диодов, резисторов, конденсаторов и соединительных проводников. Иногда использовался и фотолитографический способ образования нужного узора пленки, основанный на фотохимическом процессе травления, напоминающем получение

печатного монтажа, которым часто пользуются радиолюбители. Эти и другие уникальные технологические приемы напыления законченных электронных схем (усилителей, генераторов, электронных реле, счетчиков и др.) привели к появлению новых электронных приборов, получивших название **интегральных схем.**

Начиная с 70-х годов интегральные схемы стали широко применяться при изготовлении многих электронных изделий: радиоприемников, телевизоров, магнитофонов, блоков автоматики и ЭВМ. Электронная аппаратура, собранная на интегральных схемах, стала называться аппаратурой третьего поколения.

Преимущество этой аппаратуры сводилось к резкому уменьшению ее габаритов, массы, стоимости, повышению надежности и главное — возможности автоматизации сборки деталей схемы, пайки, монтажа и налаживания.

В аппаратуре третьего поколения сначала применяли интегральные схемы с малой степенью интеграции, при которой в одном корпусе, на одной подложке создавалось несколько десятков транзисторов, диодов и резисторов, соединенных между собой по нужной схеме. Затем начали применять интегральные схемы со средней степенью интеграции, насчитывающие сотни и тысячи электронных элементов. Далее появились **большие интегральные схемы** (сокращенно **БИС**), сверхбольшие (СБИС) и ультрабольшие (УБИС), число элементов которых достигало сотен тысяч и миллионов. Электронная аппаратура с использованием БИС стала называться аппаратурой четвертого поколения.

Разработка и изготовление БИС оказались трудоемкими, сложными и, главное, очень дорогостоящими. Когда БИС представляла собой электронную схему широкого спроса (например, БИС для электронных часов или телефонных аппаратов), изготавливаемую в большом количестве экземпляров, то цена ее была умеренной. Стоимость же уникальных БИС, используемых в единичных экземплярах, оказывалась такой высокой, что приходилось воздерживаться от разработки и изготовления узко специализированных БИС. Эта ситуация изменилась лишь в середине 70-х годов, когда появилась и была реализована идея создания универсальной БИС, пригодной для изготовления различных электронных устройств путем изменения программы ее работы. Такая программируемая БИС получила название микропроцессора. Она могла быть выпущена промышленностью в большом количестве экземпляров по сравнительно низкой цене для самых разнообразных изделий. Некоторую роль в создании микропроцессоров сыграло и то обстоятельство, что, наряду с развитием технологии изготовления БИС, непрерывно совершенствовалось искусство составления программ для ЭВМ. К середине 70-х годов программирование достигло такого высокого уровня, при котором стала возможной реализация идеи создания универсальной программируемой БИС — микропроцессора.

Отметим, что до появления микропроцессора разработка и изготовление радиоэлектронной аппаратуры осуществлялись

традиционным методом, а именно: сперва разрабатывалась схема устройства, а затем подбирались нужные детали и устанавливались на плате. Далее выполнялся монтаж и, наконец, следовала серия операций по регулировке и налаживанию. Причем каждое новое устройство требовало своего комплекта деталей, своей схемы соединений и своих приемов налаживания. С появлением же микропроцессора для создания какого-либо иного электронного устройства совершенно не нужным оказался паяльник для монтажа и измерительные приборы для регулировки (так как изменение действий устройства с микропроцессором и выполнение им операций всецело зависит от программы, заложенной в память). Задумали мы, например, сделать электронное устройство для телевизионных игр. Составляем программу, помещаем ее в память и такое устройство готово. Сменив программу, вместо телеигры получаем электронный арифметический вычислитель (микрокалькулятор) или электронные часы, обучающее устройство или календарь, будильник или записную книжку, устройство для программируемого управления бытовыми приборами и др. Поначалу даже трудно поверить в такую метаморфозу, в мгновенную перемену действий микропроцессора.

На следующих страницах нас ждет близкое знакомство с этим удивительным электронным прибором. Мы поймем, в частности, почему и как происходит изменение действий микропроцессора в соответствии с заложенной программой.

1.3 ЧТО УМЕЕТ МИКРОПРОЦЕССОР

Трудно найти сферу человеческой деятельности, где нельзя было бы применить микропроцессор. Специалисты насчитали более ста тысяч профессий этого замечательного прибора, которые условно можно поделить на три класса:

1) встраивание микропроцессоров в технические изделия;
2) использование микропроцессоров для сборки микроЭВМ;
3) применение микропроцессоров для изготовления персональных компьютеров.

Рассмотрим особенности каждого из них.

Малые размеры и низкая стоимость микропроцессора позволяют легко устанавливать его во многие изделия, начиная от авиалайнера и кончая бытовыми приборами. При этом включение микропроцессора в состав технического устройства совершенно преобразовывает действие последнего, вдыхая в него новую жизнь. Приведем несколько наглядных примеров.

Микропроцессор, установленный в автомобиле, регулирует расход горючего, повышает безопасность движения, уменьшает токсичность выхлопных газов и упрощает управление, а если это такси, то еще следит за маршрутом, указывая водителю кратчайший, наиболее удобный путь, сообщает время прохождения маршрута и одновременно показывает на счетчике стоимость про-

езда для каждого из четырех пассажиров, вошедших в машину в разных местах.

Микропроцессор, включенный в телефонный аппарат, позволяет вызывать абонента нажатием кнопки, запоминать его номер и автоматически повторять вызов. Имеются телефонные аппараты с микропроцессором, позволяющие набирать номер голосом, путём произношения нужного номера перед микрофоном телефонной трубки.

Светофор с установленным микропроцессором регулирует движение на перекрестке в зависимости от количества автомашин, подъезжающих к нему, и числа пешеходов, желающих перейти проезжую часть.

Измерительные приборы, снабженные микропроцессором, очень упрощают, ускоряют и делают более точными измерения. Например, почтовые весы со встроенным микропроцессором сами подсчитывают стоимость почтового сбора, а цифровой вольтметр автоматически устанавливает предел измеряемого напряжения.

Широкие возможности открывает применение микропроцессора в сфере услуг и производстве бытовых товаров. Список товаров со встроенным микропроцессором непрерывно расширяется, и, очевидно, скоро наступит день, когда любой технический прибор, режим работы которого нужно изменять, будет содержать микропроцессор. Стоимость и габариты прибора от этого почти не увеличатся, а удобство и технические возможности резко возрастут

Благодаря микропроцессору современные роботы превращаются из бездушных автоматов-манипуляторов, умеющих только подавать и штамповать детали, в «думающих» мастеров «на все руки», способных обучаться, приобретать некоторые навыки и приспосабливаться к изменяющейся обстановке. Роботы, снабженные микропроцессором, умеют собирать и сваривать конструкции в сложных условиях, очищать днища кораблей от ракушек, ухаживать за больными, производить научные исследования, обучать детей, убирать в комнате, готовить обед и выполнять целый ряд других дел.

Второй областью использования микропроцессоров является создание на их основе микроЭВМ, способных решать многие задачи, которые до сих пор были посильны только большим, дорогостоящим ЭВМ. Малые размеры и низкая стоимость микроЭВМ позволяют им успешно соперничать с большими ЭВМ в выполнении несложных научно-технических и планово-экономических расчетов, а также в создании различных автоматизированных систем управления (АСУ).

Электронная промышленность освоила серийный выпуск нескольких типов микроЭВМ.

Если микроЭВМ используется для управления каким-либо технологическим процессом, то ее часто называют **микроконтроллером.** Это название произошло от английского слова

controller, что означает управление. (Слово «контроль» в русском и английском языках имеет разное значение. Следует иметь в виду, что микроконтроллер является не устройством контроля, а устройством управления).

В определенном смысле близки к микроЭВМ хорошо известные вам микрокалькуляторы.

Слово «калькулятор» образовано от латинского глагола calcus, что означает считать, вычислять. По своей структуре (или, как говорят специалисты, архитектуре) микрокалькуляторы очень похожи на микроЭВМ. Как и микроЭВМ, они имеют устройство ввода данных (клавиши от 0 до 9), микропроцессор, память, устройство вывода данных (цифровой индикатор) и источник питания. Отличие состоит лишь в способе программирования и в количестве выполняемых программ. В микрокалькуляторе программа работы задается нажатием клавиши. Нажмешь клавишу с надписью «+» — будет выполнена программа сложения, нажмешь «—» — включится программа вычитания. При этом количество программ равно числу программных клавиш. А в микроЭВМ число программ не ограничено. Любую программу можно записать в устройство памяти, и она будет выполняться.

В последнее время благодаря ряду усовершенствований грань между микрокалькулятором и микроЭВМ начинает стираться. Появились программируемые микрокалькуляторы, позволяющие составлять короткие программы для разнообразных вычислений. Отечественный микрокалькулятор МК-52, например, допускает запись пяти программ по 100 команд каждая. В отличие от всех ранее выпускаемых микрокалькуляторов, запись программ у МК-52 не стирается при выключении питания. Микрокалькулятор МК-72 позволяет подключение бытового телевизора в качестве **дисплея,** отображающего данные, и бытового магнитофона — как **внешнего запоминающего устройства**. Конечно, возможности микрокалькуляторов, даже модернизированного типа, далеки от возможностей микроЭВМ.

Третьей, самой молодой, но очень важной и наиболее перспективной, областью применения микропроцессоров является создание на их основе так называемых **персональных компьютеров.** Слово **компьютер** происходит от латинского computo, что означает считать, вычислять. Вместо трех слов — электронная вычислительная машина — чаще пользуются одним словом — компьютер. Таким образом, можно сказать, что персональный компьютер — это индивидуальная микроЭВМ, предназначенная, подобно телевизору или магнитофону, для личного пользования.

Как мы узнаем из дальнейшего, персональный компьютер способен выполнять очень широкий круг задач, начиная от сложных, профессиональных, и кончая простыми, бытовыми. Однако основной задачей персонального компьютера, по мнению одного из его создателей С. Джобса, является «индивидуальное усиление природных возможностей человеческого разума». Аналогично тому, как велосипед способен усилить мускульную энергию при

передвижении, так и персональный компьютер позволяет увеличивать эффективность мыслительной деятельности человека при выполнении им различной умственной работы.

Персональные компьютеры условно делят на профессиональные и бытовые.

Профессиональные компьютеры чаще всего используют для автоматизации конторских работ, что является само по себе чрезвычайно важным, так как конторский труд весьма распространен, а его производительность очень низкая. Персональный же компьютер резко, в десятки раз увеличивает производительность конторского труда. Происходит это оттого, что при составлении служебных документов приходится выполнять большое количество однообразной, утомительной, рутинной работы в виде поиска необходимых данных в различных справочниках, архивных делах, отчетах и книгах, поиска самих справочников и архивных дел. Много времени тратится на редактирование текста, организацию печатания документа, его оформление и пересылку.

При использовании персонального компьютера вся эта работа сводится к нулю. В память компьютера заранее закладываются тексты различных документов, которые могут быть использованы в качестве исходных. Путем нажатия клавиш изображение любого документа мгновенно вызывается на экран компьютера. Если в тексте нужно заменить какое-либо слово, то его стирание и вывод нового слова выполняется при помощи простых технических приспособлений — клавиш, ручек управления, светового пера и др.

После такого редактирования **пользователь** (так называют того, кто пользуется персональным компьютером) нажимает клавишу «печать» и получает отпечатанный документ. Если же документ пересылается лицу, имеющему персональный компьютер, то его можно и не печатать, а переслать по радио или телефонному каналу в виде электрических сигналов в память компьютера адресата, с автоматическим уведомлением о поступившем к нему документе при помощи условного сигнала и нужного текста на экране его компьютера.

Помимо автоматизации конторских операций, персональный компьютер широко применяется при выполнении проектно-конструкторских работ, научных исследований, медицинской диагностики и др. В последнее время им с успехом начинают пользоваться писатели, редакторы, художники и даже композиторы. При этом персональный компьютер, слушая мелодию, сочиняемую композитором, сразу записывает ее в виде нотных знаков на листе бумаги, а также, читая нотную запись, воспроизводит записанную мелодию.

Бытовые персональные компьютеры используются в основном для развлечений, путем выполнения большого количества разнообразных игр, сопровождаемых цветным, стереоскопическим изображением и гаммой сопутствующих звуков. Кроме того, их используют для управления бытовыми приборами, охраны квартир, обучения детей, изучения иностранных языков, выполнения

различных домашних расчетов за бытовые услуги, а также для осуществления доступа к общественным информационным центрам по телевизионному или телефонному каналам, с целью получения всевозможных справочных сведений о времени, погоде, расписании поездов, адресах и номерах телефонов различных учреждений и частных лиц, наличии товаров в магазинах, времени демонстрации и названий фильмов в кинотеатрах и т. д. Домашний компьютер может подсказать дату рождения родных и знакомых, ответить по телефону, записать на видеомагнитофон интересную телевизионную передачу, напомнить о намеченном мероприятии и выполнить большое количество других домашних дел.

В литературе появляются сообщения об использовании персонального компьютера для выполнения работы на дому. Многие люди смогут работать, не выходя из своей квартиры, что особенно удобно для женщин, имеющих малых детей, и пенсионеров. Следует иметь в виду, что сфера применения персональных компьютеров все время расширяется, а сам он непрерывно совершенствуется. Особенно широкие возможности, заложенные в персональном компьютере, откроются тогда, когда он сможет понимать человеческую речь и отвечать на человеческом языке. В настоящее время общение пользователя с компьютером выполняется, в основном, при помощи **клавиатуры** и текста, воспроизводимого компьютером на телевизионном экране. Иногда этот способ общения дополняется устройством синтезированной речи, графическими изображениями, получаемыми на экране, и печатающим устройством. Однако все это весьма далеко от привычного вида отношений между людьми. Поэтому во многих странах ведутся усиленные работы по совершенствованию средств взаимодействия человека и компьютера. По мере завершения этих работ общение будет все время приближаться к естественной форме, выполняемой при помощи разговорного языка и различных изобразительных средств. Работать с таким компьютером будет легко и просто. Отпадает необходимость в программировании, так как компьютер сам будет составлять программы в зависимости от сложившейся обстановки и словесных приказов пользователя.

С 1 июля 1987 г. в нашей стране введен ГОСТ на персональные компьютеры, согласно которому они называются **персональными машинами** (сокращенно **ПМ**) и делятся на пять классов: ПМ1 — индивидуального пользования, ПМ2 — микроЭВМ ученика, ПМ3 — микроЭВМ учителя, ПМ4 и ПМ5 — компьютеры для автоматизации профессиональной деятельности.

ГОСТ определяет также технические требования (быстродействие, разрядность, размер памяти и др.) для выпускаемых ПМ.

Глава 2. МИКРОПРОЦЕССОР И МИКРОЭВМ — УСТРОЙСТВО И ПРИНЦИПЫ РАБОТЫ

2.1. НЕМНОГО ТЕОРИИ

Об электронике

Микропроцессор — прибор электронный, а поэтому для успешной работы с ним нужно владеть азбукой электроники. Необходимо, в частности, знать, что такое электрический ток, напряжение, сопротивление. Уяснить последовательное и параллельное соединение сопротивлений и источников тока. Разобраться в общих принципах работы электронных элементов, из которых состоят средства вычислительной техники: резисторов, конденсаторов, диодов, транзисторов, логических элементов И, ИЛИ, НЕ. Усвоить работу простейших электронных устройств: триггера, регистра, генератора импульсов, выпрямителя; и наконец, научиться чертить и читать электрические схемы.

Чтобы лучше разобраться в перечисленных вопросах, полезно как можно чаще иллюстрировать физический смысл различных явлений с помощью аналогий и житейских сравнений, приводимых в популярной литературе. Например, электрический ток часто сравнивают с потоком воды в трубах. Как известно, заставить воду течь по трубе можно, прикладывая определенное усилие в виде напора, создаваемого водяным насосом или сосудом, поднятым на высоту. Аналогично, чтобы в проводнике возник электрический ток, также необходима некоторая сила, называемая электродвижущей силой, напряжением или разностью потенциалов и, в отличие от напора воды, создаваемая не насосом, а химическим (батарея, аккумуляторы) или механическим (динамомашины) способами, с помощью света (фотоэлементы) или тепла (термоэлементы).

Общеизвестно, что чем больше напор воды, тем с большей силой вода вытекает из крана. Аналогично: чем больше напряжение, тем больше сила тока. В этом состоит смысл основного закона электротехники — закона Ома.

Знакомясь с понятиями тока и напряжения, следует ясно представлять, что ток может протекать через какую-то точку схемы, а вот говорить о том, что в этой точке такое-то напряжение, недопустимо. Когда говорят или пишут: «На выводе микросхемы напряжение составляет плюс три вольта», то всегда имеют в виду напряжение между данным выводом и каким-то другим, чаще всего «общим», выводом источника питания. Чтобы понять сказанное, приведем такой пример: житель города Кисловодска может сказать, что дом, в котором он живет, расположен высоко, выше уровня моря на 900 м. В то же время другой житель этого же дома

может утверждать, что дом расположен низко, на 4147 м ниже вершины горы Казбек, находящейся рядом. Высота расположения дома зависит от того, по отношению к чему ее замеряют. Так происходит и с определением напряжения на выводе микросхемы. По отношению к одному выводу напряжение может составлять +5 В, по отношению к другому — 0 В, а по отношению к третьему — 5 В. Обычно принято измерять высоту расположения какой-либо местности или здания от уровня моря, а в электронных схемах измерение напряжения условились производить по отношению к «общему выводу» источника питания.

Чтобы понять смысл словосочетания «**общий** вывод», рассмотрим схему подключения источников питания к микропроцессору типа КР580ИК80А, рассмотренному далее (с. 50). Эта схема пригодится нам при сборке микроЭВМ.

Основная часть электронных элементов микропроцессора КР580ИК80А требует для своей работы источник питания, имеющий напряжение 5 В. Источник питания должен быть подключен так, как показано на рис. 1 вклейки. Вывод «+» источника питания нужно подключить к выводу *20* микропроцессора, а вывод «—» — к выводу *2*.

Некоторая часть элементов микропроцессора требует для своей работы напряжения 12 В. С этой целью к микропроцессору подключают второй источник питания, вывод «+» которого подключают к выводу *28* микропроцессора, а вывод «—» — к выводу *2*. И наконец, микропроцессор содержит ряд элементов, требующих напряжения 5 В, но отрицательной полярности. Поэтому приходится подключать третий источник питания, у которого вывод «—» подсоединяется к выводу *11* микропроцессора, а вывод «+» к выводу *2*.

Вывод *2*, к которому подключено напряжение — 5 В (первого источника тока),— 12 В (второго источника тока) и +5 В (третьего источника тока) называют общим выводом (иногда 0 В). По отношению к этому выводу указывают напряжение, имеющееся на всех других выводах микропроцессора.

Изучая принцип действия электронных элементов, можно, на первых порах, ограничиться упрощенным рассмотрением сложных процессов, происходящих в этих элементах. Достаточно, например, усвоить, что диод является полупроводниковым прибором, пропускающим ток в одном направлении — от анода к катоду (в направлении, указанном на рис. 2, *а* вклейки стрелкой-треугольником, условно обозначающей анод диода). Диод часто сравнивают с водяным клапаном, пропускающим воду в одном направлении, и в литературе, благодаря этой аналогии, диод иногда именуют клапаном.

Транзистор тоже можно рассматривать как устройство, пропускающее ток в одном направлении. У транзистора типа *р—n—р* (рис. 2, *б* вклейки) ток протекает от эмиттера к коллектору, а у транзистора типа *n—р—n* (рис. 2, *в* вклейки) от коллектора к

эмиттеру. Направление тока указывает стрелка на конце эмиттера. Отличие транзистора от диода заключается в том, что ток, протекающий через транзистор, управляется при помощи тока, подаваемого на базу. Если в цепи базы тока нет, то транзистор закрыт и ток через него не проходит. Подключение тока к базе (в нужной полярности) приводит к открытию транзистора и он начинает пропускать ток. Небольшой ток базы (единицы миллиампер) способен управлять большим (сотни миллиампер) током коллектора. Процессы, происходящие в транзисторе, довольно сложные, особенно когда транзистор работает в режиме усиления, преобразования или генерирования электрических сигналов. В микропроцессорной технике транзисторы используются в ключевом режиме, который сводится к включению или выключению электрического тока, протекающего через транзистор, путем подключения или отключения тока базы. Этот переключающий режим является наиболее простым, позволяющим упростить рассмотрение процессов, происходящих в транзисторе.

Помимо транзисторов биполярного типа (*p—n—p* и *n—p—n*), существуют полевые транзисторы, у которых прохождение тока управляется электрическим полем (отсюда название «полевой»), создаваемым напряжением, которое подается на вывод, именуемый затвором (рис. 2, *г* и *д* вклейки). Роль коллектора здесь выполняет сток, а эмиттера — исток. При этом следует, конечно, оговориться, что электрические процессы, происходящие в полевом транзисторе, совершенно не похожи на те, которые имеют место в биполярном транзисторе, также как и затвор конструктивно выполняется не так, как база, однако для первого знакомства достаточно пока знать, что ток в биполярном транзисторе управляется током базы, а в полевом — напряжением, подаваемым на затвор.

Наши читатели-радиолюбители уже имеют некоторое представление об электронике, позволяющее им собирать и налаживать простейшие радиоэлектронные устройства. Им мы советуем литературу [27, 34, 50, 54, 63] [1], дающую возможность пополнить свои знания об электронике в объеме, необходимом для изучения микропроцессора. А тем, кто еще не обладает электронной грамотностью, но желает познать устройство микропроцессора и овладеть сборкой и налаживанием микроЭВМ, порекомендуем литературу [10, 51, 53, 64].

Помимо теоретических знаний, необходимо иметь практические навыки в работе с электронными элементами и схемами. Поэтому перед сборкой микроЭВМ следует провести серию практических учебных опытов, собирая простейшие электронные схемы и наглядно уясняя принципы их действия. Примерная тематика таких практических опытов приведена в литературе [3, 5, 26, 27, 63].

---

[1] Здесь и далее в квадратных скобках указан номер, под которым рекомендуемая книга значится в списке литературы, приведенном на с. 205.

О системах счисления

Каждый, кто решил серьезно заняться изучением вычислительной техники, непременно должен владеть двоичной арифметикой, знакомство с которой предварим изучением систем счисления.

Под системой счисления понимают способ наименования (записи) чисел при помощи определенных знаков, чаще всего цифр. В наиболее привычной для нас десятичной системе счисления запись чисел выполняется десятью арабскими цифрами: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

С первых школьных лет мы так привыкли вести счет именно в десятичной системе, что она кажется нам самой простой и самой удобной. Многие даже полагают, что иначе считать нельзя. На самом же деле систем счисления может быть очень много, потому что каждое целое положительное число является родоначальником своей системы счисления, то есть может существовать единичная, двоичная, троичная, ..., десятичная, ... системы счисления.

Число, выбранное для образования системы счисления, называют ее основанием. С помощью основания можно определить три важнейших показателя системы: во-первых, узнать название системы; во-вторых, установить количество знаков (цифр), которыми записывают число в системе; в-третьих, определить значение (вес) каждой цифры в зависимости от ее положения в числе. Так, например, если выбрать для основания системы число 10, то получится известная нам система, называемая десятичной, которая пользуется десятью цифрами для написания чисел и у которой значение каждой цифры изменяется в зависимости от ее положения в числе в десять раз. В числе 333, написанном в десятичной системе, тройка справа означает «три». Такая же тройка, но написанная левее, означает «тридцать» (в десять раз больше), а еще левее — «триста» (снова в десять раз больше), то есть значение цифры 3 изменяется в зависимости от ее положения в числе в десять раз.

Выбрав основание системы, отличное от десяти, можно получить любую другую систему счисления.

При работе с микропроцессором наряду с десятичной пользуются двоичной и шестнадцатеричной системами. Поэтому далее будем рассматривать именно эти системы счисления, не касаясь остальных. Но прежде ознакомимся с понятием разрядности числа.

Разрядом называют то место, которое занимает цифра в данном числе. Самое правое, крайнее место именуют начальным, нулевым разрядом. Левее от него располагается первый разряд, еще левее — второй, затем — третий разряд и т. д.

Для примера возьмем число 1263 (рис. 1). В этом числе цифра 3 написана в начальном, нулевом, разряде; цифра 6 — в первом; цифра 2 — во втором; а цифра 1 — в третьем разряде. Сколько цифр в числе, столько в нем и разрядов. Поэтому вместо «четырехзначное» число, часто говорят «четырехразрядное» число. Таким

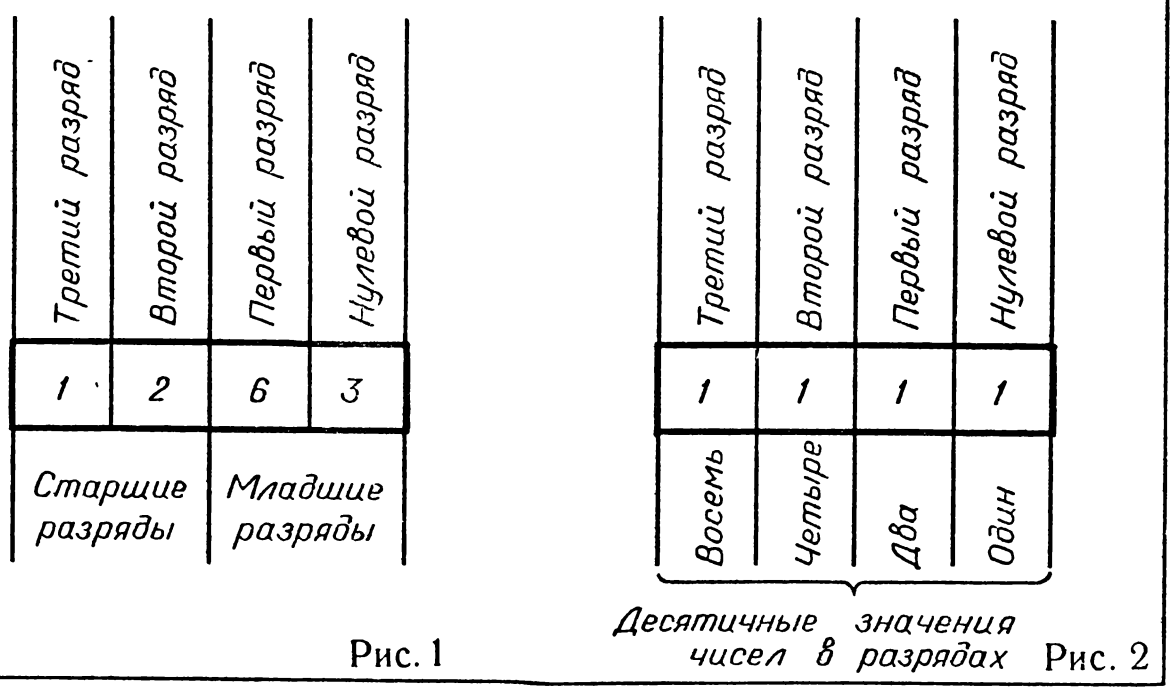


Рис. 1 Рис. 2

образом, число 76 будет двухразрядным числом, а 28365471 — восьмиразрядным. Количество разрядов в числе иногда условно делят пополам и разряды, расположенные справа, называют младшими, а расположенные слева — старшими. Так, в числе 1263 (рис. 1) нулевой и первый разряды с цифрами 3 и 6 будут называться младшими, а второй и третий разряды, с цифрами 2 и 1 — старшими. Кроме того, любой разряд, занимающий место левее данного, считается старшим по отношению к данному, а правее — младшим. Например, второй разряд, где находится цифра 2, будет старшим по отношению к первому разряду с цифрой 6 и младшим по отношению к третьему разряду с цифрой 1.

Как известно, счет каких-либо величин обычно ведут, начиная с единицы, один, два, три и т. д. А вот счет разрядов условились вести иначе, начиная с нуля, то есть считают так: нулевой разряд, первый разряд, второй разряд, третий и т. д.

Тот, кто начинает работать с микропроцессором, обычно забывает о таком порядке счета разрядов и допускает ошибку, называя, например, самый старший разряд восьмиразрядного числа — восьмым, в то время, как он является седьмым.

Двоичная система счисления

В этой системе счисления основанием является число 2, поэтому система называется двоичной; она пользуется двумя символами (0 и 1) для записи чисел и в ней значение символа изменяется от разряда к разряду в два раза.

Символ 1, записанный в нулевом разряде (рис. 2), означает, как и в десятичной системе,— «один». Символ 1, будучи записанным в первом разряде, соответствует десятичному «два» (в два раза больше), а записанный во втором разряде — десятичному «четыре» (снова в два раза больше), затем — «восемь», «шестна-

дцать» и т. д. С каждым разрядом значение (вес) символа 1 увеличивается в два раза.

Число 1111, написанное в двоичной системе счисления, читается так: «Один, один, один, один в двоичной системе», что означает: одна восьмерка, одна четверка, одна двойка и одна единица, а всего — число, равноценное пятнадцати в десятичной системе счисления.

При чтении чисел, записанных в двоичной системе, сначала возникают трудности. Невольно пытаемся рассматривать их как десятичные. Поэтому к двоичному счету приходится постепенно привыкать.

Если в каком-либо разряде данного числа записан 0, то значение этого разряда не входит в состав числа и им пренебрегают при чтении. Например, число 3002 в десятичной системе счисления читается так: «три тысячи два». О сотнях и десятках числа здесь ничего не говорится. Если же быть более точным, то это число следовало бы прочесть иначе, а именно: «три тысячи, нуль сотен, нуль десятков, две единицы». Соответственно, и в двоичной системе счисления число 1001 читается так: «Один, нуль, нуль, один в двоичной системе», что означает: восемь и один, всего — девять, или, точнее: одна восьмерка, нуль четверок, нуль двоек и одна единица, итого число, равноценное десятичной девятке.

Символы 1 и 0 в двоичной системе счисления не являются цифрами в том количественном понимании, к которому мы привыкли, пользуясь десятичной системой. Здесь они лишь служат условными знаками, показывающими, входит или нет десятичный эквивалент двоичного символа данного разряда в состав числа. Если в каком-либо разряде написано 1, то десятичное значение этого разряда (единица, двойка, четверка, восьмерка и т. д.) входит в состав числа, если же написано 0, то не входит.

Чтобы усвоить способ записи чисел в двоичной системе, полезно составить таблицу десятичных значений четырехразрядных двоичных чисел. В дальнейшем при работе с микропроцессором желательно запомнить эту таблицу, подобно тому, как мы когда-то запоминали школьную таблицу умножения. В первой колонке табл. 2.1 размещены различные сочетания символов 0 и 1 для четырехразрядных двоичных чисел, а во второй — соответствующее десятичное значение для каждого двоичного числа.

Тщательно разбираясь в этой таблице, нужно понять и твердо усвоить способ записи двоичных чисел. Для облегчения, прокомментируем несколько строк таблицы.

В первой строке первой колонки написано двоичное число 0000. Во всех его разрядах значится 0, следовательно, ни одна цифра из всех четырех разрядов не входит в состав числа. Число равно нулю. Во второй строке написано 0001. Здесь число определяется значением символа 1, записанного в нулевом разряде (в разряде единицы). Поэтому десятичный эквивалент числа будет равен единице. В третьей строке написано 0010. В этом числе

Таблица 2.1

Двоичные числа	Десятичные числа	Шестнадцатеричные числа
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

символ 1 записан в первом разряде (в разряде двойки), поэтому число равно двум. В четвертой строке записано 0011. Здесь символ 1 имеется и в разряде единицы, и в разряде двойки. Десятичный эквивалент числа будет три (2 + 1 = 3). Рассуждая аналогично, можно определить значение чисел во всех других строках таблицы.

Двоичная система счисления широко применяется в современных устройствах вычислительной техники благодаря удобству записи, хранения и простоте ввода двоичных чисел в эти устройства. Кроме того, арифметические действия с двоичными числами намного проще, чем с десятичными, что облегчает конструирование вычислительных устройств. Однако в двоичной системе счисления есть и недостатки. Они заключаются в большой длине (громоздкости) двоичных чисел и малой их наглядности. Возьмем, к примеру, два числа, записанных в десятичной системе: 232 и 143. Достаточно беглого взгляда на эти числа и мы сразу можем сказать, что первое число (232) примерно в полтора раза больше второго (143). Запишем те же числа в двоичной системе: 11101000 и 10001111. Что теперь можно сказать об этих числах? Во-первых, они стали менее выразительны и их трудно количественно оценить; во-вторых, их запись стала намного длиннее: они занимают теперь по восемь разрядов, вместо трех; в-третьих, при написании двоичных чисел легко ошибиться, написав вместо единицы нуль и наоборот.

Учитывая эти недостатки, двоичные числа часто записывают в шестнадцатеричном виде. Поэтому, составляя табл. 2.1, целесообразно предусмотреть в ней третью колонку для символов шестнадцатеричных чисел.

Эти символы можно будет вписать в таблицу после ознакомления с шестнадцатеричной системой счисления.

Шестнадцатеричная система счисления

В качестве основания этой системы счисления взято число 16, поэтому система называется шестнадцатеричной. Для записи чисел пользуются шестнадцатью символами и значение (вес) каждого символа изменяется от разряда к разряду в шестнадцать раз.

Символы шестнадцатеричной системы счисления записывают так: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E и F, то есть с 0 по 9 — в виде цифр, а с 10 по 15 — в виде латинских букв, где буква A означает десять, B — одиннадцать (и т. д. до буквы F, означающей пятнадцать).

Все перечисленные символы (от 0 до F), будучи написанными в самом крайнем правом (нулевом) разряде шестнадцатеричного числа, соответствуют значению цифр этого ряда (от 0 до 15). Такие же символы, но написанные в первом разряде, означают число, в шестнадцать раз больше, то есть 1 означает теперь 16 ($1 \times 16 = 16$), 2 означает 32 ($2 \times 16 = 32$), а F соответствует 240 ($15 \times 16 = 240$). Таким образом, число 111, записанное в шестнадцатеричной системе счисления, читается так: «Один, один, один в шестнадцатеричной системе счисления», что означает: двести пятьдесят шесть ($1 \times 16 \times 16 = 256$), плюс шестнадцать ($1 \times 16 = 16$), плюс один, итого двести семьдесят три в десятичной системе счисления.

Еще пример. Число F03. Оно расшифровывается так: «ЭФ, нуль, три в шестнадцатеричной системе счисления», что означает три тысячи восемьсот сорок ($15 \times 16 \times 16 = 3840$) плюс три, итого три тысячи восемьсот сорок три в десятичной системе счисления.

Если шестнадцатеричное число встречается в тексте совместно с буквенными символами, то для отличия чисел, записанных буквами (от A до F), от символов текста, левее шестнадцатеричного числа приписывают нуль. Например, пишут OAB или OCF2. Такой нуль, предшествующий старшему разряду любого числа, не имеет значения, его можно писать или не писать при любой системе счисления.

Шестнадцатеричные числа не применяются в микропроцессорной технике для выполнения вычислений. Они лишь используются как средство сокращения записи двоичных чисел при написании программ. О способах выполнения такой сокращенной записи будет сказано далее.

Обозначение систем счисления

Если написать число 101 и не указать его систему счисления, то возникает неопределенность, так как в десятичной системе это: «Одна сотня, нуль десятков, одна единица, а всего — сто один», в двоичной системе: «Одна четверка, нуль двоек, одна единица, а всего — пять», а в шестнадцатеричной: «Одно число двести пятьдесят шесть, нуль чисел шестнадцать и одна единица, а всего — двести пятьдесят семь».

Поэтому, применяя числа в устройствах, использующих различные системы счисления (например, в микропроцессорах), следует обязательно обозначать систему счисления для каждого числа.

Рассмотрим некоторые способы таких обозначений. В научной и технической литературе на немецком языке для обозначения чисел, написанных в двоичной системе, вместо символа 1 иногда употребляют латинскую букву L, записывая не 101, а L0L. Тогда 101 означает «сто один в десятичной системе», а L0L — «четверка плюс единица в двоичной системе».

В отечественной литературе для обозначения системы счисления чаще всего пишут внизу справа от числа мелким шрифтом, цифровой показатель (индекс), заключая его иногда в скобки. Например: запись 101_{10} или $101_{(10)}$ означает: сто один в десятичной системе счисления; запись 101_2 или $101_{(2)}$ говорит о том, что число написано в двоичной системе счисления. Читается оно так: «Один, нуль, один в двоичной системе», что означает: одна четверка, нуль и одна единица, или пятерка в десятичной системе ($101_2 = 5_{10}$).

Очень часто в отечественной и зарубежной литературе по микропроцессорной технике системы счисления обозначают при помощи заглавных латинских букв, следующих за цифрами. Для обозначения двоичной системы используется буква *B*, для восьмеричной 0 или *Q*, для десятичной *D* и для шестнадцатеричной *H*. Иногда буква *D* в десятичной системе опускается.

Таким образом, число «пять» может быть записано в десятичной системе как 5_{10}, $5_{(10)}$ или 5D, а также просто как 5; в двоичной системе — как 101_2, $101_{(2)}$ или 101B; в шестнадцатеричной — как 5_{16}, $5_{(16)}$ или 5H; $5_{10} = 5_{(10)} = 5D = 5 = 101_2 = 101B = 5_{16} = 5_{(16)} = 5H$.

Арифметика двоичных чисел

Арифметические действия над двоичными числами производят по тем же правилам, что и над десятичными, но так как двоичная система имеет всего два символа (1 и 0), то вычисления получаются значительно проще.

Вот так выглядит таблица сложения двоичных чисел:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= (1)0. \end{aligned}$$

Первые три строки этой таблицы пояснений не требуют. О записи четвертой строки нужно сказать следующее. Когда мы складываем один плюс один, то получаем два, но такой цифры в двоичной системе счисления нет, так как она пользуется только символами «единица» и «нуль». В то же время, если вспомнить, то «два» — это единица в более старшем разряде числа. Поэтому, получив двойку, мы переносим ее, то есть добавляем в старший

разряд в виде единицы, а в данном разряде пишем нуль: $1 + 1 = (1)0$.

Таблица вычитания для двоичных чисел пишется так:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$(1)0 - 1 = 1$$

В четвертой строке приведена запись для вычитания с займом. Так как от нуля отнять единицу нельзя, то приходится занимать единицу в старшем разряде. Но эта единица для данного разряда означает «два». Вычитая из этой двойки единицу, получаем — один, $(1)0 - 1 = 1$.

Таблица умножения двоичных чисел выглядит так:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

Она полностью соответствует таблице умножения для десятичных чисел.

В заключение приводим таблицу деления двоичных чисел:

$$0 : 1 = 0$$
$$1 : 1 = 1$$
$$0 : 0 = ?$$
$$1 : 0 = ?$$

Здесь, в двух последних строчках имеет место неопределенность, известная из школьной арифметики.

Из четырех арифметических действий наибольшее внимание следует уделить сложению, потому что в микропроцессоре все арифметические действия выполняются при помощи сложения. Объясним, как именно.

1. Вычитание. Действие вычитания можно заменить сложением, если к уменьшаемому добавить число, являющееся дополняемым к вычитаемому, то есть дополняет его до следующего старшего разряда. Сложив уменьшаемое и дополняемое, значение старшего разряда отбрасывают и получают результат вычитания. Объясним сказанное на примере вычитания десятичных чисел. Пусть требуется выполнить действие: 68 — 35.

Находим дополняемое к вычитаемому 35 (число, дополняющее 35 до высшего разряда — сотен). Это будет число 65, так как $100 - 35 = 65$. Складываем уменьшаемое 68 и дополняемое 65, получаем $68 + 65 = 133$. Отбросив старший разряд (сотню), имеем 33. Действие завершилось правильным ответом.

При десятичном счете к такому приему вычитания обычно не прибегают, так как вычисление дополняемого не менее сложно, чем выполнение самого вычитания. А вот для двоичных чисел этот прием применим. Здесь дополняемое число находится очень просто.

Оно определяется так. Заменяют в двоичном числе все единицы на нули, а нули — на единицы, получая так называемый обратный код двоичного числа. Прибавляя к этому обратному коду единицу в нулевом разряде, получаем нужное нам дополняемое число, сложение которого с уменьшаемым приводит к разности двух чисел.

Для уяснения этих приемов выполним вычитание двоичных чисел методом сложения. Воспользуемся теми же числами 68_{10} и 35_{10}, но записанными в двоичной форме:

уменьшаемое $1000100_2 = 68_{10}$
вычитаемое $0100011_2 = 35_{10}$

Находим обратный код вычитаемого:

0100011 (вычитаемое)
1011100 (обратный код вычитаемого)

Прибавляем единицу и получаем дополняемое (его часто называют дополнительным кодом):

$$\begin{array}{rl} & 1011100 \text{ (обратный код)} \\ + & \quad 1 \text{ (единица в нулевом разряде)} \\ \hline & 1011101 \text{ (дополнительный код)} \end{array}$$

Складываем уменьшаемое и дополнительный код

$$\begin{array}{rl} & 1000100 \text{ (уменьшаемое)} \\ + & 1011101 \text{ (дополнительный код)} \\ \hline & 10100001 \end{array}$$

Отбрасываем единицу старшего разряда:

0100001

Получаем итог вычитания:

$$0100001_2 = 33_{10}.$$

Процесс вычисления получился довольно длинным, хотя и простым. Такие операции микропроцессор выполняет очень быстро. Его электронная схема устроена так, что удобнее вычитание заменять сложением.

2. Умножение. При умножении двоичных чисел приходится умножать на единицу или на нуль, так как других цифр в двоичной системе нет.

При умножении на единицу множимое точно воспроизводится, а при умножении на нуль — получаются одни нули, которые можно и не писать, а просто сдвинуть следующее слагаемое на один разряд влево.

Рассмотрим пример:

$$
\begin{array}{r}
1001 \quad (9_{10}) \\
\times \quad 101 \quad (5_{10}) \\
\hline
1001 \\
0000 \\
1001 \\
\hline
101101 \quad (45_{10})
\end{array}
$$

Как видим, умножение здесь заменяется сложением двоичных чисел множимого, сдвинутых влево на нужное количество разрядов.

В микропроцессоре имеется специальное устройство (сдвигающий регистр), которое сдвигает двоичное число на требуемое количество разрядов при выполнении умножения.

3. Деление. Деление многоразрядных двоичных чисел является наиболее сложной и длинной процедурой из всех четырех арифметических действий. Не рассматривая его подробно, заметим, что деление сводится к многократному вычитанию и сдвигу чисел, но так как вычитание можно заменить сложением в дополнительном коде, то процесс деления состоит из таких трех операций:

1) сложения делителя в дополнительном коде с делимым;
2) сдвига;
3) сложения делителя с остатками, предварительно сдвигаемыми на каждом шаге деления влево на один разряд.

Представление двоичных чисел

Существует много способов представления чисел. Например, обычную пятерку можно обозначить при помощи символа 5, написанного на бумаге, или отобразить с помощью пяти косточек русских счетов, положенных справа налево, или же показать при помощи пяти раскрытых пальцев одной руки.

Еще больше способов существует для представления двоичных чисел. Это объясняется тем, что любое физическое устройство, имеющее два устойчивых состояния, может быть использовано для выражения двух символов (единицы и нуля) двоичной системы счисления, в то время как отображение десятичных чисел требует десять таких состояний (десять символов от 0 до 9, десять косточек на счетах, десять пальцев на руках и т. п.). Из всех способов отображения двоичных чисел укажем только те, которые чаще всего применяются в микропроцессорной технике.

Рассмотрим, к примеру, как можно отобразить двоичное число 1001. Такое число можно записать на бумаге в виде сочетания единиц и нулей, что мы и сделали. Можно собрать цепочку из четырех электрических лампочек и условиться, что каждая светящаяся лампочка будет обозначать единицу, а несветящаяся — нуль.

Иногда двоичное число отображают на бумажной ленте (называемой перфолентой) путем пробивания отверстий. Каждое пробитое отверстие считается единицей, а непробитое — нулем.

Пробивание отверстий часто заменяют намагничиванием специального магнитного слоя, нанесенного на магнитную ленту, барабан или диск. Намагниченный участок слоя считается единицей, а размагниченный — нулем.

Чтение такой магнитной записи выполняется при помощи электромагнитной головки. При прохождении намагниченного участка возле зазора головки в обмотке головки образуется импульс тока (единица), а при прохождении размагниченного участка — пауза (нуль).

Как будет указано далее, ввод двоичных чисел в микропроцессорной технике выполняют путем представления чисел в виде электрического напряжения. Высокий уровень напряжения означает единицу, а низкий — нуль.

На рис. 3, *а* вклейки показан ввод двоичного числа 1001 в электронный элемент DD1 при помощи переключателей П0—П3. В верхнем положении переключатель подсоединяется к источнику напряжения высокого уровня (+3 В), а в нижнем — к источнику низкого уровня (+0,3 В). С правой стороны к элементу DD1 подсоединяют светодиоды С0—С3, с помощью которых путем засвета верхнего (С0) и нижнего (С3) светодиодов отображено число 1001.

В дальнейшем очень часто будут применяться схемы, показывающие, как вводятся и выводятся двоичные числа в микропроцессорных элементах. Для большей наглядности и упрощения чертежей условимся обозначать переключатели, подающие уровень единиц, в виде красного кружка и отрезка прямой, направленного вверх (переключатели П0 и П3 на рис. 3, *б* вклейки), а переключатели, на выходе которых имеется уровень нуля,— в виде зеленого кружка и отрезка прямой, направленного вниз (переключатели П1 и П2 на рис. 3, *б* вклейки). На этих же схемах светодиоды, излучающие свет, мы будем обозначать кружками красного цвета (С0 и С3 на рис. 3, *б* вклейки), а неизлучающие — обычными, светлыми кружками (С1 и С2), окаймленными зеленой линией.

Все провода и элементы электрических цепей, относящиеся к высокому электрическому уровню (1), будем обозначать красным цветом, а относящиеся к низкому (0),— зеленым.

Очень часто в микропроцессорной технике двоичные числа представляют и запоминают при помощи триггеров. Триггер может находиться в одном из двух устойчивых состояний — в единичном и нулевом. В первом случае на его выходе имеется высокий уровень напряжения (единица), а во втором — низкий (нуль). Поэтому, установив четыре триггера (рис. 4 вклейки), называемые регистром, можно с помощью такого четырехразрядного регистра триггеров обозначить и запомнить двоичное число 1001. Для этого нужно крайние триггеры установить в единичное состояние, в внутренние — в нулевое.

Наш рассказ о других способах представления двоичных чисел мы продолжим при более подробном рассмотрении различных технических устройств, служащих для записи, хранения и воспроизведения двоичных чисел.

Перевод чисел из одной системы счисления в другую

Существует много способов перевода чисел из одной системы в другую. На наш взгляд, наиболее простым, удобным и самым надежным способом перевода небольших чисел является применение таблицы переводов. Эту таблицу можно составить, как и табл. 2.1, но длиной, примерно, в сто строк. Мы настоятельно рекомендуем составить такую таблицу. Она окажет большую помощь при написании и отлаживании программ. Кроме того, составляя таблицу, читатель получит необходимую практику перевода и сможет закрепить полученные сведения о системах счисления. Если под рукой такой таблицы не окажется или встретится число, большее, чем указано в таблице, то перевод нужно уметь выполнить с помощью расчета. При этом полезно запомнить следующие простые правила перевода.

1. ПЕРЕВОД ДЕСЯТИЧНОГО ЧИСЛА В ДВОИЧНОЕ

Такой перевод выполняется путем последовательного деления десятичного числа на 2 до тех пор, пока частное от деления не станет равным 1. Тогда все остатки, полученные от деления начиная с последнего, составят двоичную запись десятичного числа.

Пусть, например, требуется перевести в двоичную систему десятичное число 53. Делим это число на 2, но не доводим деление до конца, а обрываем его, как только остаток будет меньше двух (рис. 5 вклейки). Остаток получился 1. Запоминаем его, обводя для наглядности прямоугольником. Теперь берем частное 26 и снова делим на 2. В результате деления получается частное 13 и остаток 0. Обводим остаток 0 тоже прямоугольником, а с частным 13 поступаем также, как и прежде, то есть делим его на 2 и регистрируем прямоугольником остаток 1. Новое частное 6 опять делим на 2. Получаем частное 3 и остаток 0. Делим 3 на 2, фиксируя частное 1 и остаток 1. Если теперь записать все остатки, обведенные прямоугольником в обратной последовательности (в направлении, указанном стрелкой на рис. 5 вклейки), а в начале приписать 1 от частного, полученного при последнем делении (эта единица подчёркнута), то образуется такая запись десятичного числа 53 в двоичном виде:

$$\underline{1}\,\boxed{1}\,\boxed{0}\,\boxed{1}\,\boxed{0}\,\boxed{1} = 110101_2 = 53_{10}$$

Иногда процесс этого перевода несколько упрощают. Пишут заданное десятичное число, а справа от него проводят вертикальную линию вниз. Затем начинают устное деление числа на 2. При этом число может делиться на 2, тогда правее, за чертой,

ставят 0. Если же число не делится на 2, то ставят остаток 1. Результат же деления в обоих случаях подписывают под числом. В конце операции деления независимо от того получился остаток или нет, справа пишут единицу. Если теперь переписать снизу вверх все единицы и нули, расположенные правее черты, то получим искомое двоичное число.

Пример. Переведем десятичное число 53 в двоичное. Пишем 53 и правее проводим вертикальную черту вниз. Делим 53 на 2. Число 53 нечетное и на 2 не делится, поэтому пишем правее, за чертой, остаток 1, а результат деления 26 подписываем снизу под числом 53. Делим 26 на 2. Это число делится без остатка. Ставим правее 0, а под числом — результат деления 13.

53	1
26	0
13	1
6	0
3	1
	①

Число 13 на 2 не делится. За линией пишем 1, а результат деления 6 подписываем ниже числа. Делим 6 на 2. Деление выполняется без остатка. Правее линии пишем 0, а под числом результат деления 3. Это число на 2 не делится. Снова правее линии пишем 1 и ниже этой единицы — последний остаток 1, а правее, как говорилось выше, пишем 1 (для наглядности эта единица обведена кружком). Переписав снизу вверх (начиная с этой единицы, обведенной кружком) все единицы и нули, расположенные правее черты, получим искомое двоичное число: 110101.

Таким образом, $53_{10} = 110101_2$.

2. ПЕРЕВОД ДВОИЧНОГО ЧИСЛА В ДЕСЯТИЧНОЕ

Этот перевод выполняется путем сложения численных значений (веса) тех двоичных разрядов числа, где записана 1.

Пример. Дано двоичное число 101101. Требуется перевести его в десятичное. Так как символ 1 записан в нулевом, втором, третьем и пятом разрядах, а численное значение символа 1 в этих разрядах составляет: один, четыре, восемь и тридцать два, то полученная сумма $(32 + 8 + 4 + 1 = 45)$ образует искомое десятичное число 45.

При переводе многоразрядных чисел во избежание ошибок, рекомендуется двоичное число записать с некоторыми промежутками между символами. Например: 1 0 1 1 0 1 0 1. Затем, начиная с крайнего правого (нулевого) разряда, двигаясь справа налево, нужно последовательно писать возле каждого символа его численное значение. Если это символ 1, то надпись делается сверху над символом, а если 0, то снизу, под символом (рис. 6 вклейки).

Начинаем с нулевого разряда. Пишем 1 над единицей этого разряда, затем 2 под нулем первого разряда. Потом ставим 4

сверху, над единицей второго разряда. Поступая аналогичным образом, пишем в нужных местах: 8, 16, 32, 64 и 128. Сложив все числа, написанные сверху над символом 1, получим искомое десятичное число:

$$128 + 32 + 16 + 4 + 1 = 181$$
$$10110101_2 = 181_{10}$$

Для получения суммы верхних чисел, написанных над единицами, казалось бы, нижние числа можно под нулями и не писать. Однако это не так. Непрерывная запись последовательности чисел, каждый раз увеличивающихся в два раза (один-два-четыре-восемь-шестнадцать и т. д.) обеспечивает удобство и безошибочность записи, особо ощутимых при длинных, многоразрядных числах.

3. ПЕРЕВОД ДВОИЧНОГО ЧИСЛА В ШЕСТНАДЦАТЕРИЧНОЕ

Для сокращения записи двоичного числа и большей его наглядности применяют перевод в шестнадцатеричную форму. С этой целью двоичное число разбивают на группы по четыре символа, начиная с младшего разряда. Каждая такая группа называется **тетрадой.** Например, разбиение двенадцатеричного двоичного числа 101011010111 на тетрады приводит к такой записи:

1010 1101 0111

Затем, пользуясь табл. 2.1, заменяют значение каждой двоичной тетрады шестнадцатеричными символами. Для первой, левой, самой старшей тетрады (1010) это будет символ A. Для второй тетрады 1101 — символ D. Для третьей, самой младшей тетрады 0111,— символ 7.

В результате получаем искомую запись шестнадцатеричного числа AD7.

Таким образом,

$$101011010111_2 = AD7_{16}$$

Если количество разрядов двоичного числа не делится на четыре, то в старшей тетраде образуется число символов, отличное от четырех. В этом случае недостающие символы заменяют нулями.

Пример. Дано двоичное число 101011011 и требуется перевести его в шестнадцатеричный вид. Разбивка числа на тетрады, выполненная справа налево (от младших разрядов к старшим), приводит к такой записи:

1 0101 1011

В старшей тетраде оказывается записанным всего один символ. Недостающие три символа заполняем нулями:

0001 0101 1011

Теперь, пользуясь табл. 2.1, выполняем перевод значения каждой тетрады в шестнадцатеричную форму. Получаем:

$$0001_2 = 1_{16}$$
$$0101_2 = 5_{16}$$
$$1011_2 = B_{16}$$

Таким образом,

$$101011011_2 = 15B_{16}$$

4. ПЕРЕВОД ШЕСТНАДЦАТЕРИЧНОГО ЧИСЛА В ДВОИЧНОЕ

Этот перевод выполняется при помощи табл. 2.1 путем замены каждого символа шестнадцатеричного числа его двоичной тетрадой.

Пример. Имеем шестнадцатеричное число $2FB_{16}$. Требуется записать его в двоичной форме. Из табл. 2.1 находим для каждого символа шестнадцатеричного числа его двоичное значение:

$$2_{16} = 0010_2$$
$$F_{16} = 1111_2$$
$$B_{16} = 1011_2$$

Поэтому $2FB_{16} = 001011111011_2$.

5. ПЕРЕВОД ДЕСЯТИЧНОГО ЧИСЛА В ШЕСТНАДЦАТЕРИЧНОЕ И ОБРАТНЫЙ ПЕРЕВОД ШЕСТНАДЦАТЕРИЧНОГО ЧИСЛА В ДЕСЯТИЧНОЕ

Такие переводы редко встречаются в практике работы с микропроцессорами, поэтому мы опускаем их описание. В случае необходимости этот перевод можно выполнить двойным последовательным способом, который сводится к следующему. Десятичное число вначале переводится в двоичное, а затем двоичное — в шестнадцатеричное и наоборот. Шестнадцатеричное число переводится сперва в двоичное, а затем двоичное — в десятичное.

Вопросы о системах счисления, применяемых в микропроцессорной технике, доходчиво и в нужном объеме освещены в литературе [5, 11, 18, 23, 36, 46, 60, 63].

Немного о логических операциях

Помимо знаний о системах счисления, необходимых для изучения вычислительной техники, не менее важными являются понятия о логических операциях, выполняемых микропроцессором. При этом следует учесть, что разобраться в логических действиях микропроцессора гораздо труднее, чем усвоить его двоичную арифметику. Эти трудности объясняются тем, что логические действия основаны на законах и правилах математической науки, называемой алгеброй логики, изучение которой требует от начинающего читателя больших усилий.

Рассмотрим в популярной форме основные положения алгебры логики, чтобы читатель сознательно смог использовать их при освоении логических схем и составлении программ логических действий микропроцессора.

Науку о человеческом мышлении создал древнегреческий философ и ученый Аристотель (384—322 до н. э.). Он назвал ее логикой. Логика предписывала общие правила, по которым человек должен мыслить, рассуждать, делать умозаключения и приходить к истине.

Немецкий математик Г. В. Лейбниц (1646—1716) сблизил логику с вычислениями. У него возникла мысль создать новую науку — математическую логику, в которой логические понятия обозначались бы математическими знаками. «Если для этих знаков,— говорил Лейбниц,— установить правила соединения, то логические высказывания примут вид математических формул и логика станет искусством исчисления, позволяющим ученым и философам разрешать свои споры спокойными вычислениями». К сожалению, Лейбницу не удалось осуществить свою мечту. Только почти через двести лет английский математик Дж. Буль (1815—1864) частично реализовал идеи Лейбница.

Буль создал для логических обоснований и рассуждений необычную алгебру (булеву алгебру), в которой логические высказывания обозначаются особыми символами, подобно тому, как в школьной алгебре понятие числа или предмета обозначается буквами. Оказалось, что, оперируя этими символами и логическими связками в виде определенных знаков, можно выполнять логические рассуждения при помощи обычных вычислений.

Исчисление высказываний

Исследования показывают, что в человеческой речи чаще всего встречаются повествовательные предложения, излагающие что-нибудь или описывающие какие-нибудь события. Особенно много таких предложений в языке деловых отношений, а также в научной и технической литературе. Эти предложения называют высказыванием.

В булевой алгебре высказывание рассматривается не по содержанию и не по смыслу, а только в отношении того, истинно оно или ложно. Приведем примеры логических высказываний алгебры логики:

«Снег холодный» — высказывание и притом истинное.

«Дважды два четыре» — тоже высказывание и тоже истинное.

«Снег теплый» — высказывание, но ложное.

«Дважды два пять» — тоже высказывание и тоже ложное.

«Речка движется и не движется» (из популярной песни) — не высказывание, так как из этого предложения нельзя понять, ложно оно или истинно.

«Который час?» — это не высказывание, а вопросительная фраза. Вопросительные и восклицательные фразы, наряду с предложениями, выражающими просьбу, приказ или условность действий, не являются повествовательными предложениями, не употребляются для логических рассуждений и поэтому не могут быть высказываниями алгебры логики.

Истинность или ложность логического высказывания называется значением истинности и условно обозначается символом 1, если высказывание истинно, и 0, если оно ложно. Кроме того, само высказывание формально (условно) обозначается буквами латинского алфавита (A, B, C или x, y, z, ...), мысленно отвлекаясь от конкретного содержания высказывания, то есть не принимая его во внимание.

Поначалу такое абстрактное отношение к логическому высказыванию воспринимается с трудом. К нему приходится привыкать. Но привыкнув, удается понять, что формальное представление высказывания позволяет не только исчислять высказывания, то есть численно выражать логические мысли и действия, а и обрабатывать их при помощи электронных элементов и узлов микропроцессора.

В качестве примера заметим, что высказывание «дважды два четыре» можно условно обозначить буквой A и, принимая во внимание, что оно истинно, написать $A = 1$. Аналогично, высказывание «снег теплый» обозначить буквой B и написать $B = 0$, так как это высказывание ложно.

Высказывания вроде «земля круглая», «листья зеленые», «сахар сладкий» и им подобные являются простейшими. Буль показал, что простейшие высказывания, связанные между собой символами И, ИЛИ, НЕ, образуют сложное составное высказывание. В то же время каждое сложное высказывание может быть выражено через простейшие с помощью этих же связывающих слов И, ИЛИ, НЕ. Выполняя действия со сложными высказываниями, надо помнить о принятой формальной абстракции, помнить о том, что для алгебры логики важно не содержание, не смысл сложного высказывания, а только его значение истинности, в зависимости от ложности или истинности составлющих простейших высказываний.

Попробуем уяснить эту зависимость. Возьмем к примеру, такое высказывание: «Завтра будет тепло и завтра будет сухо». Это сложное высказывание в виде прогноза хорошей погоды состоит из двух простейших высказываний. Первое высказывание говорит о том, что завтра будет тепло, а второе о том, что завтра будет сухо. Оба они соединены связывающим союзом И. Если обозначить первое высказывание буквой A, связывающее слово буквой И, второе высказывание буквой B, а сложное высказывание (прогноз погоды) буквой Q, то условно, отвлекаясь от содержания высказывания, формально можно написать

$$A \text{ И } B = Q.$$

Чтобы выяснить значение истинности сложного высказывания в зависимости от значения истинности составляющих высказываний A и B, вспомним, что A и B могут быть либо истинными (равными 1), либо ложными (равными 0). При этом возможны такие четыре варианта сочетаний значения истинности A и B, сведенных в табл. 2.2.

Таблица 2.2

Вариант	Значение истинности		
	A	B	A И $B = Q$
1	1	0	0
2	0	1	0
3	0	0	0
4	1	1	1

Пользуясь здравым смыслом, покажем правильность значений Q, записанных в четвертой колонке.

Рассмотрим вариант 1, при котором $A = 1$ и $B = 0$. Когда $A = 1$, то первое простейшее высказывание истинно, оно говорит о том, что завтра будет тепло. Однако $B = 0$, а это означает, что второе высказывание ложно, то есть завтра не будет сухо, будет дождливо. Если, поверив прогнозу, мы не захватим зонтик и промокнем, то, несмотря на теплую погоду ($A = 1$), справедливо сможем считать прогноз хорошей погоды ложным, требующим записи в четвертую колонку таблицы значения $Q = 0$.

Рассмотрим вариант 2. Когда $A = 0$ и $B = 1$, то погода будет холодной, хотя и сухой. Продрогнув от холода, можно с уверенностью сказать, что предсказание погоды не оправдалось и в четвертой колонке нужно снова записать $Q = 0$.

При варианте 3, когда $A = 0$ и $B = 0$, то есть когда и холодно и дождливо, то, конечно, каждый посчитает прогноз ложным, требующим записи $Q = 0$ в четвертой колонке.

Только тогда, когда $A = 1$ и $B = 1$ (вариант 4), то есть когда будет тепло и сухо, мы вправе утверждать, что прогноз оправдался, оказался истинным, и в четвертой колонке сможем записать $Q = 1$.

Отсюда следует очень важное и часто используемое на практике правило булевой алгебры: «Сложное высказывание, которое состоит из двух простейших высказываний, связанных между собой союзом И, будет истинно (равно 1) тогда и только тогда, когда оба простейшие высказывания истинны, и ложно (равно 0), тогда, когда хоть одно из простейших высказываний ложно».

Математически в соответствии с данными табл. 2.2. это правило записывают так:

$$1 \text{ И } 0 = 0$$
$$0 \text{ И } 1 = 0$$
$$0 \text{ И } 0 = 0$$
$$1 \text{ И } 1 = 1$$

Если вместо символа И поставить знак умножения (в виде точки), то получим такую таблицу:

$$1 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$0 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

Сравнивая эту таблицу с таблицей умножения двоичных чисел (см. с. 22), можно обнаружить их абсолютное сходство. В связи с этим логическую операцию определения меры истинности сложного высказывания в зависимости от значения истинности составляющих высказываний, связанных союзом И, называют логическим умножением, а табл. 2.2 — таблицей истинности логического умножения. Часто эту таблицу записывают так, как это сделано в табл. 2.3.

Таблица 2.3

Таблица истинности логического умножения		
A	B	$A \cdot B$
1	0	0
0	1	0
0	0	0
1	1	1

Учитывая разнообразие терминов и обозначений, встречающихся в литературе по микропроцессорной технике, укажем, что вместо символа «И» иногда пишут знак & или знак $\wedge$, а логическое умножение называют конъюнкцией.

Прежде чем перейти к рассмотрению логических высказываний, связанных союзом ИЛИ, заметим, что в русском языке этот союз имеет двоякий смысл: исключающий (либо—либо) и неисключающий.

Приведем пример высказывания с исключающим ИЛИ: «Ты поедешь в Москву ИЛИ поедешь в Ленинград». Здесь поездка в Москву исключает возможность одновременной поездки в Ленинград. Можно поехать либо в Москву, либо в Ленинград.

А теперь пример неисключающего ИЛИ: «К нам в гости приедет Иванов ИЛИ приедет Петров». Тут не исключается одновременность двух событий — приезд Иванова или Петрова. В гости может приехать один Иванов или один Петров, или они оба.

В алгебре логики чаще всего пользуются понятием НЕИСКЛЮЧАЮЩЕГО ИЛИ. Помня о таком значении союза ИЛИ, перейдем к рассмотрению истинности сложного высказывания в зависимости от истинности простейших высказываний, связанных этим союзом. Анализ будем выполнять в том же порядке, как мы это делали, рассматривая логические высказывания, связанные союзом И.

Для анализа воспользуемся таким предложением: «Ночью выпадут осадки в виде падающего дождя ИЛИ падающего снега».

Это сложное высказывание состоит из двух простейших высказываний. Первое простейшее высказывание о дожде условно обозначим буквой A и заметим, что $A = 1$, когда дождь идет, и $A = 0$, если дождя нет. Второе простейшее высказывание о снеге обозначим буквой B и положим, что $B = 1$, когда снег падает, и $B = 0$, если снега нет.

Сложное высказывание в виде прогноза осадков обозначим буквой Q и условимся, что $Q = 1$, когда осадки выпадают, и $Q = 0$, если осадков нет.

Формально, исследуемое логическое высказывание можно записать в таком виде:

$$A \text{ ИЛИ } B = Q,$$

где A — первое простейшее высказывание, ИЛИ — связывающее слово, B — второе простейшее высказывание, Q — сложное высказывание.

Для выяснения значения истинности Q в зависимости от значения истинности A и B составим табл. 2.4, аналогичную таблице 2.2.

Таблица 2.4

Вариант	Значение истинности		
	A	B	A ИЛИ $B = Q$
1	1	0	1
2	0	1	1
3	1	1	1
4	0	0	0

Рассмотрим вариант 1, когда $A = 1$ и $B = 0$ — идет дождь, но нет снега. В этом случае осадки будут иметь место, поэтому $Q = 1$.

При варианте 2, когда $A = 0$ и $B = 1$ — нет дождя, но падает снег, осадки все же имеют место и поэтому, как и раньше, $Q = 1$.

При варианте 3, когда $A = 1$ и $B = 1$ — идет дождь и падает снег, осадки, конечно, выпадают и поэтому $Q = 1$.

Только при варианте 4, когда $A = 0$ и $B = 0$ — нет дождя или снега, то нет и осадков. В этом случае $Q = 0$.

Отсюда следует такое правило булевой алгебры:

Сложное высказывание, которое состоит из двух простейших высказываний, связанных между собой союзом ИЛИ, будет истинно (равно 1), если истинно хотя бы одно из простейших высказываний или истинны они оба, и ложно (равно 0), тогда и только тогда, когда будут ложны оба простейшие высказывания

Математически это правило записывают так:

$$1 \text{ ИЛИ } 0 = 1$$
$$0 \text{ ИЛИ } 1 = 1$$
$$1 \text{ ИЛИ } 1 = 1$$
$$0 \text{ ИЛИ } 0 = 0$$

Если вместо символа ИЛИ поставить знак сложения (+), то получим такую таблицу:

$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$(1)^* + (1) = 1$$
$$0 + 0 = 0$$

Соответствующая таблица истинности — табл. 2.5.

Таблица 2.5

Таблица истинности логического сложения		
A	*B*	*A + B*
1 0 1 0	0 1 1 0	1 1 1 0

В литературе вместо символа ИЛИ иногда пишут знак $\vee$, а логическое сложение называют дизъюнкцией.

Помимо логической операции НЕИСКЛЮЧАЮЩЕЕ ИЛИ, в микропроцессорной технике иногда используют операцию ИСКЛЮЧАЮЩЕЕ ИЛИ. Не рассматривая ее подробно, отметим, что в этом случае логического сложения единица образуется тогда и только тогда, когда складываемые числа имеют различное значение истинности (когда у одного числа — 0, а у другого 1, или наоборот — у первого 1, а у второго 0). Если числа имеют одинаковые значения истинности (оба равны 0 или оба равны 1, то результат сложения будет нулевым).

Таблицу истинности логического сложения для операции ИСКЛЮЧАЮЩЕЕ ИЛИ записывают так, как это сделано в табл. 2.6.

Таблица 2.6

Таблица истинности логического сложения ИСКЛЮЧАЮЩЕЕ ИЛИ		
A	B	A ⊕ B
1 0 1 0	0 1 1 0	1 1 0 0

Логическое действие ИСКЛЮЧАЮЩЕЕ ИЛИ обозначают знаком $\oplus$

Теперь рассмотрим логическую связку НЕ. В алгебре логики вводится действие, которого нет в обычной алгебре. Это — от-

* Скобки в третьей строке поставлены для того, чтобы подчеркнуть условность (необычность) этого сложения.

рицание данного высказывания. Для каждого высказывания существует его отрицание в виде добавления слова НЕ; обозначается отрицание с помощью черты над отрицаемым словом (символом или фразой).

Если высказывание истинно, ($A = 1$), то его отрицание ложно ($A = 0$), и наоборот, если $A = 0$, то $A = 1$. Например, высказывание «Снег холодный» можно записать как $A = 1$, а отрицание этого высказывания, то есть высказывание «Снег НЕ холодный, снег теплый», можно записать: $A = 0$.

Аналогично, высказывание «Ночью светит солнце», обозначенное как $B = 0$, можно при его отрицании («Ночью НЕ светит солнце») записать: $B = 1$.

Эти правила булевой алгебры, сведенные в таблицу истинности логического отрицания, имеют вид (табл. 2.7)

Таблица 2.7

Таблица истинности логического отрицания	
A	$\overline{A}$
1 0	0 1

Чтобы лучше усвоить правила логических действий, рассмотрим несколько примеров их физического истолкования.

Представим себе простейший душ, состоящий из бака (рис. 7, *а* вклейки) и трубы с сеткой на конце. В отличие от традиционного душа установим на трубе не один, а два крана A и B и выясним, как будет зависеть поток воды от состояния этих кранов. Вполне очевидно, что вода по трубе будет протекать только тогда, когда будет открыт кран A И кран B. Если любой из кранов (или оба) будет закрыт, то вода течь не будет.

Условимся считать простейшее высказывание «Кран открыт» — истинным (равным 1), а простейшее высказывание «Кран закрыт» — ложным (равным 0). Положим также, что сложное высказывание «Вода течет по трубе» — истинное (равное 1), а «Вода не течет по трубе» — ложное (равное 0). Тогда, пользуясь правилами булевой алгебры, можно составить таблицу истинности (табл. 2.8), для нашего случая, соответствующую правилу логического умножения.

Если изменить конструкцию душа и расположить краны A и B так, как это показано на рис. 7, *б* вклейки, то пользоваться душем в этом случае можно тогда, когда открыт кран A ИЛИ кран B (ИЛИ оба крана). Нельзя будет пользоваться душем только тогда, когда будут закрыты оба крана и вода по трубам течь не будет. Таблица истинности для этого случая соответствует правилу логического сложения (табл. 2.9)

Таблица 2.8

Таблица истинности для душевой системы рис. 7, *а* вклейки		
Состояние крана		Поток воды $Q = A \cdot B$
A	B	
1	0	0
0	1	0
0	0	0
1	1	1

Таблица 2.9

Таблица истинности для душевой системы рис. 7, *б* вклейки		
Состояние крана		Поток воды $Q = A + B$
A	B	
1	0	1
0	1	1
1	1	1
0	0	0

Если конструкцию крана выполнить с левой резьбой, при которой завинчивание крана по часовой стрелке (рис. 7, *в* вклейки) будет открывать задвижку в кране и разрешать течение воды, а отвинчивание (вращение против часовой стрелки) — запрещать поток воды, то мы получим физический пример логического отрицания НЕ.

Еще нагляднее можно представить логическое умножение, сложение и отрицание на примере электрической цепи.

Составим электрическую цепь из батарейки Б (рис. 8, *а* вклейки) электрической лампочки *HL* и двух выключателей *A* и *B*, соединенных последовательно. Выясним, как будет зависеть свечение лампочки *HL* от состояния выключателей *A* и *B*. Аналогично потоку воды в душе, электрический ток в цепи будет протекать через лампочку *HL* только тогда, когда контакты выключателей *A* и *B* будут замкнуты. Если контакты какого-либо выключателя (обоих выключателей) перевести в разомкнутое состояние, то электрическая цепь прервется, ток в цепи прекратится и лампочка *HL* погаснет. Будем полагать: простейшее высказывание «Контакты выключателя замкнуты» — истинным (равным 1), а простейшее высказывание «Контакты выключателя разомкнуты» — ложным (равным 0), а сложные высказывания: «Ток протекает в цепи, лампочка *HL* горит» — истинным (равным 1) и «Ток не протекает в цепи, лампочка *HL* не горит» — ложным (равным 0). Такое обозначение высказываний позволяет составить таблицу истинности логического умножения, соответствующую табл. 2.3 (с. 33).

Если видоизменить электрическую схему и соединить батарейку Б, лампочку *HL* и выключатели *A* и *B* так, как это показано на рис. 8, *б* вклейки, то в этом случае лампочка *HL* будет гореть, если замкнуты контакты выключателя *A*, ИЛИ контакты выключателя *B*, ИЛИ контакты обоих выключателей. Только в том случае, когда будут разомкнуты контакты обоих выключателей, ток в цепи прекратится и лампочка *HL* погаснет. Таблица истинности для этого случая будет соответствовать логическому сложению (см. табл. 2.5). Операцию отрицания НЕ можно продемонстрировать с помощью электрической цепи, снабженной электромагнитным реле Р (рис. 8, *в* вклейки). При замыкании контактов выключателя *A* включается реле Р, его контакты Р.1 разрывают цепь питания лампочки *HL* и она гаснет.

Таким образом, замыкание контактов выключателя вызывает не зажигание лампочки, а ее гашение, то есть операцию отрицания (НЕзажигание).

Во всех предыдущих примерах говорилось о двух простейших высказываниях, входящих в состав сложного высказывания (две простейшие фразы в сложном предложении, два крана в конструкции душа, два выключателя в электрической цепи). В действительности сложное высказывание может состоять не из двух, а из большего числа простейших высказываний. Два простейших высказывания — это частный случай сложного высказывания. В общем же случае правила булевой алгебры для логического умножения и сложения могут быть представлены так:

Сложное высказывание, которое состоит из многих простейших высказываний, связанных между собой союзом И, будет истинно (равно 1) тогда и только тогда, когда все простейшие высказывания истинны, и ложно (равно 0), если хоть одно из простейших высказываний ложно.

Сложное высказывание, которое состоит из многих простейших высказываний, связанных между собой союзом ИЛИ, будет истинно (равно 1), если истинно хотя бы одно из простейших высказываний или истинны они все, и ложно (равно 0) тогда и только тогда, когда будут ложны все простейшие высказывания.

С увеличением числа простейших высказываний, входящих в состав сложного высказывания, таблицы истинности получаются более громоздкими. Для примера в табл. 2.10 записана таблица истинности логического умножения для случая включения в электрическую цепь трех выключателей *A*, *B* и *C* (выключатель *C* показан на рис. 8, *а* вклейки пунктиром). В этом случае таблица истинности имеет восемь строк, вместо четырех, которые были в предыдущих таблицах (см. табл. 2.3). При четырех простейших высказываниях число строк в таблице истинности увеличивается до шестнадцати и т. д.

В микропроцессорной технике находят широкое применение электронные логические элементы типа И, ИЛИ, НЕ, у которых вместо механических выключателей *A* и *B*, показанных на рис. 8

Таблица 2.10

Таблица истинности логического умножения при трех выключателях A, B и C в электрической цепи (рис. 8 вклейки)

A	B	C	$Q = A \cdot B \cdot C$
0	0	1	0
1	0	1	0
1	1	0	0
0	1	1	0
1	0	0	0
0	1	0	0
0	0	0	0
1	1	1	1

вклейки, установлены транзисторные выключатели, называемые вентилями или транзисторными ключами.

Упрощенные схемы таких логических элементов приведены на рис. 9 вклейки. Если на входы 1 (рис. 9, *а* вклейки) и 2 подать напряжение высокого логического уровня (1), то транзисторы VTA и VTB откроются и напряжение +5 В через открытые транзисторы будет поступать на выход в виде напряжения высокого логического уровня (1). Когда на один из входов (или на оба входа) подать напряжение низкого логического уровня (0), то транзистор закроется и на выходе образуется напряжение низкого логического уровня (0). Такая схема называется логической схемой 2И (двухвходовая логическая схема И).

Подключение напряжения высокого логического уровня (1) к любому из входов (входы 1 или 2 на рис. 9, *б* вклейки) приводит к открытию одного из транзисторов и появлению на выходе напряжения высокого логического уровня (1). Только тогда, когда на оба входа (входы 1 и 2) будет подано напряжение низкого логического уровня (0), оба транзистора оказываются закрытыми и на выходе будет иметь место низкий логический уровень (0). Такая схема получила название логической схемы 2ИЛИ (двухвходовая логическая схема ИЛИ).

Логическую схему И часто называют схемой совпадения, благодаря тому, что на выходе схемы единичный уровень (1) появляется тогда, когда совпадают по времени действия единичные уровни (1), подаваемые на входы.

Логическую схему ИЛИ называют собирательной или суммирующей схемой, так как она собирает (суммирует) входные единичные уровни (1) и выдает их в виде единичного уровня (1) на выходе.

Схема, представленная на рис. 9, *в* вклейки выполняет действие операции НЕ. Если на вход подать напряжение низкого логического уровня (0), то транзистор VT закрыт и на выход через резистор R поступает напряжение высокого логического уровня (1). Когда на вход подают напряжение высокого логического уровня (1), то транзистор открывается, его сопротивление

становится предельно малым и поэтому на выходе образуется напряжение низкого логического уровня (0).

Промышленность выпускает сотни различных типов электронных логических элементов в интегральном исполнении, представляющих собой сочетание элементов И, ИЛИ, НЕ.

В виде примера на рис. 10 вклейки приведена схема и расположение выводов одной из самых распространенных логических микросхем типа К155ЛА3, представляющих собой сочетание в одном корпусе четырех двухвходовых схем 2И—НЕ. Каждая логическая схема 2И имеет два входа (выводы: 1 и 2, 4 и 5, 9 и 10, 12 и 13) и один выход (выводы: 3, 6, 8 и 11). Кружком на выходах обозначено инверсное действие НЕ, в результате которого на выходе образуется уровень логического нуля, когда на оба входа поданы уровни логических единиц. Напряжение уровня логической единицы для К155ЛА3 составляет +2,4 В, а напряжение уровня логического нуля +0,4 В.

Для читателей, желающих расширить свои знания в области алгебры логики и работы логических микросхем, рекомендуем литературу [2, 3, 4, 5, 27, 36, 50, 63].

Для закрепления знаний рассмотрим пример использования логических элементов И, ИЛИ для решения практической задачи.

Представим себе электрифицированную игру в «крестики—нолики», для которой изготовлено табло, имеющее по 9 лампочек, подсвечивающих крестики или нулики в квадратах, а также тумблеры, с помощью которых игроки могут включать выбранную ими лампочку.

Ставится задача: с помощью логических микросхем И, ИЛИ собрать устройство, которое автоматически сигнализировало бы о выигрыше.

Как известно, в этой игре выигрывает тот, кто первым зажжет все три лампочки в какой-либо строке квадрата, например лампочку *1* (рис. 11 вклейки), лампочку *2* и лампочку *3* в верхней горизонтальной строке. Поэтому, подключив к лампочкам каждой строки трехвходовые логические элементы 3И, то есть подав на каждый вход этих элементов уровень напряжений логической единицы от светящейся лампочки, можно получить на выходе каждого элемента 3И уровень логической единицы, сигнализирующий о выигрыше. Так как строк в игре восемь, то необходимо подключить восемь микросхем типа 3И. Для того чтобы сигнализация о выигрыше срабатывала от любой из этих восьми микросхем 3И (от первой, второй, третьей или четвертой и т. д., вплоть до восьмой), нужно к выходу каждой микросхемы 3И подключить вход восьмивходовой логической микросхемы 8ИЛИ. Тогда уровень логической единицы, поступающей на любой из восьми входов элемента 8ИЛИ, приведет к появлению уровня логической единицы на его выходе и напряжение этого уровня включит сигнальное устройство о выигрыше.

Как видно из рассмотренного примера, в логических задачах исходными данными и результатом являются чаще всего не числа,

а некоторые, порой весьма запутанные, условия и суждения (ситуации). Эти суждения и связи между ними и являются предметом решения с помощью логических элементов типа И—ИЛИ—НЕ.

Чтобы разобраться в работе микропроцессора, необходимо совершенно четко и ясно представлять себе действие логических элементов И, ИЛИ, НЕ.

Для читателей, которые не поняли работу логических элементов при помощи таблиц истинности, советуем рассмотреть рис. 12—14 вклейки, где упрощенно изображены действия логических элементов в виде домиков, во входные двери которых входят, а из выходных — выходят человечки высокого роста — красные единички и низкого роста — зеленые нулики. Напомним, что единички представляют собой высокий уровень напряжения, а нулики — низкий, нулевой уровень напряжения.

Когда в домик типа НЕ (рис. 12 вклейки) входят единички, то они выходят оттуда в виде нуликов. И, наоборот, нулики, которые приходят в домик, покидают его единичками (рис. 12, *б* вклейки).

Если во входную дверь домика НЕ (рис. 12 *в* вклейки) приходит цепочка, состоящая из одной единички и двух нуликов, то на выходе образуется цепочка в виде одного нулика и двух единичек.

На рис. 13 вклейки показано действие домика 2И. Когда на входы A и B (рис. 13, *г* вклейки) поступают единички, то из выходной двери $A \cdot B$ тоже выходят единички. Во всех других случаях (рис. 13, *а, б, в* вклейки) на выходе образуются только нулики.

Рис. 14 вклейки поясняет действие элемента 2 ИЛИ.

На рис. 12—14 вклейки рядом с изображением домиков НЕ, 2И и 2ИЛИ представлены таблицы истинности, математически рассказывающие о том, кто появляется на выходе домика, когда на входы A и B приходят единички или нулики. Внизу каждой таблицы истинности изображен логический элемент в том виде, в каком его применяют для черчения принципиальных электрических схем электронных устройств.

Помимо простейших логических элементов И—ИЛИ—НЕ в виде однокомнатных домиков, изображенных на рис. 12—14 вклейки, промышленность выпускает более сложные, комбинированные логические элементы, состоящие как бы из многокомнатных квартир. На рис. 15 вклейки изображено устройство и принцип действия логического элемента — домика типа четыре 2И—НЕ, электрическая схема которого была приведена на рис. 10 вклейки (логическая микросхема типа К155ЛА3).

В заключение отметим, что микропроцессор представляет собой такой же логический домик, но только домик-гигант, состоящий из миллионов многокомнатных квартир типа И, ИЛИ, НЕ и особых помещений типа транзистор. При этом открывание и закрывание огромного количества дверей этого грандиозного сооружения выполняет специальное устройство управления, создавая этим не-

обходимые условия для перемещения, хранения или преобразования бесчисленного количества единичек и нуликов, переходящих из комнаты в комнату в виде упорядоченных потоков и движущихся цепочек.

Далее мы подробнее рассмотрим работу устройства управления и других элементов микропроцессора, а сейчас еще раз напомним читателям о том, что перед изучением следующего раздела (2.2) настоящей главы необходимо совершенно ясно представлять себе принцип действия логических элеменов И, ИЛИ, НЕ. Без такого представления понять работу микропроцессора будет невозможно.

Искусственные языки

Всем хорошо известно, что лучшим средством выражения мыслей и способом описания всего того, что происходит в мире, является наш разговорный язык. Однако существует ряд понятий и предметов, рассказать о которых более удобно на специальном, искусственно придуманном языке. Примерами такого языка являются: нотный язык музыки, язык расположения шахматных фигур, телеграфный язык азбуки Морзе, язык химических формул и др. Применение нужного языка зависит от того, что предстоит описать — простую житейскую ситуацию, устройство электронного аппарата или состав вещества. В первом случае удобен обычный разговорный язык, во втором — язык электрических схем, в третьем — язык химических формул.

Чтобы ознакомиться с работой микропроцессора, собрать микроЭВМ и научиться писать для нее программы, необходимо освоить несколько искусственных языков. Прежде всего язык схем (структурных, принципиальных и монтажных), с помощью которых описывают электрические цепи, соединяющие электронные элементы. Затем — язык временных диаграмм, наглядно отображающих изменение событий с течением времени, и, наконец, язык программирования, на котором пишут программы для микроЭВМ.

Поскольку радиолюбители знакомы в общих чертах со схемами и временными диаграммами, то на этих языках останавливаться не будем. Читателям, которые не знают, как рисовать и читать радиоэлектронные схемы и временные диаграммы, рекомендуем ознакомиться с ними по литературе [5, 27, 50, 57, 63]. Что же касается языка программирования, то ему посвящен раздел 2 Приложения (с. 172).

Термины

Тому, кто впервые берется за работу с микропроцессорной техникой, прежде всего следует ознакомиться со специальными терминами и сокращениями слов, используемых в этой области.

В противном случае чтение литературы по микропроцессорам и микроЭВМ потребует очень больших усилий и времени. На каждой странице могут встретиться непреодолимые трудности, а порой могут возникнуть совершенно неправильные представления о принципе работы и процессах, происходящих в отдельных устройствах этой сложной техники. Трудности будут усугубляться и тем обстоятельством, что в сфере микропроцессорной техники используется много жаргонных терминов с довольно расплывчатой формулировкой, а в специальной технической литературе существует немалая терминологическая путаница.

Неустоявшаяся терминология — явление, характерное для любой новой, быстро развивающейся отрасли техники. Этой участи не избежала и микропроцессорная техника. В ряде книг, у разных авторов, а порой даже у одного и того же автора, можно встретить совершенно разные термины, означающие одно и то же. Не всякий, например, поймет, что «эмулятор», «имитатор» и «стимулятор» — это названия одной и той же служебной программы микроЭВМ, а термины «тетрада», «нибла» и «полубайт» — означают четырехразрядное двоичное число.

Такой разнобой и неупорядоченность терминов, а также значительное их количество (более ста) создают известные трудности для неспециалиста в области вычислительной техники, желающего с минимальной затратой сил и времени ликвидировать «белое» пятно в своих знаниях и получить хотя бы первоначальные самые общие понятия о существе дела и возможностях практического использования этой техники.

Для ознакомления с терминами целесообразно пользоваться также энциклопедическим словарем юного техника [65].

Черный ящик

Черный ящик — кибернетический термин. Когда кибернетики не хотят или не могут рассказать о работе очень сложного устройства, то они употребляют понятие черный ящик, поясняя при этом только действия, происходящие на его входе и выходе, и умалчивая о внутреннем устройстве и принципе работы устройства.

Понятие черного ящика кибернетики заимствовали от иллюзионистов, которые выносят на сцену волшебный ящик черного цвета, закладывают в его входное отверстие какой-либо предмет (например, часы), а на выходе получают другой предмет (например, букет цветов). При этом зрителю совершенно неясно, что находится внутри черного ящика и как происходит волшебное превращение одного предмета в другой. Однако, если зритель посетит несколько представлений иллюзиониста, превращающего с помощью черного ящика часы в букет цветов, то при каждом последующем посещении зритель уже сможет предсказать поведение черного ящика, то есть угадать, какой предмет появится на его выходе, если такой-то предмет заложить в его вход. Вот этой особенностью черного ящика и пользуются кибернетики, физики и специалисты

других отраслей знаний. Используют ее и при изучении микропроцессорной техники.

Взять, например, логическую микросхему типа К155ЛА3 (четыре 2И—НЕ, рис. 10 вклейки). Достаточно провести с ней несколько экспериментов, чтобы убедиться в том, что уровень логического напряжения на выходах 3, 6, 8 и 11 будет равен нулю, если на входы 1, 2 или 4, 5, а также 9, 10 или 12, 13 подавать напряжение уровня логической единицы. А какое устройство имеет схема К155ЛА3, из каких элементов она состоит и как эти элементы соединены — нам часто бывает безразлично.

При рассмотрении действий логических элементов в виде домиков (рис. 12—14 вклейки) можно, например, также пренебречь вопросом о том, как единички превращаются в нулики при входе в домик типа НЕ.

Нам это неважно. Нам нужно только знать результат, следствие, закон преобразования элемента НЕ, а причины, по которым единички превращаются в нулики, должны больше интересовать того, кто конструирует элемент НЕ.

На практике большинство интегральных микросхем вычислительной техники, как и сам микропроцессор, можно рассматривать как черный ящик, позволяющий предсказывать, что будет на его выходе, если совершать то или иное действие на входе, пренебрегая при этом знанием очень сложного внутреннего устройства.

2.2. МИКРОПРОЦЕССОР — АВТОМАТ ПО ОБРАБОТКЕ ДАННЫХ

Чтобы легче было разобраться в работе микропроцессора, выполняющего роль программного автомата по обработке **данных,** рассмотрим торговый автомат. Опуская монету нужного достоинства во входное отверстие такого автомата, на его выходе можно получить стакан газированной воды, пачку сигарет или проездные билеты. Конструкция автомата бывает довольно сложной. Но тот, кто пользуется автоматом, может ничего о ней не знать, считая автомат черным ящиком и имея представление лишь о том, где находится вход для монеты и где расположен выход товара.

Упрощенная схема подобного автомата приведена на рис. 16 вклейки. Здесь прямоугольником обозначен черный ящик торгового автомата. Слева от него стрелкой А указан вход монеты, а справа стрелкой Б — выход товара.

Более сложные автоматы, помимо входа А и выхода Б, имеют еще кнопки для выбора вида товара. Примером может служить автомат, отпускающий газированную воду. Три кнопки В1, В2 и В3 на рис. 17 вклейки такого автомата позволяют получить газированную воду без сиропа, с лимонным или апельсиновым сиропом. Назовем эти кнопки программными и объясним, что при нажатии кнопки В1 автомат срабатывает по программе, обеспечивающей выдачу газированной воды без сиропа; при нажатии кнопки В2 работа автомата происходит по другой программе, добавляющей в воду лимонный сироп, а нажатие кнопки В3 включает третью

программу, в результате которой стакан наполняется водой с апельсиновым сиропом. Аналогично можно представить себе автомат, выдающий разные газеты или почтовые открытки при нажатии соответствующей программной кнопки.

Микропроцессор является таким же автоматом. Только, в отличие от торгового автомата, он не выдает товар, а производит обработку **информации.** По аналогии со схемой, изображенной на рис. 17, на рис. 18 вклейки приведена упрощенная схема микропроцессора. На этой схеме левая стрелка А изображает вход информации в виде цифровых данных, поступающих для обработки, а правая стрелка Б указывает, откуда выходят обработанные данные. Обработка выполняется по программе, подаваемой на вход В.

Предположим, что на вход А поступают цифровые данные в виде двух чисел 3 и 2, обведенных на рис. 18 вклейки прямоугольниками, а на программный вход В подается команда «Сложить» Тогда на выходе Б появится число 5($3 + 2 = 5$) Если бы на программный вход В была подана команда «Вычесть», то на выходе Б появилось бы число 1 ($3 - 2 = 1$).

Таких элементарных понятий о микропроцессоре, подобно начальным сведениям о торговом автомате, было бы достаточно, чтобы с помощью простейшего устройства ввода данных подать на вход А исходные цифровые величины, на вход В — команду выполняемых действий и на выходе Б получить искомый результат. Однако, идя навстречу читателям, которые сами захотят собрать микроЭВМ, написать и отладить программу ее работы, приоткроем черный ящик микропроцессора и по реальной схеме опишем, из каких частей он состоит и как они взаимодействуют при обработке данных.

Прежде всего укажем, что любая информация (цифровые данные, буквенный текст, графическое или телевизионное изображение, различные виды механических действий, физиологических состояний и др.), подлежащая вводу в микропроцессор для обработки предварительно **кодируется**, то есть условно представляется в виде двоичных чисел, называемых цифровым двоичным кодом, а сами двоичные числа вводятся в микропроцессор путем подключения к его электрическим выводам электрического напряжения различного уровня (высокого или низкого), представляющего собой единицы (1) или нули (0) двоичного числа.

В табл. 2.11 приведен пример кодирования различных данных [1] с помощью восьмиразрядного двоичного кода.

---

[1] В литературе по микропроцессорной технике вместо термина **информация** часто употребляется слово **данные**. Пишут: «Переслать данные по шине данных в микропроцессор», «Запомнить данные в ОЗУ», «Вывести данные на экран дисплея» и др. Поэтому во всех последующих главах книги мы чаще всего будем пользоваться этим словом, понимая под данными цифровую, текстовую графическую и другие виды информации, поступающей в микропроцессор по шине данных в виде цифрового двоичного кода для обработки.

Таблица 2.11

Данные	Двоичный код данных
Десятичное число «5»	00000101
Буква «А»	00100001
Слово «начало»	00110010
Действие «включено»	00100000
Действие «выключено»	00010000

В зависимости от объема данных, подлежащего кодированию, применяют двоичные числа различной длины, имеющие два, три, четыре и больше разрядов.

Соответственно и микропроцессоры, служащие для обработки двоичного кода данных, выпускают четырех-, восьми-, шестнадцати- или тридцатидвухразрядными.

Рассмотрим устройство четырехразрядного микропроцессора. На рис. 19 вклейки приведена его схема. На этой схеме слева указаны четыре вывода (А0, А1, А2 и А3) для включения электрических уровней (1 или 0) вводимого цифрового двоичного кода данных. Справа изображены четыре вывода (Б0, Б1, Б2 и Б3) для выдачи двоичного кода обработанных данных. Снизу находятся четыре вывода (В0, В1, В2 и В3) для подключения двоичного кода операции по обработке данных.

Теперь кратко ознакомимся с внутренним устройством микропроцессора. Как уже говорилось, микропроцессор состоит из десятков тысяч и даже миллионов электронных элементов, соединенных между собой и образующих сотни взаимосвязанных узлов, блоков и устройств, выполняющих различные функциональные действия.

Среди такого множества сложных частей все же удается выделить несколько главных, основных. К ним относятся: **регистры, арифметическо-логическое устройство** (сокращенно **АЛУ**) и **устройство управления (УУ)**, изображенные на рис. 19 вклейки. Регистр служит для временного хранения двоичного числа. Он состоит из триггеров, общее число которых соответствует количеству разрядов регистра. В рассматриваемом четырехразрядном микропроцессоре регистр содержит четыре триггера.

Разные типы микропроцессоров имеют в своем составе разное количество регистров (от 4 до 32). Чтобы отличить один регистр от другого, их обозначают латинскими буквами. На рис. 19 вклейки изображено четыре регистра *A*, *B*, *C* и *D*

Арифметическо-логическое устройство (АЛУ) осуществляет операции по арифметической и логической обработке данных, а именно выполняет действия сложения, вычитания, сравнения, сдвига, преобразования, увеличения или уменьшения числа на единицу и др.

Устройство управления (УУ) является самым сложным из всех устройств микропроцессора. Используя поток электрических сиг-

налов, поступающих в виде единиц и нулей на входы данных (А0—А3), входы кода операции (В0—В3) и другие входы микропроцессора (не показанные на рис. 19 вклейки), устройство управления вырабатывает серию последовательных управляющих электрических сигналов, передаваемых в нужные узлы, блоки и элементы микропроцессора для координации их работы, а также для согласования действий микропроцессора с внешними устройствами, например памятью. Все, что происходит в микропроцессоре, подчинено электрическим сигналам устройства управления. Образно УУ можно сравнить с дирижером большого оркестра, состоящего из сотен различных электронных узлов и тысячи логических элементов, входящих в состав микропроцессора. При этом партитурой музыкального произведения служит программа, заложенная в память.

Каждый тип микропроцессора имеет свой двоичный код операций, подаваемый на входы В0—В3 для выполнения нужных действий. Список кодов этих операций называют системой команд.

Двоичный код некоторых операций, взятых из системы команд нашего четырехразрядного микропроцессора, приведен в табл 2.12 В левой колонке таблицы описаны действия выполняемой операции, а в правых колонках указано, какие уровни напряжений (1 или 0) необходимо подавать на входы В0, В1, В2 и В3, чтобы выполнить эти действия.

Рассмотрим сложение двух чисел (3 и 2) при помощи микропроцессора. На рис. 24—27 вклейки последовательно изображены все операции, которые он выполняет, складывая числа.

Поясним эти операции. На рис. 20 вклейки показано, как с помощью простейшего устройства ввода двоичных чисел, описанного в разделе 2.1 и состоящего из четырех переключателей (П0—

Таблица 2 12

Коды некоторых операций системы команд четырехразрядного микропроцессора				
Название операции	Двоичный код операции			
	В3	В2	В1	В0
Запомнить данные в регистре А	0	0	0	1
Запомнить данные в регистре В	1	0	0	0
Сложить данные регистров В и А. Результат поместить в регистре А	1	1	0	0
Вычесть из регистра А данные регистра В. Результат поместить в регистр А	1	0	1	0
Переслать данные из регистра А на выход Б	0	1	1	0

П3), можно подать на выводы А0—А3 двоичный код десятичного числа 3. Для этого нужно переключатели П0 и П1 установить в верхнее положение, обеспечивающее выдачу 1, а переключатели П2 и П3 — в нижнее положение, при котором выдается 0. В этом случае на входы А0—А3 будет подано четырехразрядное двоичное число 0011, означающее десятичную тройку. Теперь нужно запомнить поданное число в каком-либо регистре микропроцессора, например в регистре А. Для этого следует подать на входы В0—В3 с помощью таких же переключателей (рис. 21 вклейки), согласно табл. 2.12, двоичный код 0001. Этот код поступит в устройство управления, расшифруется там (определится смысл операции) и преобразуется в сигналы, поступающие в регистр А в виде указания о необходимости подключиться к выводам А0—А3 и запомнить имеющиеся там логические уровни (1 и 0), а после запоминания отключиться от выводов А0—А3. Таким образом, в результате произведенных операций одно из слагаемых чисел (тройка) поступит в микропроцессор и запомнится в регистре А. Аналогично (рис. 22 вклейки), с помощью того же устройства ввода нужно набрать и ввести код второго слагаемого (двойки). Для этого переключатель П1 следует установить в верхнее положение, а все остальные — в нижнее. Тогда на входе А1 появится 1, означающая десятичную двойку. Если теперь на входы В0—В3 (рис. 23 вклейки) подать двоичный код 1000, означающий согласно табл. 2.12. операцию запоминания данных в регистре В, то двоичное число (0010), находящееся на выводах А0—А3, поступит в регистр В и запомнится там.

Для выполнения операции сложения чисел, расположенных в регистрах А и В (рис. 24 вклейки), следует подать на выводы В0—В3 код 1100, взятый из табл. 2.12. При этом числа из регистров А и В переместятся в АЛУ и там произойдет их сложение, после чего устройство управления (УУ) подаст в АЛУ и в регистр А управляющие сигналы для пересылки результата сложения в регистр А. Таким образом, операция сложения будет состоять из трех действий или трех тактов: первый такт — пересылка данных из регистров А и В в АЛУ; второй — сложение данных в АЛУ и третий — пересылка результата сложения в регистр А. Некоторые, более сложные операции (не указанные в табл. 2.12), микропроцессор выполняет не за три, а за десять и более тактов своей работы.

Чтобы закончить решение поставленной задачи по сложению двух чисел, нужно еще подать на выводы В0—В3, согласно табл. 2.12, код 0110 для выполнения операции по пересылке данных из регистра А на выводы Б0—Б3 (рис. 26 вклейки). Если на это время к выводам Б0—Б3 подключить светодиоды С0—С3, то по свечению светодиодов С0 и С2 можно будет прочесть двоичную цифру 0101, обозначающую десятичную пятерку.

Все рассмотренные операции по сложению чисел микропроцессор выполняет не беспорядочно, а через строго определенные промежутки времени, повинуясь специальному электрическому сигналу (короткому импульсу), поступающему в устройство управ-

ления из тактового генератора. Этот короткий сигнал, подобно стартовому выстрелу, подаваемому на соревнованиях бегунов, заставляет устройство управления начать выполнение такта своей работы, заключающегося в создании определенного количества управляющих воздействий, которые зависят от кода операции. После исполнения этих воздействий работа микропроцессора приостанавливается и устройство управления переходит в состояние ожидания следующего тактового сигнала.

Частота тактовых сигналов, вырабатываемых тактовым генератором, бывает довольно высокой. Она исчисляется сотнями тысяч и даже миллионами тактовых сигналов в секунду.

Учитывая, что единицей измерения частоты служит 1 Герц (один тактовый сигнал, выдаваемый за одну секунду), укажем, что частота тактового генератора современных микропроцессоров составляет сотни килогерц, и даже несколько мегагерц. Конструктивно тактовый генератор чаще всего выполняется отдельно от микропроцессора и размещается рядом так, как это показано на рис. 27 вклейки. Однако встречаются микропроцессоры, у которых тактовый генератор входит в состав его внутренней схемы.

Мы коротко рассмотрели работу в некоторой степени иллюстративного четырехразрядного микропроцессора при сложении двух чисел. По своему действию такой микропроцессор близок к четырехразрядному микропроцессору типа К584ИК1, выпускаемому промышленностью.

Конечно, схема К584ИК1 выглядит более сложной — она содержит ряд дополнительных устройств, узлов и выводов, но принцип ввода и обработки данных у нее, примерно, такой же, как и у описанного выше микропроцессора.

Помимо микропроцессора типа К584ИК1, наша промышленность выпускает и другие типы, отличающиеся разрядностью, быстродействием, системой команд, технологией изготовления, конструктивными данными, стоимостью и др.

Каждый тип микропроцессора имеет свои достоинства и свои недостатки и применяется в тех или иных конкретных условиях, в зависимости от требований, предъявляемых к ним.

Структурная схема четырехразрядного микропроцессора типа К584ИК1 оказалась, например, очень удобной для начального знакомства с принципом ввода и вывода данных. Поэтому мы выбрали именно этот тип для иллюстрации рассказа об устройстве микропроцессора. На базе таких микропроцессоров, соединяемых в группы парами (для увеличения разрядности обрабатываемых чисел), строят микроЭВМ, решающие разнообразные задачи.

Однако для сборки простейшей микроЭВМ целесообразнее взять другой тип микропроцессора, а именно К580ИК80, обладающий рядом преимуществ. К ним относятся: доступность и сравнительно низкая стоимость, что очень важно при самодельном изготовлении микроЭВМ. Большим достоинством микропроцессора К580ИК80 является типичность структуры — то, что многие

его модификации имеют сходное внутреннее устройство. Изучив микропроцессор К580ИК80, можно без особых затруднений понять принцип работы и других типов. Существует также обилие разработанных и отлаженных программ для этого микропроцессора. Следует отметить, что он был выпущен в начале 70-х годов и по своим электрическим качествам во многом уступает современным микропроцессорам, но созданные за эти годы программы (игровые, расчетные, управляющие, служебные) можно с успехом использовать и сейчас, не теряя зря силы и время на их разработку. И наконец, очевидное превосходство заключается в популярности этого микропроцессора. Даже теперь некоторые заводы изготовляют микроЭВМ на его базе. Правда, теперь он выпускается под названием КР580ИК80А. Изменение приставки К на КР обусловлено иной конструкцией и технологией изготовления микропроцессора данной серии. По остальным показателям микропроцессор КР580ИК80А не отличается от микропроцессора К580ИК80.

Учитывая упомянутые преимущества КР580ИК80А, в дальнейшем будем изучать именно этот тип микропроцессора и примем его за основу для сборки микроЭВМ.

В связи с тем, что название «микропроцессор типа КР580ИК80А» будет часто повторяться в тексте нашего дальнейшего изложения, условимся называть его «МП—К580».

Рассмотрим его отличия от уже знакомого нам четырехразрядного микропроцессора. МП—К580 является восьмиразрядным микропроцессором, служащим для ввода, обработки и вывода восьмиразрядных двоичных чисел. Три группы выводов четырехразрядного микропроцессора (группа А — для ввода данных, группа Б — для вывода и группа В — для ввода кода операции, согласно рис. 19 вклейки) в МП—К580 объединены в одну общую группу, имеющую восемь выводов (D0—D7 на рис. 28 вклейки) К этим выводам подключаются восемь проводов (линий), называемых шиной данных. По шине данных в течение какого-то короткого промежутка времени осуществляется ввод данных, т. е. действия, которые в четырехразрядном микропроцессоре выполнялись по проводам группы А. По этой же шине данных в течение другого промежутка времени передается код операции, который раньше пересылался по проводам группы В и наконец, в какой-то промежуток времени шина данных выполняет то, что прежде выполнялось проводами группы Б — вывод данных из микропроцессора во внешнее устройство.

Такая замена трех групп выводов одной группой приводит к уменьшению числа выводов микропроцессора (8 вместо 24) Отметим также, что при вводе данные следуют по шине в одном направлении, а при выводе — в другом.

Следующей отличительной особенностью МП—К580 является наличие у него так называемых адресных выводов. Прежде чем рассказать о них, укажем, что вводить в микропроцессор код операции с помощью переключателей (см. рис. 20 и 22 вклейки)

можно лишь в условиях специально замедленного ритма работы микропроцессора, осуществляемого в учебных целях. При нормальной же работе микропроцессора код вводится и исполняется в течение нескольких микросекунд, поэтому вводить его вручную невозможно. В этом случае высокое быстродействие микропроцессора достигается тем, что все коды операций, а часто и данных, заранее помещают в запоминающее устройство (память), а затем микропроцессор с нужной ему скоростью последовательно выбирает из ячеек памяти двоичный код одной операции за другой и сейчас же их исполняет.

Чтобы микропроцессор мог из той или иной ячейки памяти выбирать код операции или данных, все ячейки нумеруют и этот номер называют **адресом** ячейки памяти. Адрес, в отличие от номера, выражается не десятичным числом, а его двоичным эквивалентом в виде нулей и единиц, появляющихся на 16 адресных выводах A0—A15 (см. рис. 28 вклейки). Таким образом, адресные выводы A0—A15 служат для размещения на них адреса той ячейки памяти, к которой желает обратиться микропроцессор. Адресные выводы A0—A15 с помощью адресной шины соединяются с адресными выводами устройства памяти.

МП—К580, как и четырехразрядный микропроцессор, состоит из регистров, арифметическо-логического устройства (АЛУ) и устройства управления (УУ). Его 10 регистров обозначены на рис. 28 вклейки буквами A, B, C, D, E, H, L, F, УС и СК. Регистры УС и СК шестнадцатиразрядные, а остальные — восьмиразрядные.

Регистр А, изображенный на рис. 28 вклейки отдельно от группы регистров, называется **аккумулятором**. Он является основным рабочим регистром, участвующим во многих операциях. Например, при сложении двух чисел одно из них обязательно должно находиться в регистре А. Результат вычислений также автоматически размещается в регистре А. Работа регистра А неразрывно связана с АЛУ, поэтому на схеме его часто изображают вблизи или внутри АЛУ.

Регистры A, B, C, D, E, H и L образуют группу так называемых **регистров общего назначения** (РОН). Характерной особенностью РОН является доступность каждого регистра для пользователя, выражаемая в том, что с помощью определенной команды можно поместить данные в любой регистр РОН или извлечь их из регистра, а также переслать из одного регистра в другой. В то же время МП—К580 содержит ряд регистров (F и некоторые другие, не указанные на рис. 28 вклейки), которыми распоряжается только устройство управления, и пользователь не может по своему усмотрению с помощью команд применить эти регистры для ввода и вывода данных.

Регистры РОН восьмиразрядные, но они допускают попарную работу и поэтому позволяют вводить и запоминать шестнадцатиразрядные числа. Принято такое попарное соединение регистров: BC, DE и HL.

Регистр СК называют **счетчиком команд,** он содержит адрес очередной ячейки памяти, к которой обращается микропроцессор. Все триггерные выходы этого регистра подключены к выводам А0—А15. После выполнения каждой команды адрес в регистре СК автоматически увеличивается на единицу и поэтому двоичное число на выводах А0—А15 тоже увеличивается на единицу, что приводит к обращению к соседней ячейке памяти, где размещены код операции или данные следующей команды.

Регистр F называют **регистром признаков,** а регистр УС — **указателем стека.** Назначение и принцип действия этих регистров будут рассмотрены далее.

Устройство АЛУ и УУ выполняют в МП—К580 те же действия, что и в четырехразрядном микропроцессоре, а именно: АЛУ производит арифметическо-логические операции, а УУ — управляет работой АЛУ, внутренними регистрами и рядом дополнительных узлов и устройств, не показанных на рис. 28 вклейки.

Генератор тактовых сигналов МП—К580 отличается от генератора четырехразрядного микропроцессора. Это отличие сводится к тому, что он вырабатывает два тактовых сигнала, подаваемые на два вывода Ф1 и Ф2. Оба сигнала сдвинуты по времени один относительно другого, то есть когда действует один, бездействует второй, и наоборот. Такая система запускающих сигналов напоминает подачу сдвоенной команды, например: «Напра-во», «Кругом» Аналогично и здесь, сперва подается часть команды в виде сигнала Ф1, а потом, спустя некоторое время, вторая часть — сигнал Ф2. Формирование таких сдвоенных сигналов представляет известные трудности, что является недостатком МП—К580, с которым приходится мириться.

В заключение укажем, что МП—К580 имеет 10 управляющих выводов (ПМ, ВД, ГТ, ОЖ, ЗХ, ПЗХ, ЗПР, РПР, С и СБР), изображенных на рис. 28 вклейки снизу. К этим выводам подключается шина управления, по которой сигналы передаются из микропроцессора и в микропроцессор для согласования совместных действий с внешними устройствами, а также для изменения режимов их работы.

По управляющей шине пересылаются и другие сигналы управления, не указанные на рис. 28 вклейки, так что общее количество проводов у этой шины может не совпадать с количеством управляющих выводов МП—К580.

Действия управляющих сигналов можно будет понять только тогда, когда мы изучим принцип работы МП—К580, а также узнаем основные правила составления программ. Поэтому сигналы управления будут рассматриваться нами постепенно, по мере ознакомления с микропроцессором, а сейчас мы лишь расшифруем название управляющих выводов МП—К580 и коротко поясним назначение сигналов, появляющихся на этих выводах.

Название вывода ПМ происходит от сокращения слова ПРИЕМ. На этот вывод микропроцессор выдает управляющий сигнал, указывающий на то, что он находится в режиме приема,

и, следовательно, может принять данные по шине данных из памяти или внешнего устройства.

Вывод ВД означает сокращение слова ВЫДАЧА. Появление сигнала на этом выводе происходит в том случае, когда микропроцессор может выдать данные по шине данных в память или внешнее устройство.

Обозначение вывода ГТ является сокращением слова ГОТОВ. На вывод ГТ микропроцессор должен получить сигнал от внешнего устройства, сообщающий о готовности обмена данными с микропроцессором. При отсутствии сигнала на выводе ГТ микропроцессор останавливает свою работу и переходит в режим ожидания.

Вывод ОЖ означает сокращение слова ОЖИДАНИЕ. На этом выводе появляется выходной сигнал микропроцессора, извещающий о том, что он находится в состоянии ожидания сигнала от внешних устройств о их готовности к обмену данными.

Наименование вывода ЗХ происходит от сокращения слов ЗАХВАТ ШИН. На вывод ЗХ поступает сигнал от внешних устройств о запросе разрешения на использование ими (захвата) шин адреса и данных, подключенных к микропроцессору. Этот захват бывает необходим для прямого доступа внешних устройств к памяти, минуя микропроцессор.

Вывод ПЗХ означает ПОДТВЕРЖДЕНИЕ ЗАХВАТА ШИН. На этот вывод микропроцессор выдает сигнал, разрешающий внешним устройствам использовать шины данных и адреса, а сам отключается от этих шин.

Обозначение вывода ЗПР является сокращением слов ЗАПРОС ПРЕРЫВАНИЯ. На вывод ЗПР поступает сигнал от внешних устройств с требованием прерывания основной программы, выполняемой микропроцессором, и внеочередного обслуживания более важной и нужной программы внешнего устройства.

Значение букв вывода РПР относится к сокращению слов РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ. На выводе РПР появляется сигнал микропроцессора, указывающий на то, что он прервал выполнение своей программы и готов выполнить внеочередную программу внешнего устройства.

Буква С на управляющем выводе означает слово СИНХРОНИЗАЦИЯ. Этот вывод микропроцессор использует для выдачи сигнала о начале нового цикла работы, о том, что он приступил к очередному этапу выполнения команды.

Название вывода СБР означает СБРОС. На вывод СБР в микропроцессор поступает входной сигнал, сбрасывающий счетчик команд (СК) в начальное, нулевое состояние, в результате чего микропроцессор начинает свою работу с обращения к ячейке памяти с нулевым адресом.

На все перечисленные выводы управления микропроцессор выдает или получает от внешних устройств управляющие сигналы в виде высокого (1) или низкого (0) логического уровня.

Помимо обозначений выводов в виде сокращенной записи русских слов, в литературе можно встретить обозначения в виде

латинских букв. Соответствие различных видов обозначений выводов микропроцессора и микросхем приведено в табл. 9 Приложения.

Прежде чем перейти к рассмотрению внутреннего устройства и принципа работы МП—К580 напомним, что микропроцессор не может работать самостоятельно, без запоминающего устройства (памяти), так как программа его работы и данные располагаются в памяти. Поэтому перед изучением микропроцессора нужно ознакомиться с устройством памяти.

Память состоит из запоминающих **ячеек.** Обычно под словом «ячейка» подразумевают нечто одиночное: отдельное звено в сети, отдельную ячейку в сотах и др. Здесь же ячейка, служащая для хранения двоичного числа, состоит из нескольких отделений или элементов (рис. 29 вклейки). Количество отделений равно разрядности запоминаемого числа. Ячейка памяти, используемая для работы с восьмиразрядным МП—К580, состоит из восьми отделений, в каждом из которых хранится один разряд (**1 бит**) двоичного числа (нуль или единица), а вся ячейка запоминает восемь разрядов двоичного числа или 1 **байт** (8 разрядов = 8 бит = 1 байт).

В повседневной микропроцессорной практике под байтом понимают восемь разрядов (битов) двоичного числа. В то же время в литературе байт часто рассматривается как единица информации (данных) в 8 раз большая, чем наименьшая единица— бит. Еще более крупная единица, состоящая из нескольких байтов, называется **словом.** Слово кратно по длине байту, а иногда имеет длину в один байт.

Упрощенная схема памяти приведена на рис. 30 вклейки. Справа показаны адресные выводы a0—a15 для подключения адресной шины, снизу изображены входные выводы b0—b7, а сверху — выходные выводы c0—c7 для подключения шины данных. Слева находятся управляющие выводы ЧТ (чтение) и ЗП (запись). При появлении на адресных выводах a0—a15 двоичного адресного числа из всех ячеек памяти особое устройство, называемое **дешифратором,** выбирает одну ячейку, номер которой соответствует адресному числу.

Режим работы выбранной ячейки зависит от логических уровней (1 или 0), подаваемых на выводы ЧТ и ЗП. Обычно блок управления памяти устроен так, что режим чтения данных реализуется при подаче на вывод ЧТ уровня 1. В этом случае выходы выбранной ячейки подключаются к выводам c0—c7 и содержимое ячейки по шине данных поступает в микропроцессор для чтения. Если же уровень 1 подать на вывод ЗП, то выполняется режим записи, при котором входы ячейки памяти подключаются к выводам b0—b7, куда по шине данных поступают данные из микропроцессора для записи.

Количество ячеек, размещаемых в памяти, зависит от количества адресных выводов. При 16 выводах удается разместить 65536 ячеек, так как на 16 выводах можно установить 65536 комбинаций из нулей и единиц, представляя этим двоичные адрес-

ные числа. Начальная, нулевая ячейка (№ 0) будет выбираться дешифратором, если на выводах a0—a15 расположить двоичное число 0000000000000000, а соседняя ячейка (№ 1) — когда будет установлено число 0000000000000001. Самая последняя ячейка № 65535 будет вызываться при адресном числе, состоящем из единиц 1111111111111111 (наибольшее двоичное шестнадцатиразрядное число, эквивалентное десятичному 65535). Для размещения в ячейках памяти двоичных чисел, представляющих собой команды будущей программы, пользуются техническим приспособлением, получившим название **программатор.**

В качестве простейшего программатора можно использовать переключатели, которыми мы вводили данные в четырехразрядный микропроцессор. Как показано на рис. 31 вклейки, для этого нужно к выводам a0—a15 и b0—b7 подключить переключатели, обозначенные буквами К и Л, а к выводам c0—c7 — светодиоды С. Выводы ЧТ и ЗП следует соединить с переключателем, позволяющим подавать на один из них уровень логической единицы (1). Если переключателями К установить адрес нужной ячейки памяти, а переключателями Л — требуемые данные, то при положении переключателя, соответствующему подаче уровня логической единицы (1) на вход ЗП данные поступят и запомнятся в выбранной ячейке памяти.

Чтобы узнать, какие данные удалось поместить в эту ячейку, нужно подать уровень логической единицы на вывод ЧТ, в результате чего выходы ячейки памяти подключатся к выводам c0—c7 и на светодиодах С отобразится двоичное число, находящееся в ячейке.

В качестве примера на рис. 31 вклейки показано, как ввести десятичное число 85 в ячейку памяти № 33.

Зная, как вводятся данные в память, можно составить программу решения какой-либо задачи и ввести ее при помощи программатора в память. Затем отключить программатор и подключить МП—К580, который последовательно обратится к ячейкам памяти, где помещена программа выполняемой задачи.

Для составления программы необходимо знать систему команд МП—К580. Такая система подробно описана в Приложении. В табл. 2.12 приведен перечень некоторых команд, нужных нам для написания программы, взятый из таблицы системы команд МП—К580.

Попутно укажем, что команды МП—К580 бывают трех видов: однобайтные, двухбайтные и трехбайтные. Простейшая однобайтная команда состоит из кода операции, в виде восьмиразрядного двоичного числа, описывающего действия, которые выполняет микропроцессор. В двухбайтной команде первый байт занимает код операции, а во втором находятся данные, над которыми совершается операция. В трехбайтной команде первый байт содержит код операции, а во втором и в третьем располагается адрес того места, где находятся данные.

Приступим к написанию программы для МП—К580 по сложению двух чисел: 3 и 2. При этом будем поступать так же, как и при

составлении программы для четырехразрядного микропроцессора, а именно: поместим слагаемые числа в регистры микропроцессора, сложим содержание регистров в АЛУ, а затем результат сложения поместим в какую-либо ячейку памяти, например 33, чтобы, подключив к этой ячейке светодиоды, можно было бы прочесть итог сложения.

Программу следует записывать по форме табл. 2.14. В первой колонке этой таблицы будем указывать порядковые номера байтов программы; во второй — двоичный код адресов ячеек памяти, в которых размещаются байты команд; в третьей — описания операций выполняемых команд; в четвертой — двоичные коды, пред-

Таблица 2.13

Описание команды	Двоичный код байтов команды	Комментарии
Поместить в регистр А число, записанное во 2-м байте команды	0 0 1 1 1 1 1 0 Двоичное число	1-й байт команды (код операции) 2-й байт команды (число)
Поместить в регистр В число, записанное во 2-м байте команды	0 0 0 0 0 1 1 0 Двоичное число	1-й байт (код операции) 2-й байт (число)
Сложить содержимое регистров А и В	1 0 0 0 0 0 0 0	Однобайтная команда (коды операции и регистра В)
Переслать в ячейку памяти, адрес которой записан во 2-м и 3-м байтах команды, данные регистра А	0 0 1 1 0 0 1 0 Младшие разряды адреса Старшие разряды адреса	1-й байт (код операции) 2-й байт (1-я часть адреса) 3-й байт (2-я часть адреса)
Переслать данные из регистра А в регистр В	0 1 0 0 0 1 1 1	Однобайтная команда (коды операции и регистров)
Переслать в регистр А данные из порта, адрес которого указан во 2-м байте команды	1 1 0 1 1 0 1 1 Адрес порта	1-й байт (код операции) 2-й байт (адрес порта)
Переслать в порт, адрес которого указан во 2-м байте команды, данные из регистра А	1 1 0 1 0 0 1 1 Адрес порта	1-й байт (код операции) 2-й байт (адрес порта)

ставляющие собой восьмиразрядные двоичные числа каждого байта команды; в пятой — дополнительные пояснения (комментарии) к каждому байту, помещенному в ячейку памяти.

Заполнение табл. 2.14 начнем с записи в колонку 3 описания начальной (нулевой) команды: «Поместить в регистр А десятичное число 3». Согласно табл. 2.13 эта команда двухбайтная. Первый байт содержит код операции, представленный двоичным числом 00111110. Помещаем это число в колонку 4. Вторым байтом является вводимое число (десятичная тройка). Переводим тройку в двоичную форму. Получаем 00000011. Записываем это двоичное число во второй строке колонки. Аналогично выполняем записи в колонках 3 и 4 для следующей команды, которая осуществляет размещение в регистре В десятичного числа два. Затем заполняем строки колонок 3 и 4 для команды сложения содержимого регистров А и В.

И наконец, выполняем последнюю запись, относящуюся к пересылке результата сложения в ячейку памяти 33. Эта команда трехбайтная; в первом байте располагается код (00110010) операции пересылки; во втором и в третьем байтах размещается адрес ячейки памяти 33. Преобразовываем номер ячейки 33 в двоичное адресное число. Десятичное число 33 соответствует двоичному числу 0000000000100001. Восемь младших разрядов этого числа (00100001) помещаем во 2-й байт команды, а восемь старших разрядов (00000000) — в 3-й байт.

Все команды программы написаны. Теперь для каждого байта команды нужно записать в колонке 2 адрес ячейки памяти, в которую он будет помещен. Начинаем с первого байта начальной команды. Размещаем этот байт в нулевой, начальной ячейке памяти, имеющей двоичный адресный код 0000000000000000. Записываем этот код в колонке 2. После него в следующей строке заносим код, на единицу больший 0000000000000001, принадлежащий соседней ячейке, куда будет помещен второй байт, и так поступаем вплоть до восьмой ячейки, имеющей код 0000000000000111, где будет размещен последний (восьмой) байт команды.

Составление программы закончено. Пользуясь программатором, эту программу можно ввести в память, а затем к памяти подключить МП—К580 для выполнения команд.

При работе микропроцессора совместно с памятью возникает проблема соединения шины данных с памятью. В режиме записи, когда микропроцессор желает разместить в память какие-то данные, шину данных приходится подключать ко входам ячеек памяти, а при чтении, когда нужно получать данные,— к выходам. Причем переключение должно выполняться автоматически и с большой скоростью (сотни тысяч переключений в секунду). Решение этой проблемы осуществляется при помощи специального устройства, называемого **шинным формирователем.** Его схема приведена на рис. 32, *а* вклейки. Он имеет три группы восьмиразрядных выводов: К0—К7, Л0—Л7 и М0—М7. Снизу располагаются два управляющие вывода ВШ и ВМ. К выводам К0—К7

Таблица 2.14

Порядковый номер байтов программы	Двоичный код адреса ячейки памяти, в которой размещается байт команды или данных	Описание команды	Двоичный код байтов команды или данных	Комментарии
1	2	3	4	5
1	0000000000000000	Поместить в регистр А десятичное число 3	00111110	Код операции
2	0000000000000001		00000011	Двоичное выражение числа 3
3	0000000000000010	Поместить в регистр В десятичное число 2	00000110	Код операции
4	0000000000000011		00000010	Двоичное выражение числа 2
5	0000000000000100	Сложить содержимое регистров А и В	10000000	5 разрядов — код операции 3 разряда — код регистра В
6	0000000000000101	Результат сложения переслать из регистра А в ячейку памяти 33	00110010	Код операции
7	0000000000000110		00100001	Младшие разряды адреса ячейки памяти 33
8	0000000000000111		00000000	Старшие разряды адреса ячейки памяти 33

подключаются выходы ячеек памяти, к выводам Л0—Л7— входы ячеек памяти, а к выводам М0—М7 подключается шина данных.

Если на управляющий вывод ВШ подать низкий логический уровень (0), то выводы К0—К7 соединяются с выводами М0—М7 (рис. 32, *б* вклейки) и данные из ячеек памяти проходят через шинный формирователь в шину данных и по ней поступают в микропроцессор (реализуется режим чтения). Если же на вывод ВШ подать высокий логический уровень (1), то выводы М0—М7 соединяются с выводами Л0—Л7 (рис. 32, *в* вклейки) и данные из микропроцессора по шине данных проходят через шинный формирователь и поступают на входы ячеек памяти (реализуется режим записи). Проводники шинного формирователя под воздействием сигнала ВШ переключаются при помощи сложной комбинации логических схем И, ИЛИ, НЕ, находящихся внутри шинного формирователя и не указанных на рис. 32 вклейки.

Шинный формирователь работает при условии, когда на управляющий вывод ВМ подается сигнал 0. Если же на ВМ подать 1, то выходы Л0—Л7 и М0—М7 шинного формирователя переходят в так называемое **высокоимпедансное состояние** и шинный формирователь отключается от шины данных и выводов памяти. Поясним указанное состояние. Для этого вспомним работу переключателя устройства ввода данных. В нижнем положении ручки переключателя (рис. 33, *а* вклейки) контакт А соединяется с верхним контактом Б и на выходе переключателя устанавливается низкий логический уровень (0). При переводе ручки в верхнее положение (рис. 33, *б* вклейки) контакт А соединяется с нижним контактом В и на выходе возникает высокий логический уровень (1). Если же ручку переключателя установить в среднее, нейтральное положение (рис. 33, *в* вклейки), при котором контакт А не соединяется с контактами Б и В, то на выходе не будет никакого электрического уровня (бесконечно большое сопротивление). Такое состояние и называют высокоимпедансным. В литературе его иногда называют третьим логическим состоянием, считая первым состоянием — наличие логической единицы, вторым — наличие логического нуля.

Таким образом, при подаче на управляющий вывод ВМ (рис. 32, *а* вклейки) уровня логической единицы все выводы Л0—Л7 и М0—М7 шинного формирователя переходят в третье состояние и, следовательно, отключаются от шины данных и выводов памяти. Такой режим, как мы узнаем позже, часто используется для прямого доступа к памяти, когда нужно отключить микропроцессор от памяти и заняться программированием (вводом двоичных чисел в ячейки памяти при помощи программатора).

Несколько слов об изображении микропроцессорных схем. Характерной особенностью этих схем является наличие многопроводных шин данных, адреса и управления. Обилие проводов, из которых состоят шины, затрудняет процесс черчения и ухудшает восприятие схемы. Поэтому многопроводные шины (рис. 3) условились изображать в виде плоской полоски с фигурной стрел-

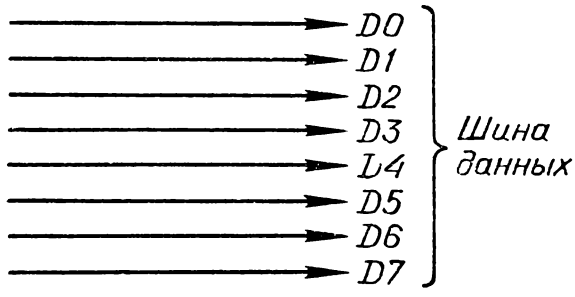


Рис. 3

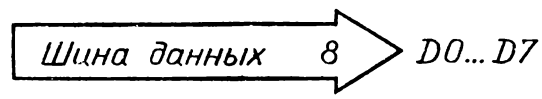


Рис. 4

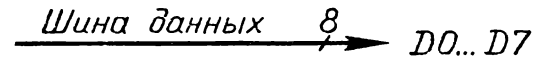


Рис. 5

кой на конце (рис. 4) или в виде толстой линии, пересеченной короткой черточкой, возле которой пишут цифру, указывающую количество проводов в данной шине (рис. 5).

Кроме того, ряд выводов (рис. 6) изображают в виде одного вывода (рис. 7) или одной точки (рис. 8), возле которых пишут обозначение (*D*0—*D*7), из которого можно понять, какое действительное количество (8) выводов соответствует этому единственному выводу или точке. Приведенные правила черчения выполняются, в основном, при изображении структурных (функциональных) схем, а на принципиальных схемах многопроводные соединения часто выполняют так, как показано на рис. 9 и 10. Если требуется показать соединение выводов микросхемы DD1 с выводами микросхем DD2 и DD3, то вместо вычерчивания всех проводников (рис. 9) их нумеруют и собирают в один общий жгут, изображаемый утолщенной линией (рис. 10). Замечая номер проводника, входящего в жгут, разыскивают аналогичный номер, выходящий из жгута и, таким образом, определяют, какие выводы микросхем соединяются друг с другом при помощи данного проводника. Напомним также, что во всех микропроцессорных схемах аналогично интегральным схемам не указывают выводы источников питания.

Для определения функциональных действий микросхем применяются условные обозначения, указанные в табл. 10 Приложения. Более подробно условные графические обозначения элементов цифровой техники приведены в [57].

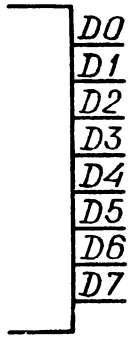


Рис. 6

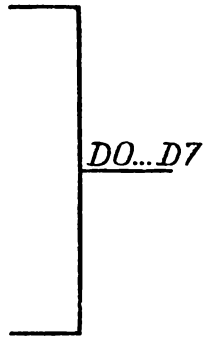


Рис. 7

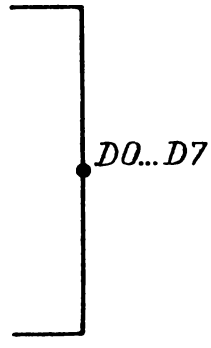


Рис. 8

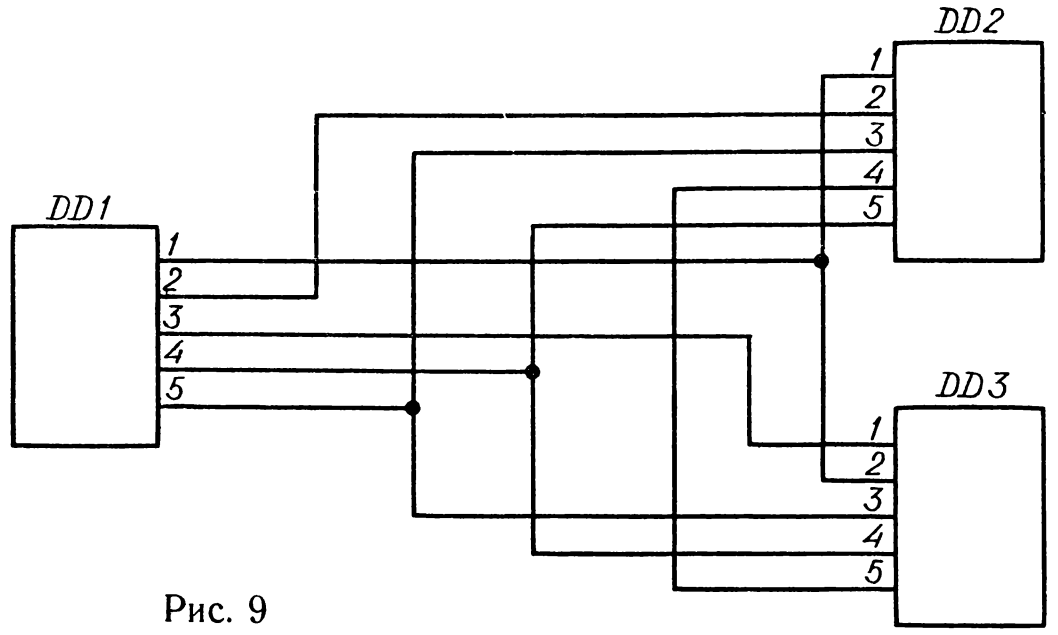


Рис. 9

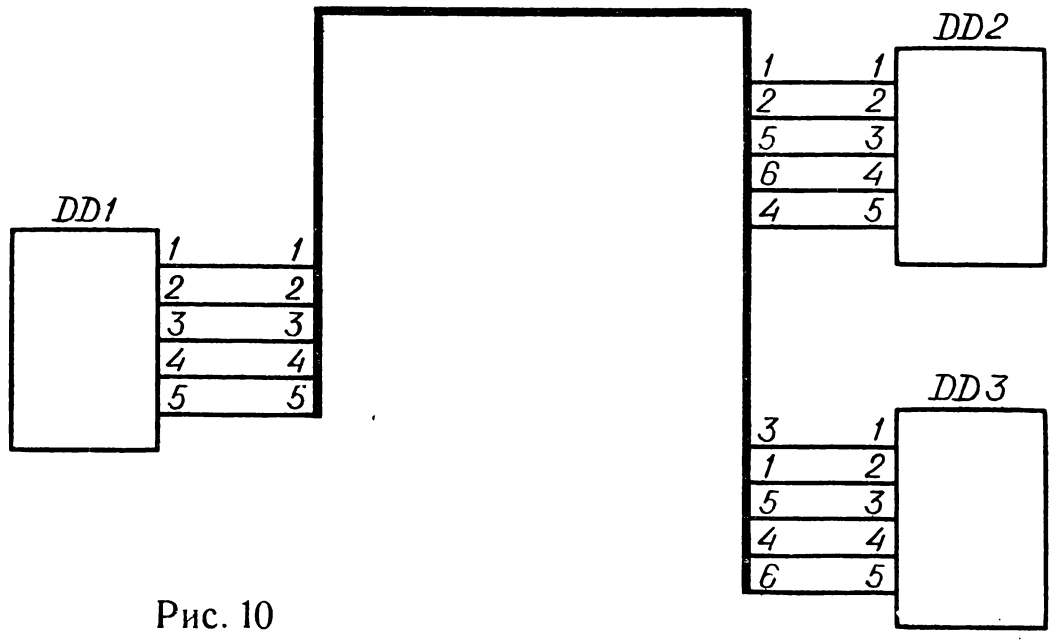


Рис. 10

Для наглядности на рисунках цветных вклеек нашей книги адресные шины и все элементы схемы, связанные с адресами, будут изображаться синим цветом, шина данных и все, что относится к хранению, пересылке и обработке данных — зеленым, а шина управления, управляющие элементы и провода, по которым проходят сигналы управления — красным.

Учитывая упомянутые правила черчения, рассмотрим схему соединения МП—К580 с памятью (рис. 34 вклейки) и объясним принцип ее действия при выполнении команд программы

При нажатии кнопки «Сброс» уровень логической единицы поступает на управляющий вывод СБР МП—К580 в результате чего счетчик команд МП—К580 сбрасывается в начальное нулевое состояние, и на выводах А0—А15, а следовательно, и на адресной шине появляется нулевое двоичное число, соответствующее адресу нулевой ячейки 0000000000000000. Это число по адресной шине

передается на адресные выводы a0—a15 памяти и поступает в дешифратор, который вызывает к действию нулевую № 0 ячейку памяти Затем МП—К580 выдает управляющий сигнал на свой вывод ПМ. Этот сигнал поступает на вывод ВШ шинного формирователя, в результате чего выводы К0—К7 соединяются с выводами М0—М7. Одновременно сигнал ПМ поступает на вывод ЧТ памяти и поэтому выходы нулевой ячейки памяти подключаются к выводам b0—b7. Таким образом, первый байт начальной команды (число 00111110), находящийся в ячейке памяти № 0, поступает по шине данных в МП—К580. Получив этот код операции и узнав из него о необходимости размещения в регистре А двоичного числа второго байта, устройство управления МП—К580 увеличивает показания счетчика СК на единицу, благодаря чему на адресной шине появляется число 0000000000000001 и это приводит к вызову соседней ячейки памяти, содержащей число, которое нужно поместить в регистр А. В результате действия ряда дополнительных сигналов, посылаемых устройством управления, это число поступает в регистр А и запоминается там. Как только операция заканчивается, счетчик СК снова увеличивает свои показания на единицу, что изменяет адрес на единицу и приводит к вызову следующей ячейки памяти. Так последовательно, один байт за другим, вызываются и исполняются все байты команд программы, записанные в ячейках памяти.

Рассмотренная нами программа сложения двух чисел (см. табл. 2.14) не является единственно возможной. МП—К580 допускает исполнение и других вариантов программы сложения чисел, при которых данные можно не помещать предварительно в память, а непосредственно вводить в микропроцессор, путем подключения устройства ввода чисел к шине данных. МП—К580 разрешает подключение к шине данных 256 устройств ввода и 256 устройств вывода данных. В разделе 2.3 подробно рассмотрены различные типы устройств ввода—вывода. Для уяснения принципа совместной работы МП—К580 с устройствами ввода—вывода воспользуемся уже известными нам переключателями, выполняющими роль устройств ввода, и светодиодами, используемыми для вывода данных.

Если к линиям шины данных подключить одновременно несколько переключателей, принадлежащих разным устройствам ввода, то они будут мешать работе друг друга. Один переключатель на линию будет подавать низкий логический уровень, а второй в это время — высокий, и микропроцессор не сможет определить, какой уровень подан. Кроме того, подключение переключателей будет влиять на передачу данных по шине данных из микропроцессора в память и обратно. Поэтому для независимого (управляемого) ввода данных каждое устройство ввода (в нашем случае — переключатели) подсоединяется к шине данных через специальное приспособление, которое получило название **порт.**

Схема порта ввода приведена на рис. 37 вклейки. Порт имеет три группы выводов: К0—К7, Л0—Л7, М0—М7 и управляющий вывод ВШ.

Двоичное число, которое нужно ввести в микропроцессор, набирается с помощью переключателей П и подается на выводы К0—К7. Эти выводы соединяются через выключатели а0—а7 с выводами М0—М7, куда подключается шина данных. В исходном состоянии все выключатели а0—а7 разомкнуты и поэтому данные с переключателей не могут поступить на выводы М0—М7. Включение а0—а7, а следовательно, и разрешение прохождения данных через порт производится при помощи управляющего сигнала ЧТВВ, подаваемого через вывод ВШ на логическую схему совпадения 2И.

Каждый порт ввода и вывода имеет свой номер, позволяющий микропроцессору распознавать порты и включать их в работу в нужный момент времени. С этой целью порт содержит адресные выводы Л0—Л7, на которые микропроцессор посылает адрес в виде двоичного числа, соответствующего номеру порта. Адресное число пересылается по восьми младшим разрядам (А0—А7) адресной шины микропроцессора, а порт имеет дешифратор, который в ответ на двоичное число пришедшего адреса выдает разрешающий сигнал ВМ, поступающий на логическую схему 2И для включения а0—а7. Таким образом, данные через порт передаются при условии поступления адреса данного порта на его выводы Л0—Л7, в результате чего образуется управляющий сигнал ВМ, и прихода управляющего сигнала ЧТВВ на вывод ВШ.

Когда оба эти сигнала (ВШ и ВМ) в виде уровней логических единиц совпадают по времени на схеме 2И, то на выходе 2И появляется сигнал, включающий переключатели а0—а7.

Иногда дешифраторы, установленные в каждом порту, заменяются одним общим дешифратором для всех портов (см. рис. 39 вклейки, справа). В этом случае порт лишается адресных выводов. Эти выводы переносятся на дешифратор адреса, который преобразовывает двоичный адресный код в разрешающий сигнал, посылаемый в нужный порт.

Выключатели а0—а7 изображены на рис. 37 вклейки условно. В реальной схеме порта их роль выполняет набор логических элементов типа И, ИЛИ, НЕ.

Порт вывода (рис. 38 вклейки) устроен так же, как и порт ввода, с той лишь разницей, что данные перемещаются в другом направлении — от выводов шины данных М0—М7 к выводам К0—К7, куда подключены светодиоды С. Кроме того, на управляющий вывод ВШ порта подается сигнал ЗПВВ. В связи с тем, что микропроцессор выдает данные в порт в течение очень короткого промежутка времени, исчисляемого микросекундами, светодиоды С не успевают отобразить поступившие данные. Поэтому порты вывода снабжают восьмиразрядным запоминающим регистром (не указанным на рис. 38 вклейки), который запоминает данные, а потом отображает их в течение нужного времени с помощью светодиодов С. Такой же регистр часто устанавливают и в портах ввода, если данные поступают в порт от кратковременно действующих источников данных.

На рис. 39 вклейки изображена более подробная схема соединения МП—К580 с памятью и устройствами ввода— вывода данных. Левая часть схемы, состоящая из МП—К580, тактового генератора, кнопки «Сброс», памяти, шинного формирователя, адресной шины и шины данных, соответствует ранее рассмотренной схеме (рис. 34 вклейки). Исключение составляет лишь наличие блока Р, вырабатывающего серию управляющих сигналов, посылаемых по шине управления. На правой стороне рис. 39 вклейки изображены два порта ввода и два порта вывода. Эти порты лишены своих дешифраторов адреса и для вызова пользуются сигналами ВМ, поступающими из общего дешифратора адреса.

МП—К580 может выполнять четыре вида действий с данными: 1) чтение (получение) данных из памяти; 2) запись (размещение) данных в памяти; 3) чтение данных из устройства ввода; 4) запись данных в устройство вывода.

Исполнение этих действий осуществляется при помощи четырех управляющих сигналов: 1) **ЧТЗУ,** посылаемых в память при чтении данных; 2) **ЗПЗУ,** посылаемых в память при записи данных; 3) **ЧТВВ**, посылаемых в порт ввода при чтении данных; 4) **ЗПВВ**, посылаемых в порт вывода при записи данных.

Эти сигналы формируются при помощи сложной электронной схемы, использующей специальные кратковременно появляющиеся на шине данных логические уровни (называемые сигналами состояния), в начале выполнения микропроцессором каждого байта команды. Эти уровни (1 или 0) совместно с управляющими сигналами ПМ, ВД и С, выдаваемыми микропроцессором, поступают в электронный блок Р, состоящий из логических элементов И, ИЛИ, НЕ и регистра (не указанных на рис. 39 вклейки), где и образуются сигналы: ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ. Указанные сигналы являются основными управляющими сигналами, позволяющими реализовать различные операции по пересылке данных.

Рассмотрим эти операции, пользуясь схемой рис. 39 вклейки. Пусть требуется микропроцессору получить данные из ячейки памяти № 1. Для этого микропроцессор выдает на адресную шину адресное число 0000000000000001. Это число одновременно поступает и в дешифратор памяти, разрешая работу ячейки № 1, и в дешифратор адресов портов, вызывая порт ввода № 1 и порт вывода № 1. Однако из этих трех вызванных источников данных к шине данных подключится только один — ячейка памяти № 1, потому что микропроцессор пошлет ей дополнительный сигнал ЧТЗУ. Аналогично, если нужно получить данные из порта ввода № 1, то микропроцессор должен послать по шине управления в порт ввода № 1 сигнал ЧТВВ, в результате чего данные только этого порта поступят на шину данных, так как порт ввода № 1 получит два разрешающих сигнала: сигнал ВМ из дешифратора и сигнал ЧТВВ из шины управления. В случае записи данных в ячейку памяти № 1 нужно послать в память сигнал ЗПЗУ, а при записи в порт вывода № 1 — сигнал ЗПВВ.

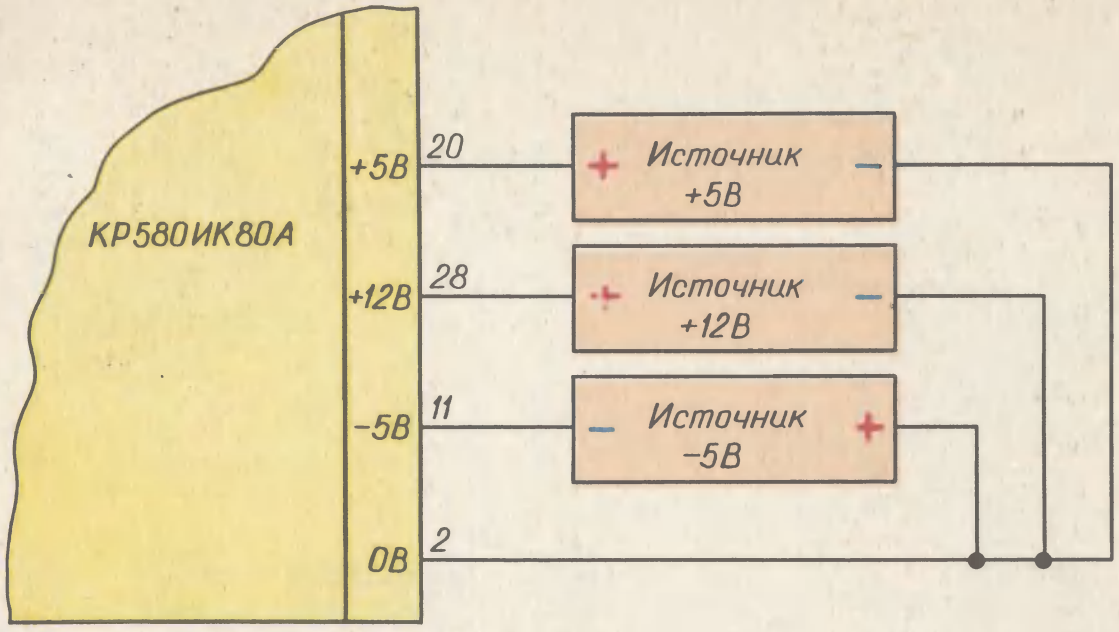


Рис. 1

Рис. 2

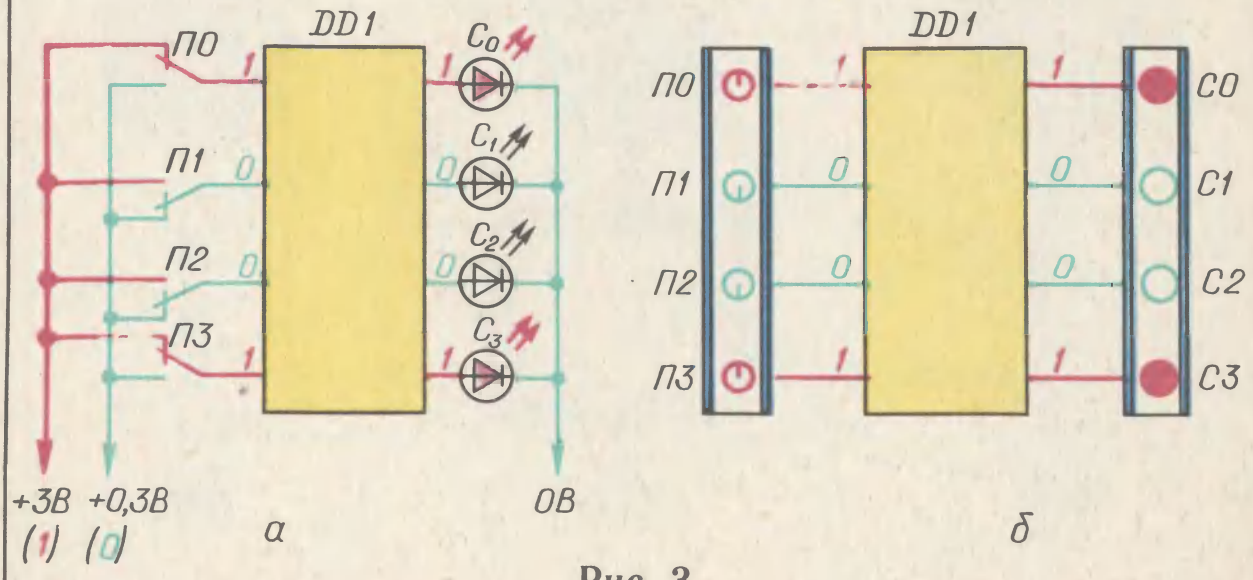


Рис. 3

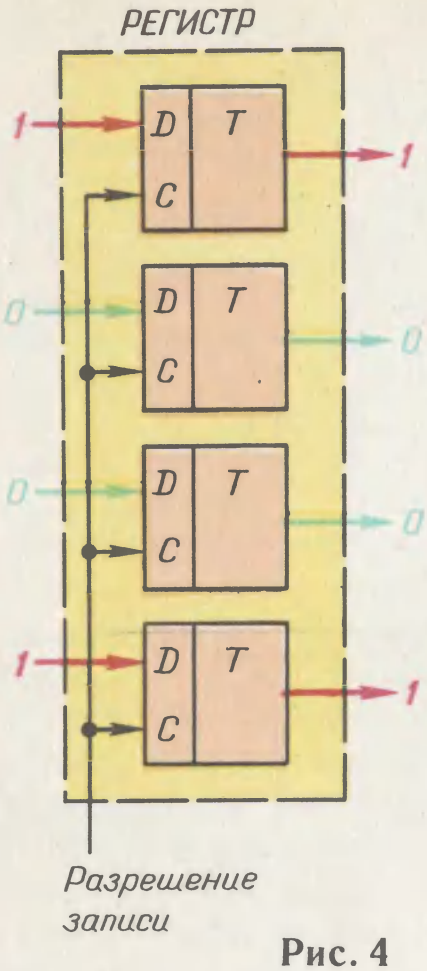


Рис. 4

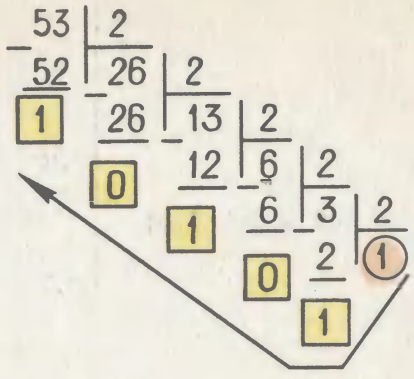


Рис. 5

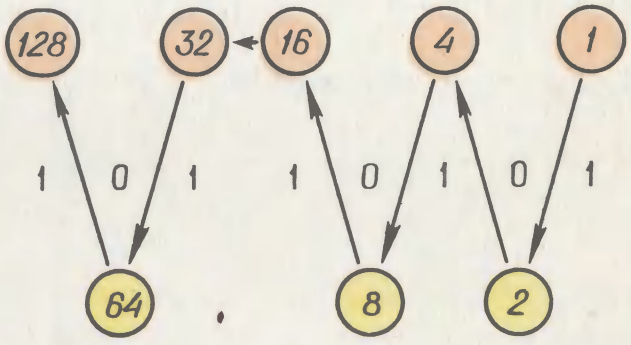


Рис. 6

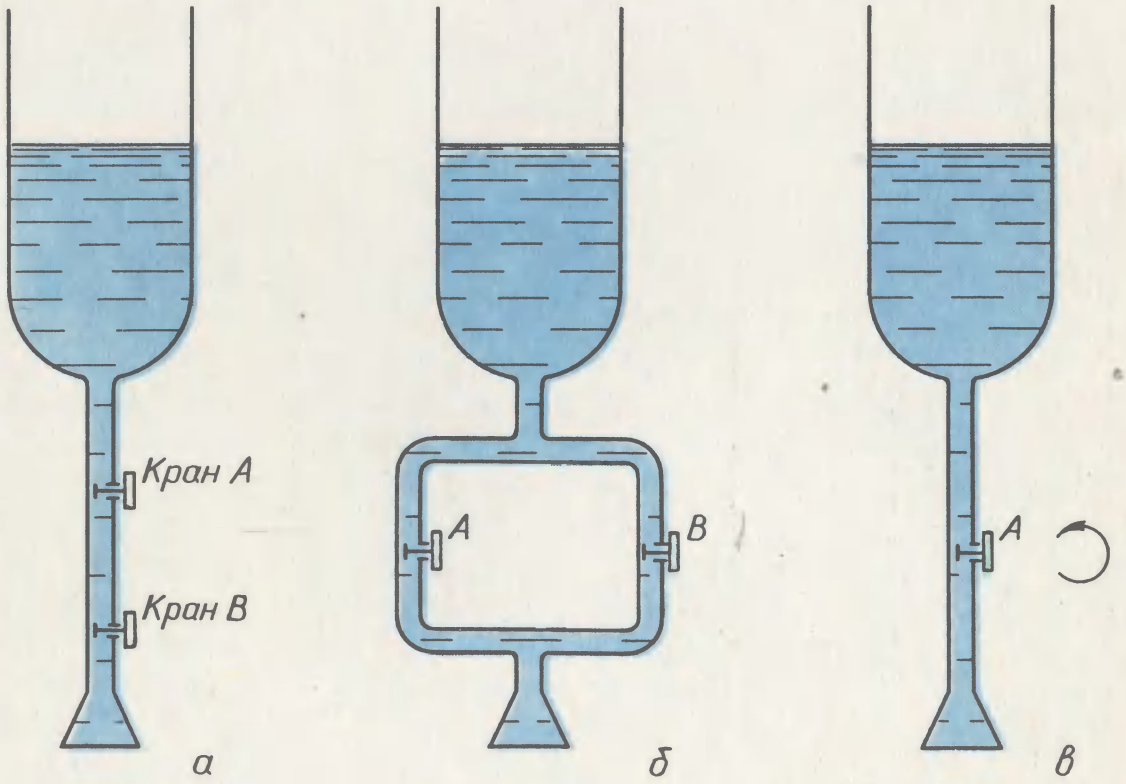
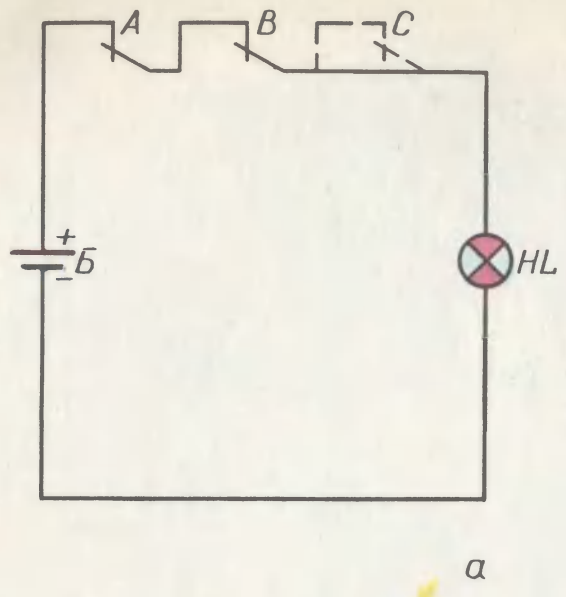
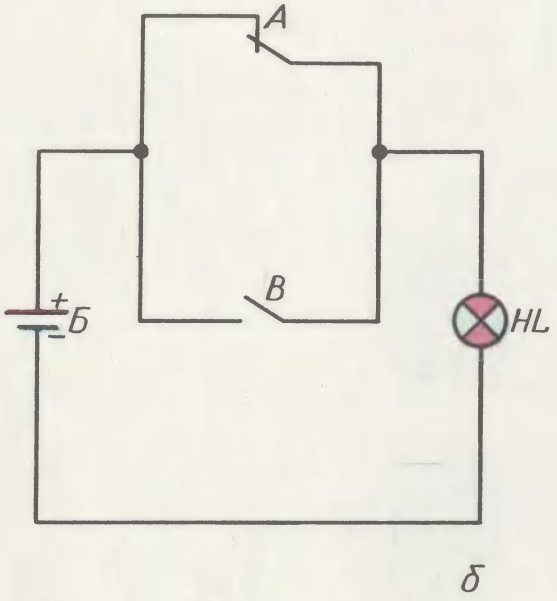


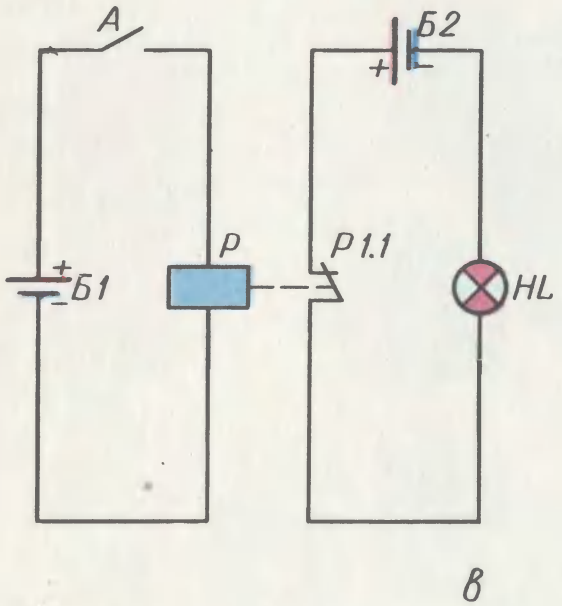
Рис. 7



а

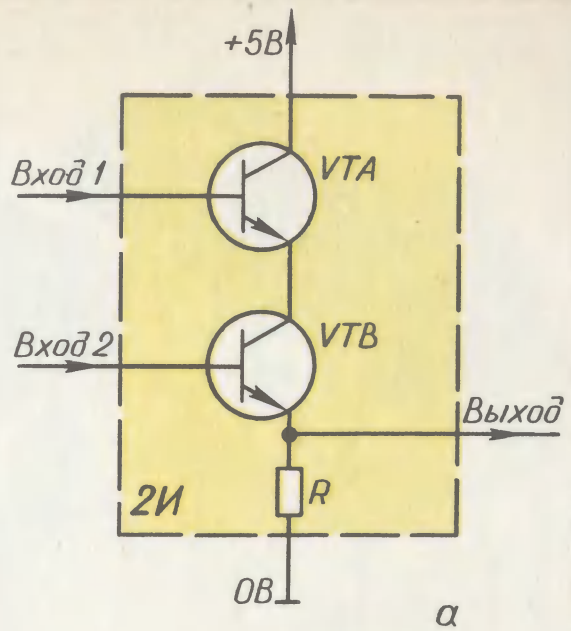


б

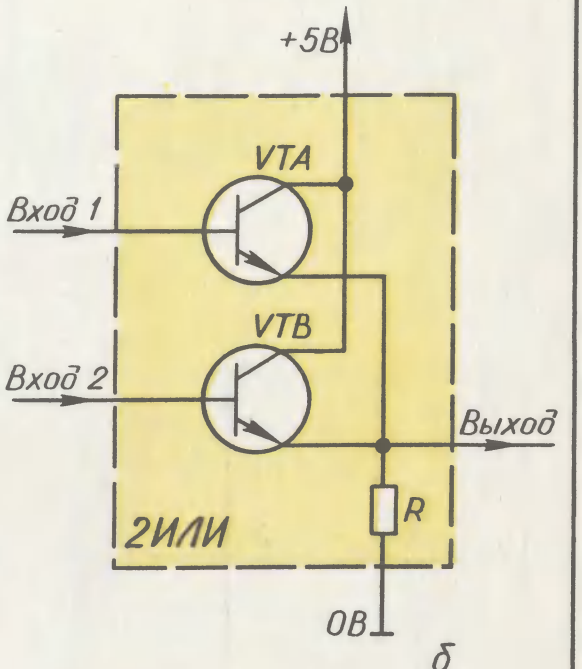


в

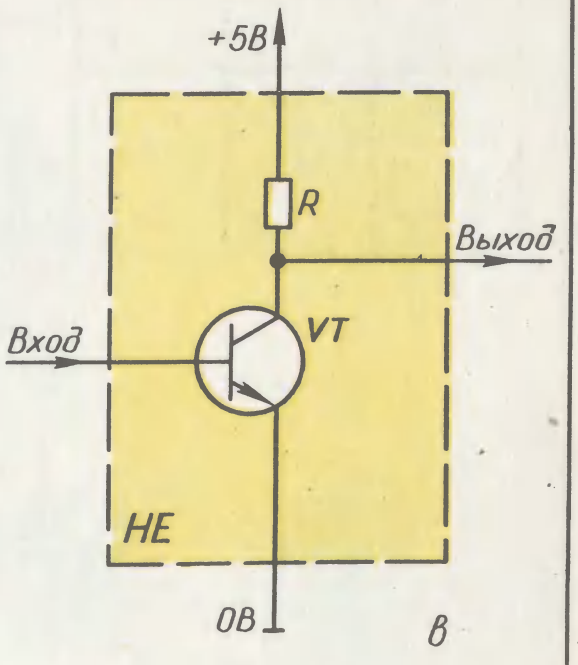
Рис. 8



а



б



в

Рис. 9

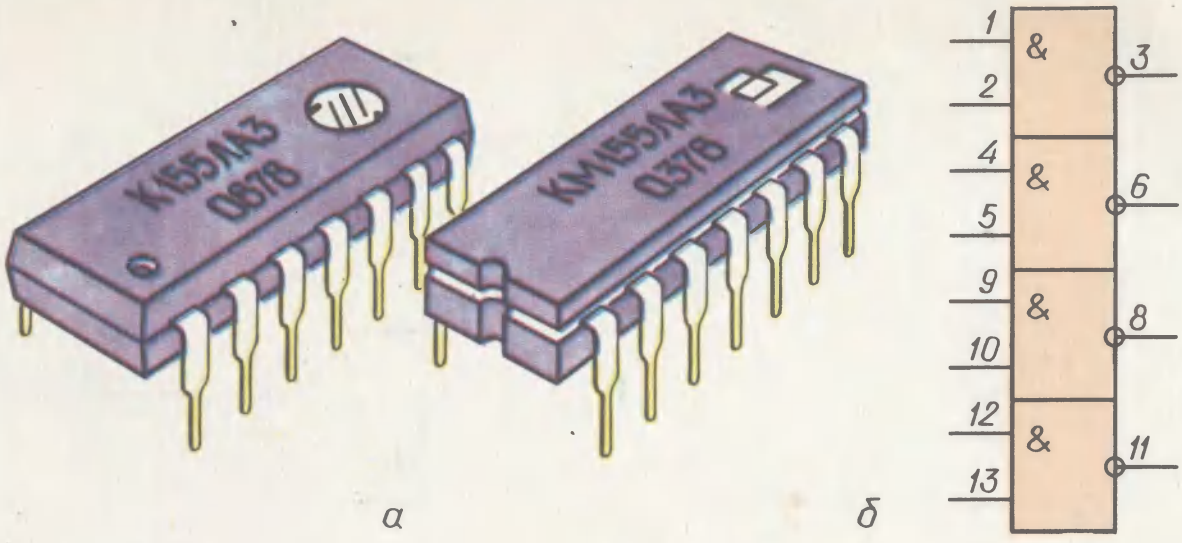

К155ЛА3
0878
КМ155ЛА3
0378
1
2
4
5
9
10
12
13
&
3
6
8
11
а
б

Рис. 10

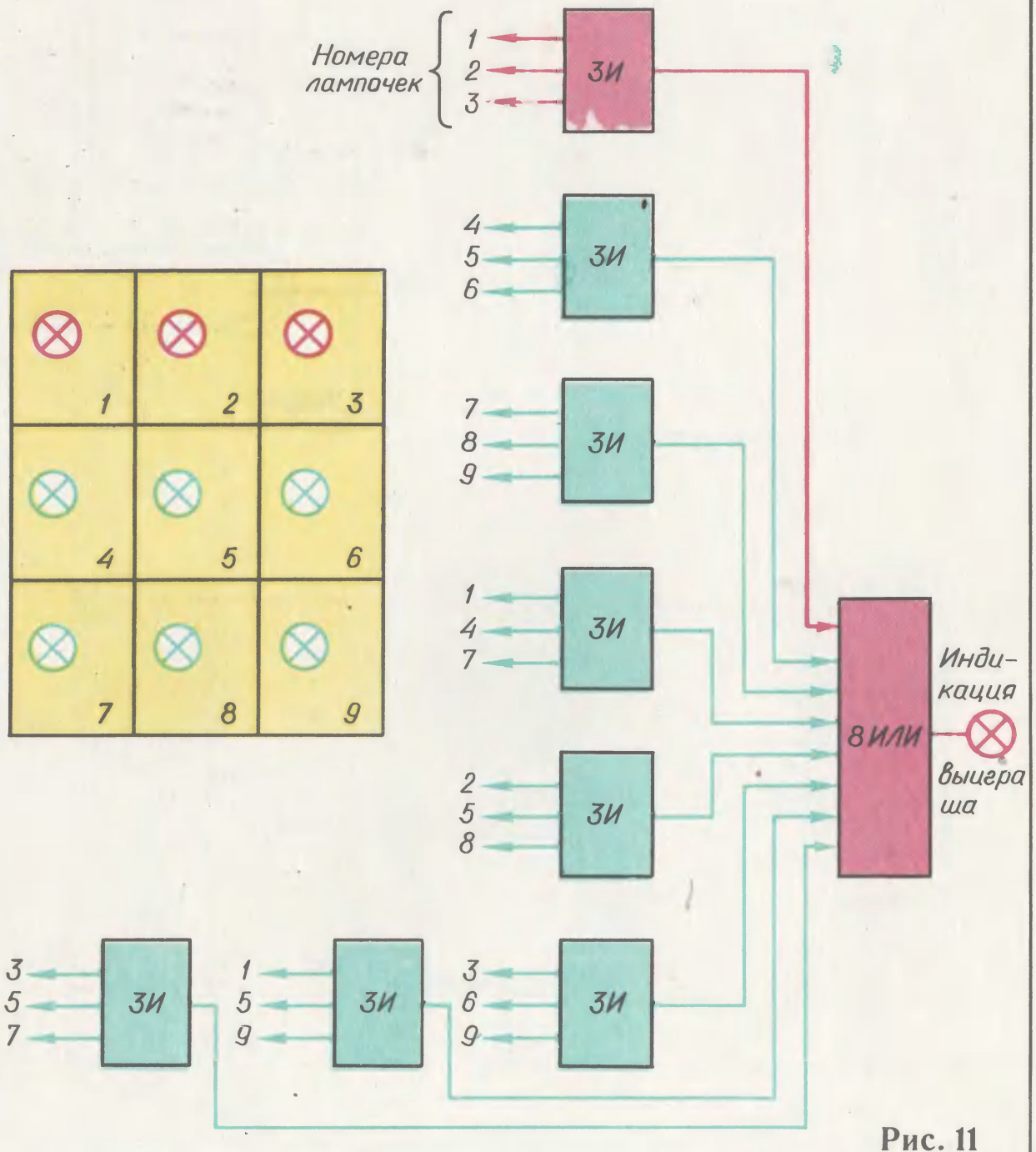

Номера лампочек
1
2
3
ЗИ
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9
8 ИЛИ
Индикация выигрыша

Рис. 11

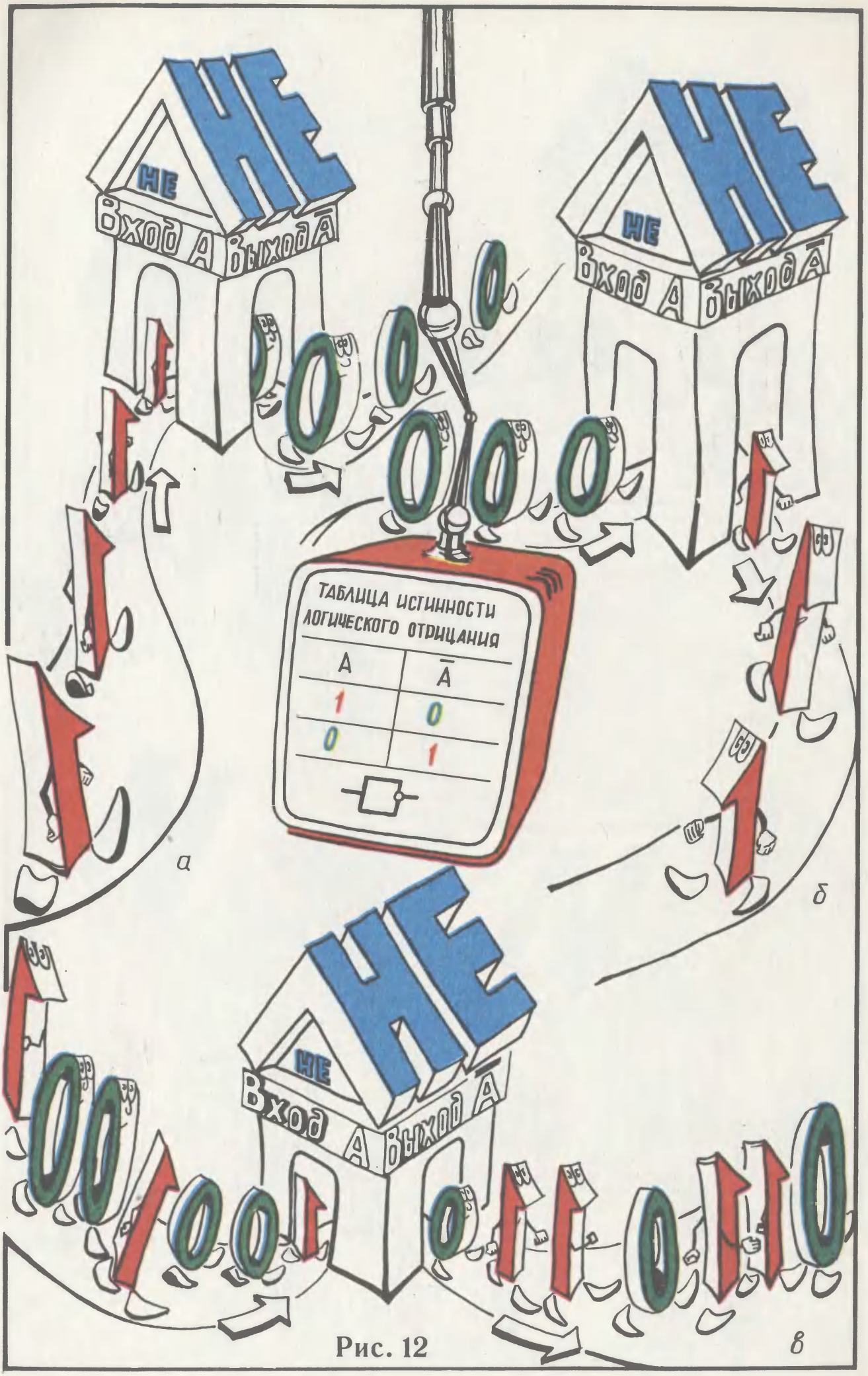

НЕ
НЕ
Вход A Выход $\bar{A}$
НЕ
НЕ
Вход A Выход $\bar{A}$
ТАБЛИЦА ИСТИННОСТИ
ЛОГИЧЕСКОГО ОТРИЦАНИЯ
A | $\bar{A}$
1 | 0
0 | 1
НЕ
НЕ
Вход A Выход $\bar{A}$
а
б
в

Рис. 12


И
Вход А
Вход В
Выход А·В
ТАБЛИЦА ИСТИННОСТИ
ЛОГИЧЕСКОГО УМНОЖЕНИЯ
A
B
A·B
1 0 0
0 1 0
0 0 0
1 1 1
&
а
б
в
г

Рис. 13

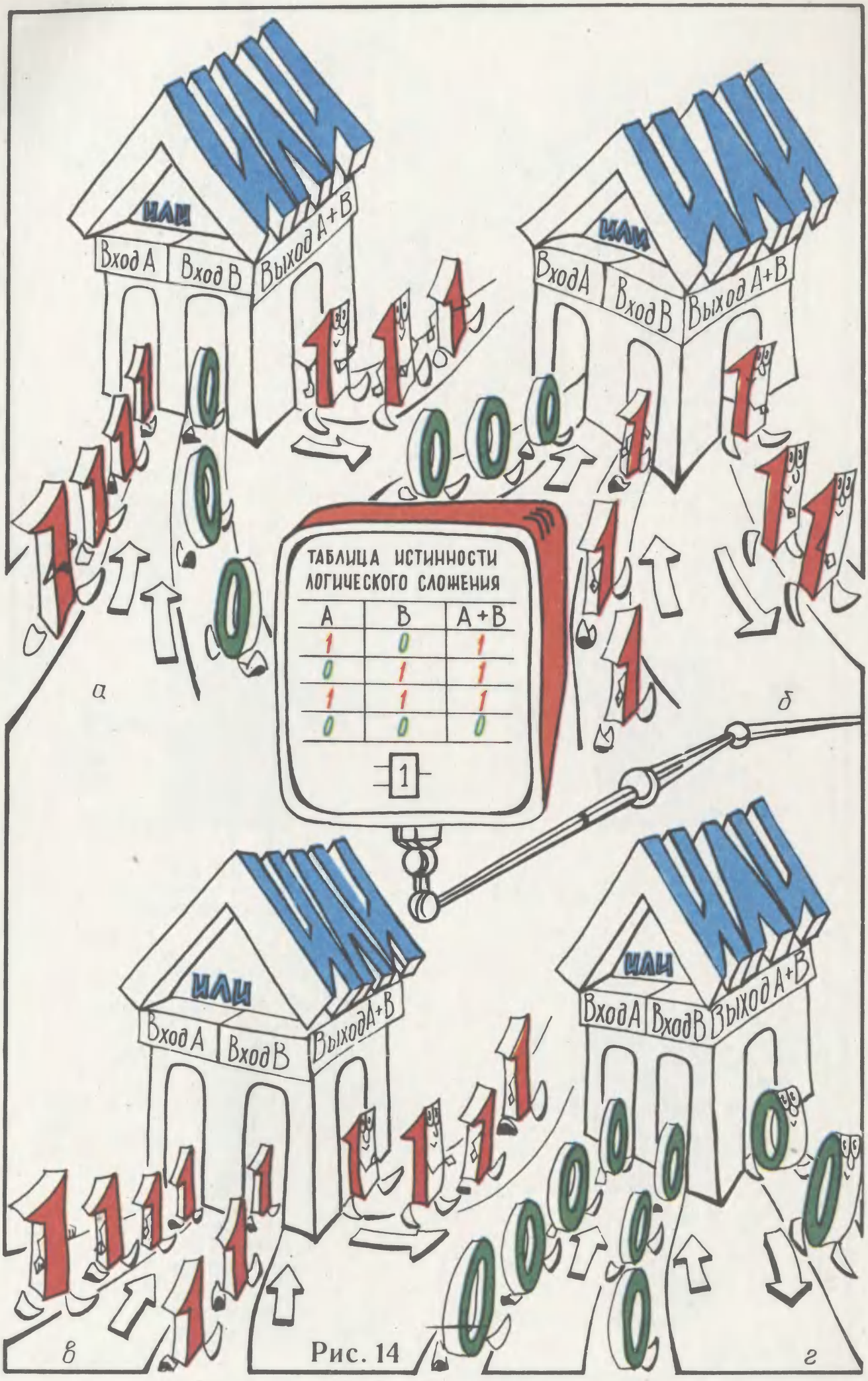
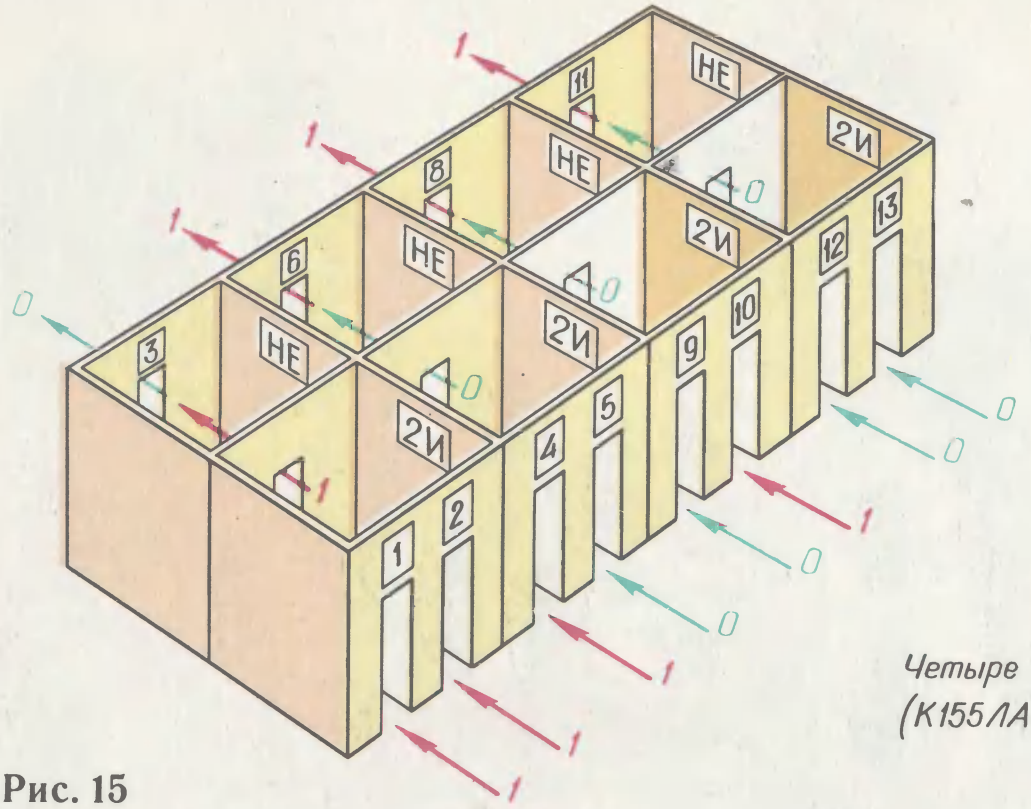

ИЛИ
Вход А
Вход В
Выход А+В
ТАБЛИЦА ИСТИННОСТИ
ЛОГИЧЕСКОГО СЛОЖЕНИЯ
А
В
А+В
1 0 1
0 1 1
1 1 1
0 0 0
1
а
б
в
г

Рис. 14



Четыре 2И-НЕ
(К155ЛА3)

Рис. 15

Рис. 16

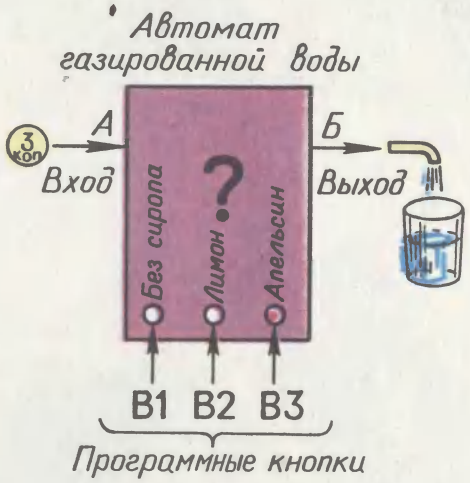


Рис. 17

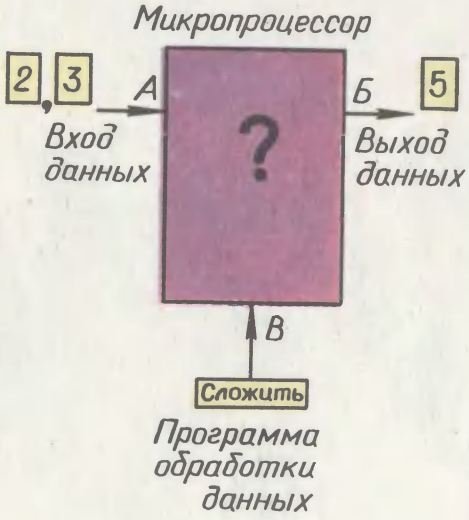


Рис. 18

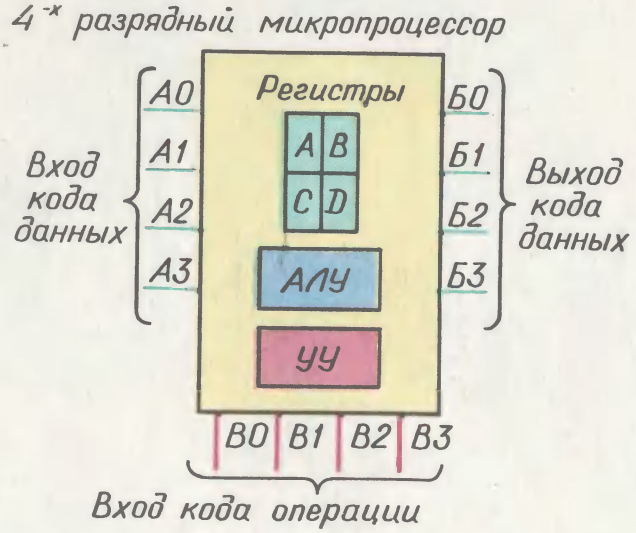


Рис. 19

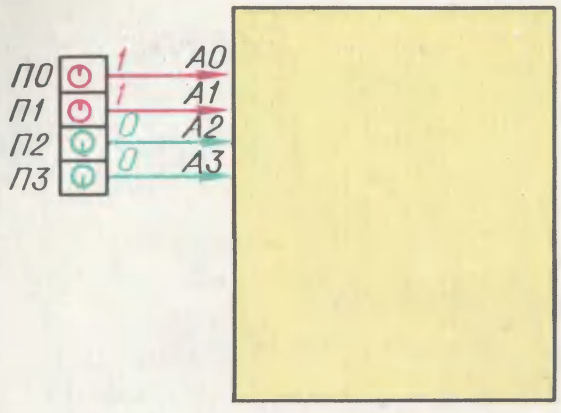


Рис. 20

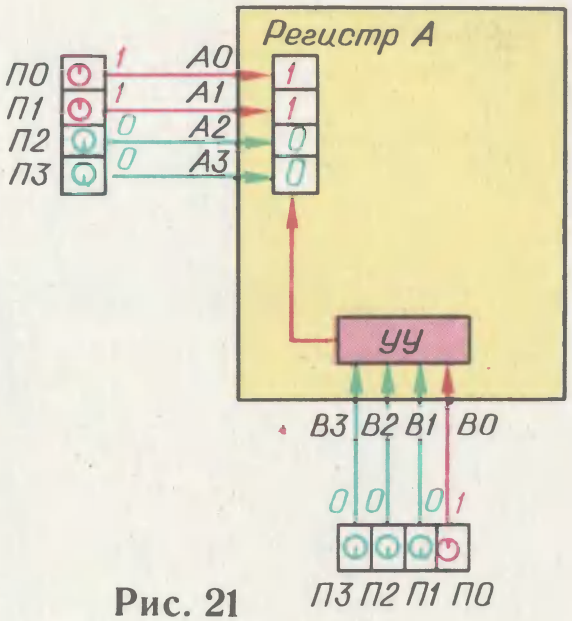


Рис. 21

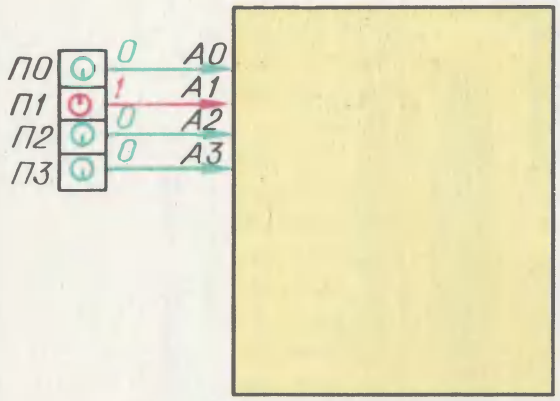


Рис. 22

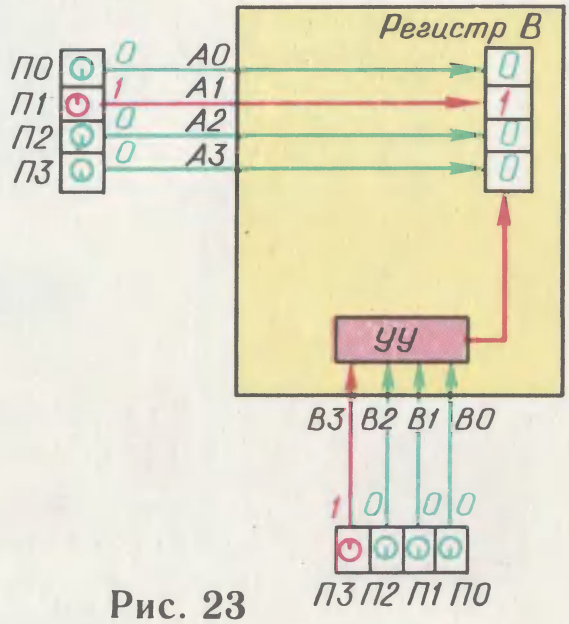


Рис. 23

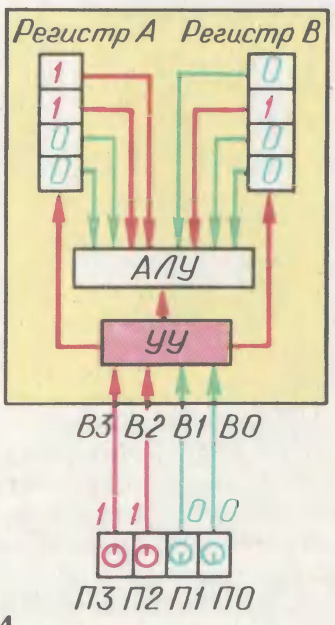


Рис. 24

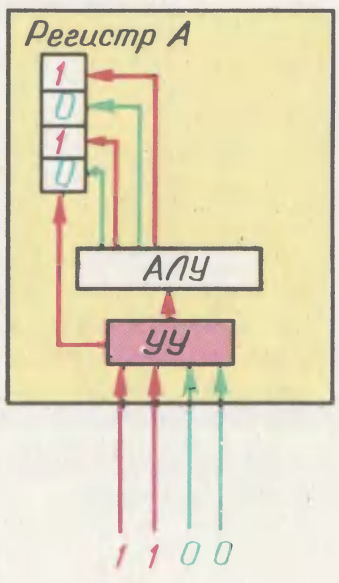


Рис. 25

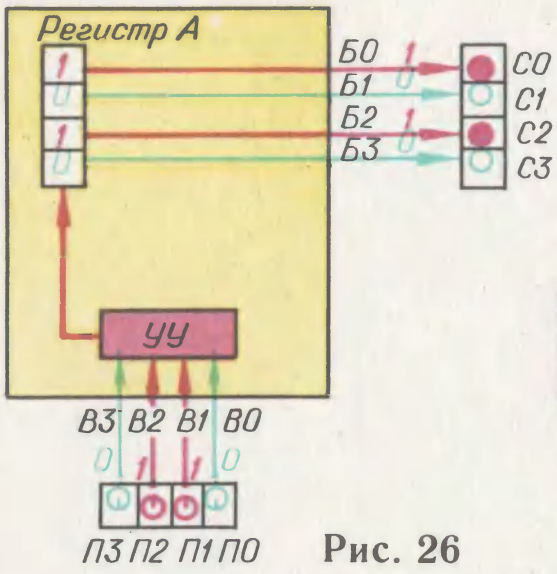


Рис. 26

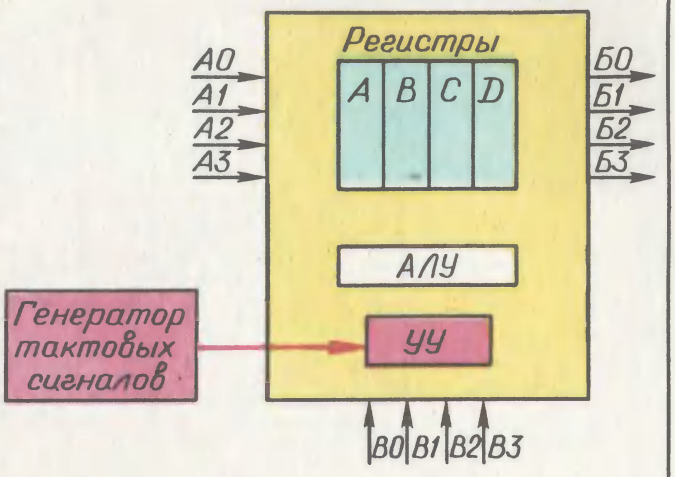


Рис. 27

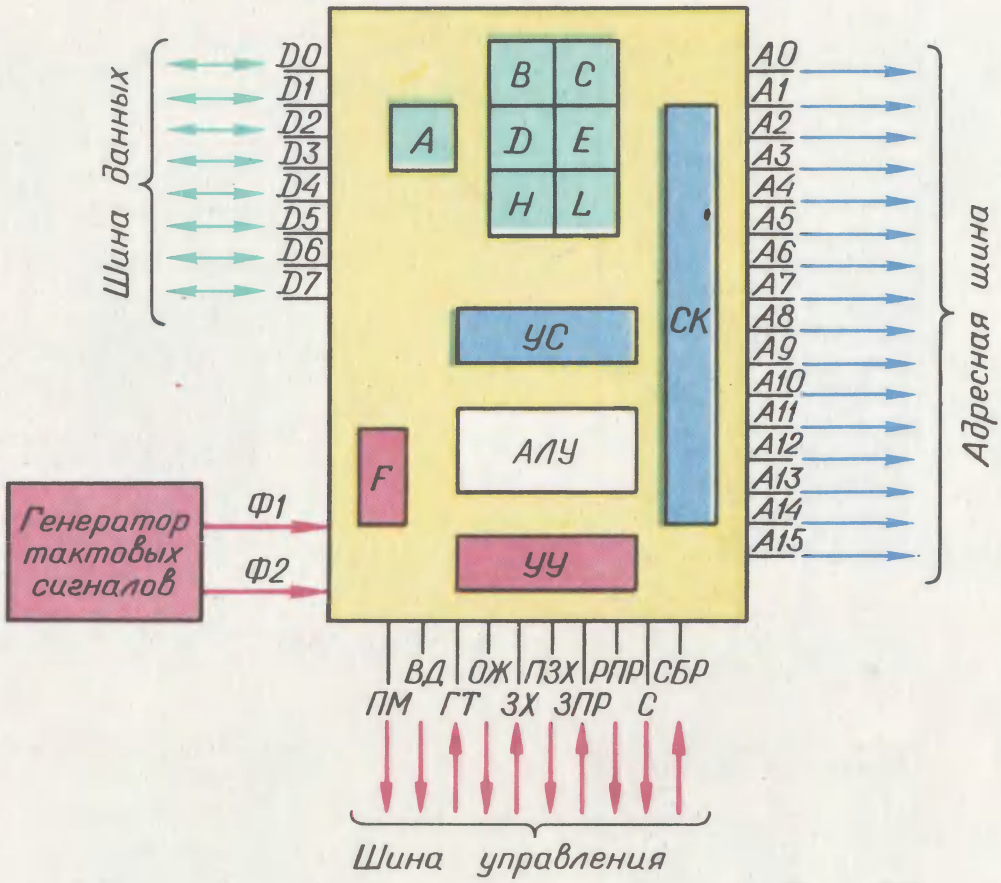


Рис. 28

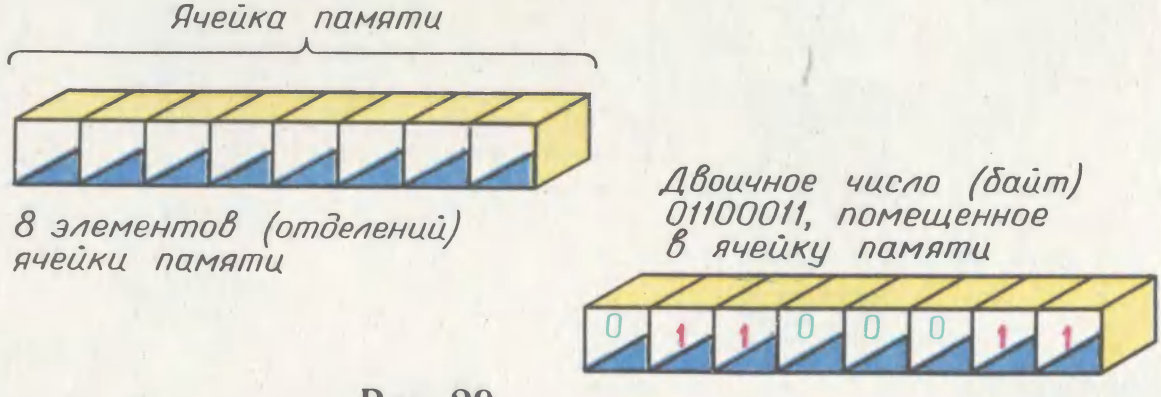


Рис. 29

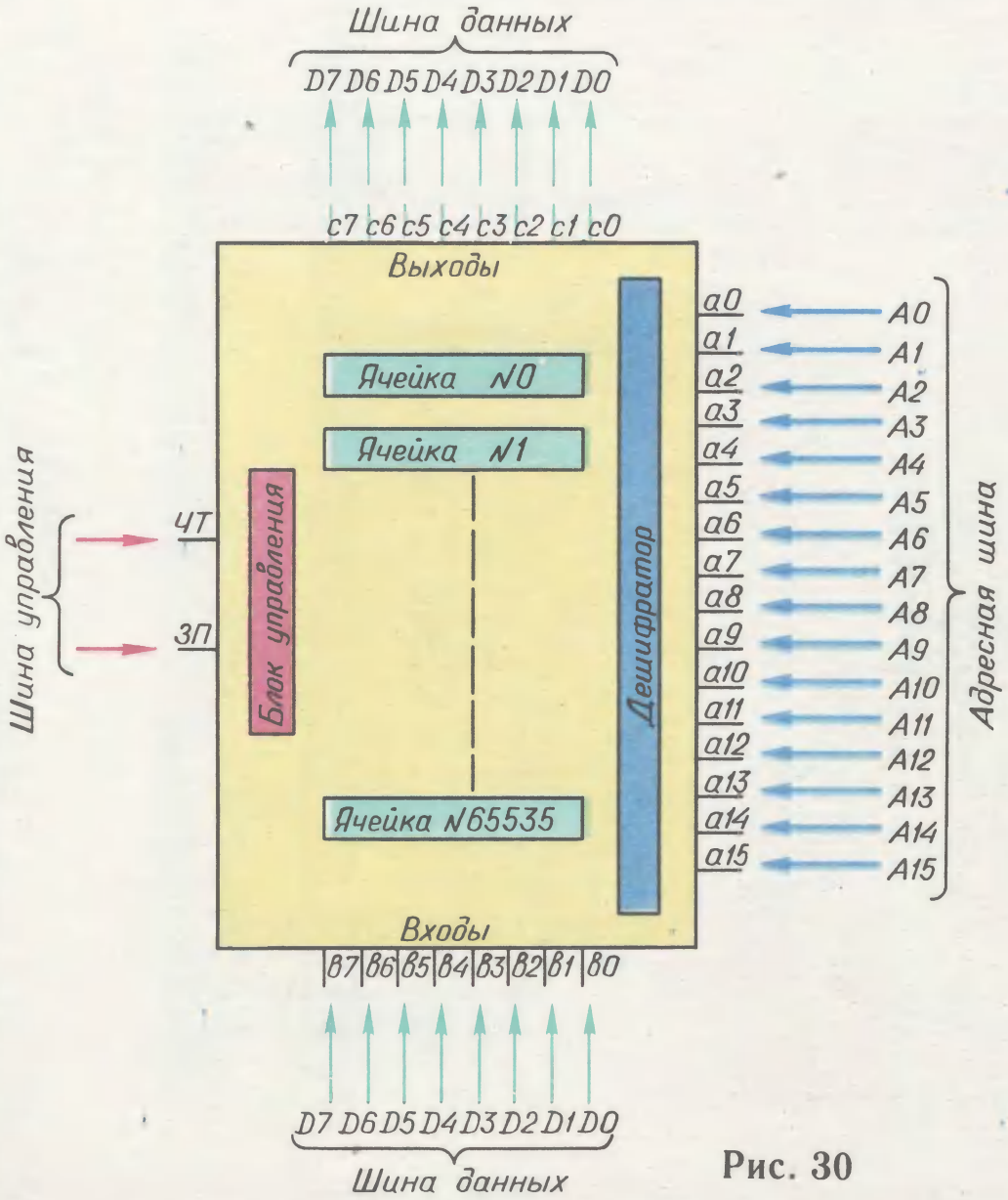

Шина данных
D7 D6 D5 D4 D3 D2 D1 D0
c7 c6 c5 c4 c3 c2 c1 c0
Выходы
Ячейка N0
Ячейка N1
Ячейка N65535
Блок управления
Дешифратор
Шина управления
ЧТ
ЗП
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
Адресная шина
Входы
в7 в6 в5 в4 в3 в2 в1 в0
D7 D6 D5 D4 D3 D2 D1 D0
Шина данных

Рис. 30

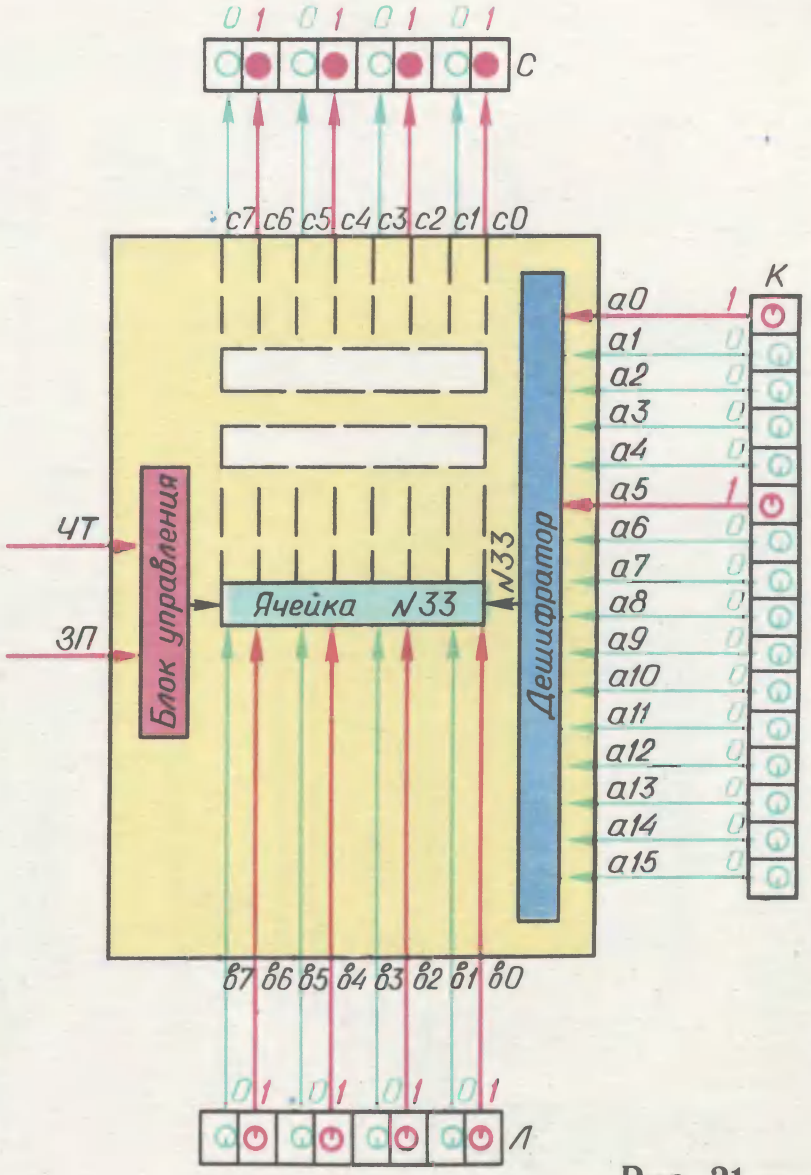
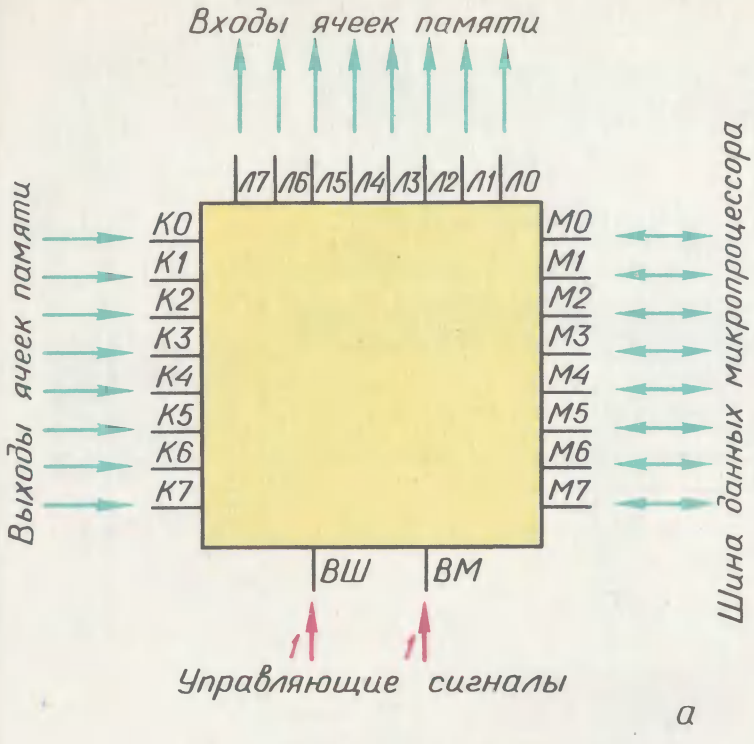
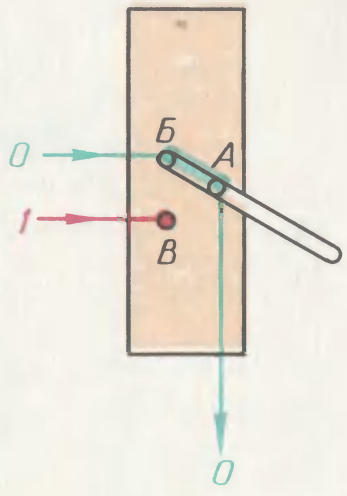

0 1 0 1 0 1 0 1
С
c7 c6 c5 c4 c3 c2 c1 c0
К
ЧТ
ЗП
Блок управления
Ячейка N33
N33
Дешифратор
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
в7 в6 в5 в4 в3 в2 в1 в0
0 1 0 1 0 1 0 1
Л

Рис. 31

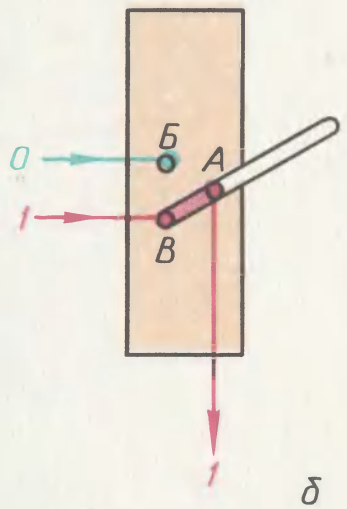


а

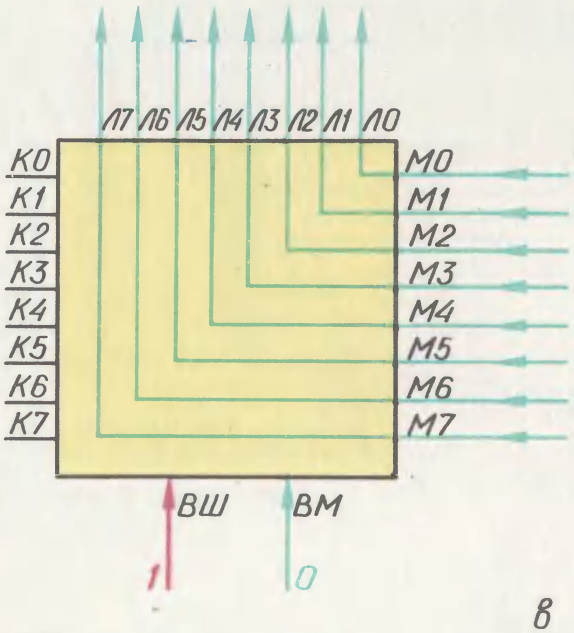


а

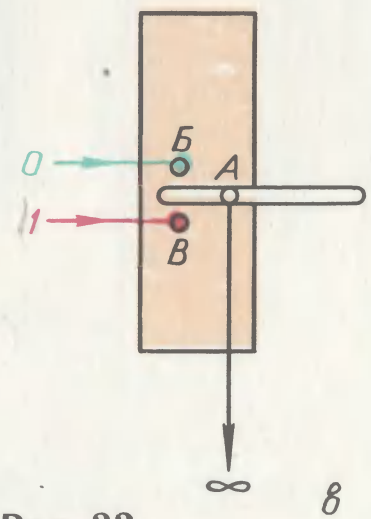
б



б



в



в

Рис. 32

Рис. 33

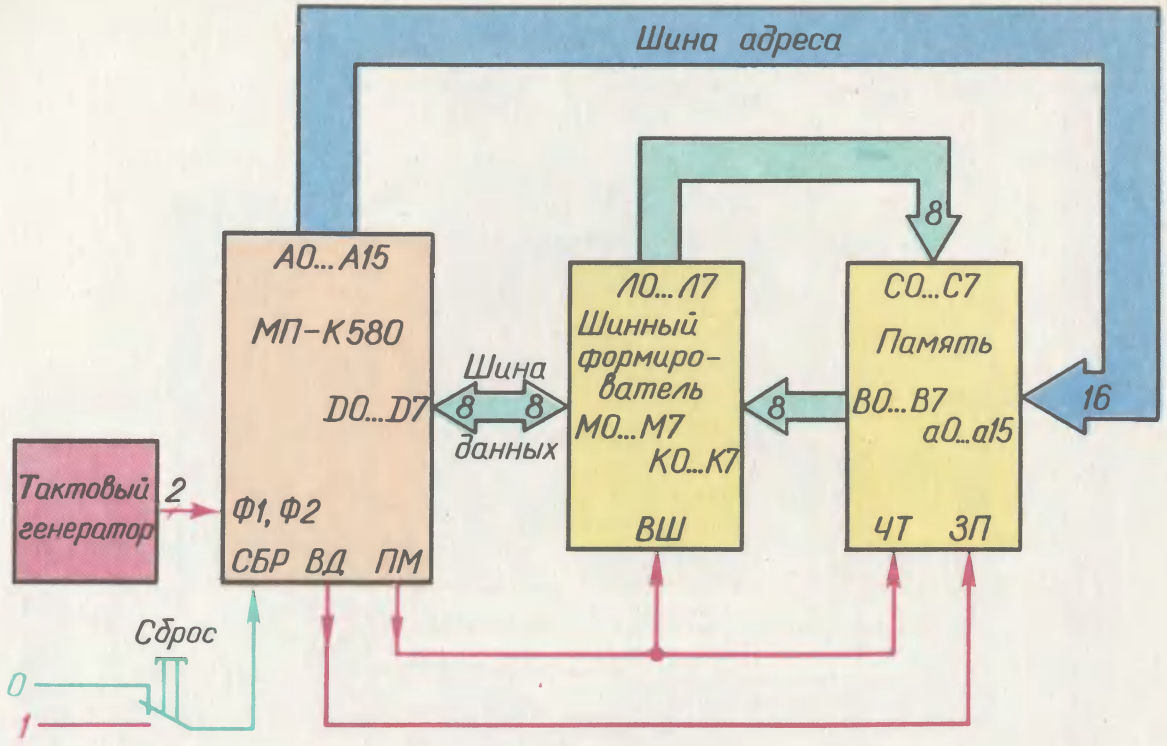


Рис. 34

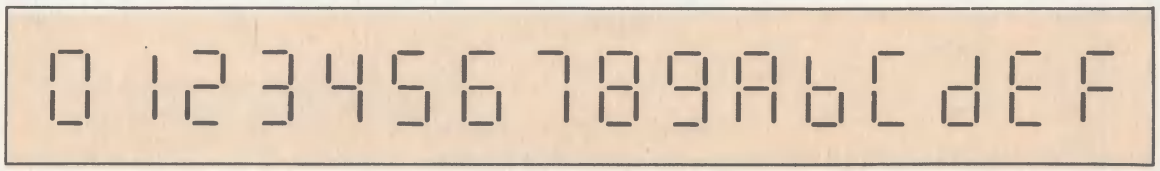
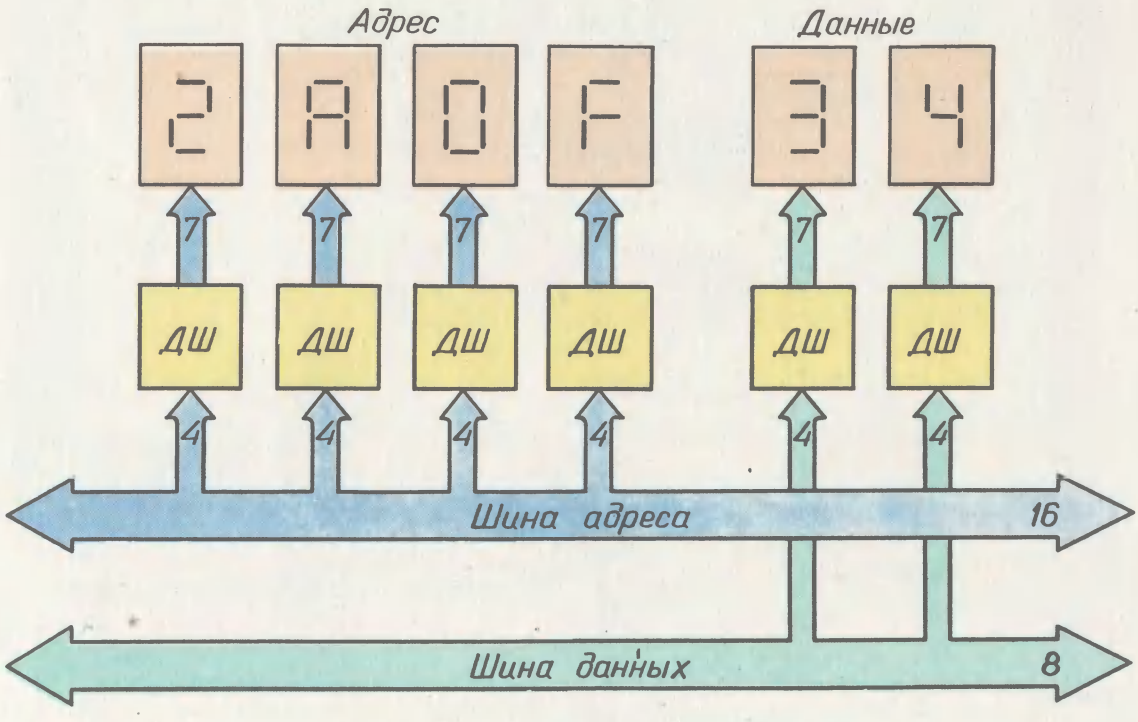


Рис. 35



ДШ – дешифратор

Рис. 36

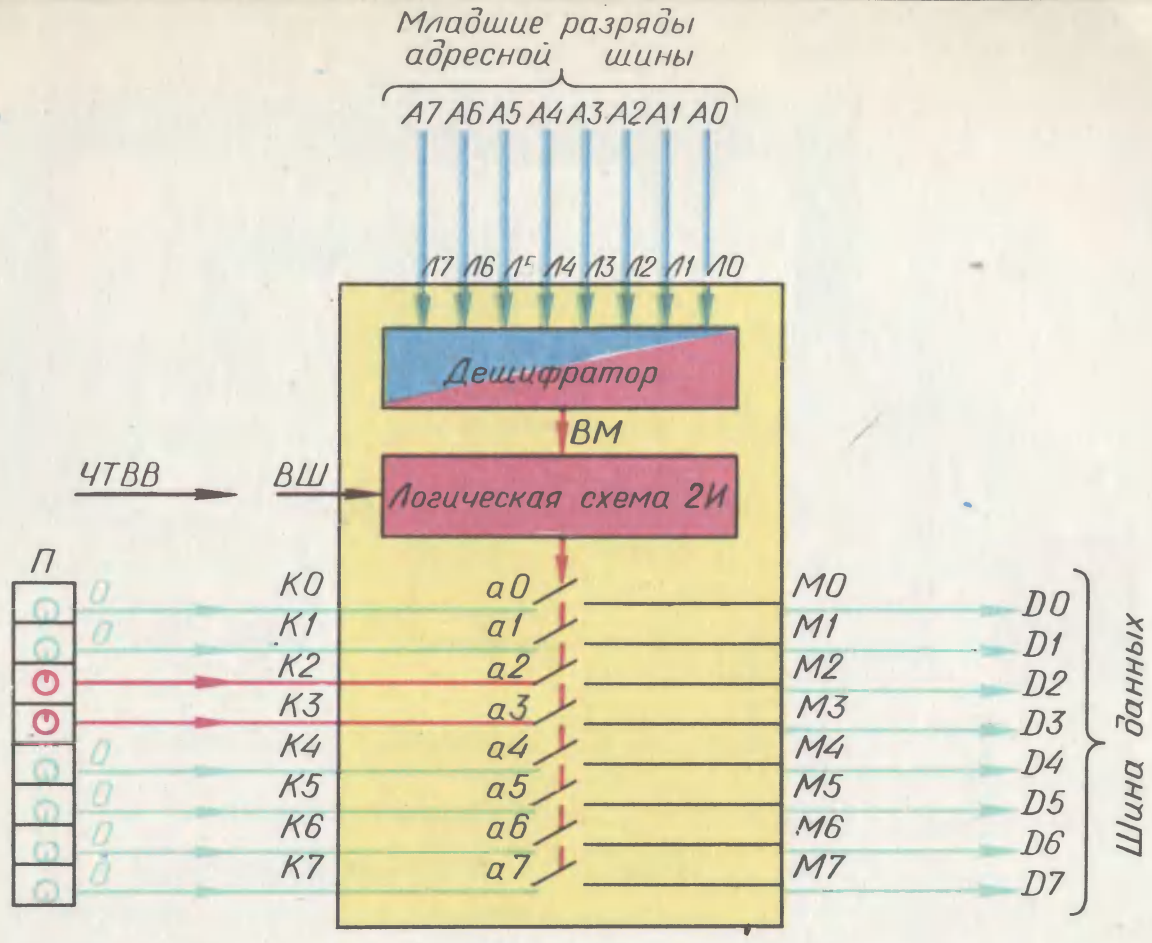

Младшие разряды адресной шины
A7 A6 A5 A4 A3 A2 A1 A0
Л7 Л6 Л5 Л4 Л3 Л2 Л1 Л0
Дешифратор
ВМ
ЧТВВ
ВШ
Логическая схема 2И
П
К0
К1
К2
К3
К4
К5
К6
К7
а0
а1
а2
а3
а4
а5
а6
а7
М0
М1
М2
М3
М4
М5
М6
М7
D0
D1
D2
D3
D4
D5
D6
D7
Шина данных

Рис. 37

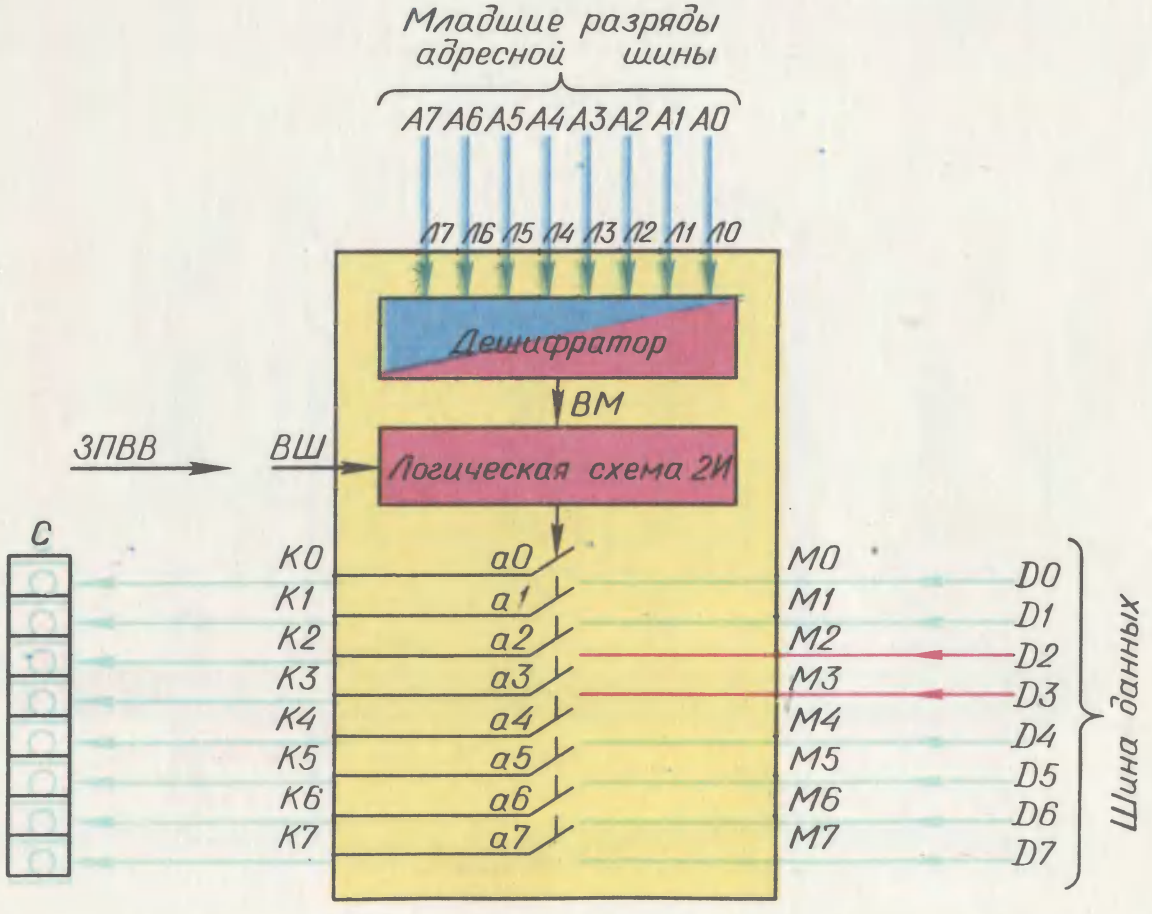

Младшие разряды адресной шины
A7 A6 A5 A4 A3 A2 A1 A0
Л7 Л6 Л5 Л4 Л3 Л2 Л1 Л0
Дешифратор
ВМ
ЗПВВ
ВШ
Логическая схема 2И
С
К0
К1
К2
К3
К4
К5
К6
К7
а0
а1
а2
а3
а4
а5
а6
а7
М0
М1
М2
М3
М4
М5
М6
М7
D0
D1
D2
D3
D4
D5
D6
D7
Шина данных

Рис. 38

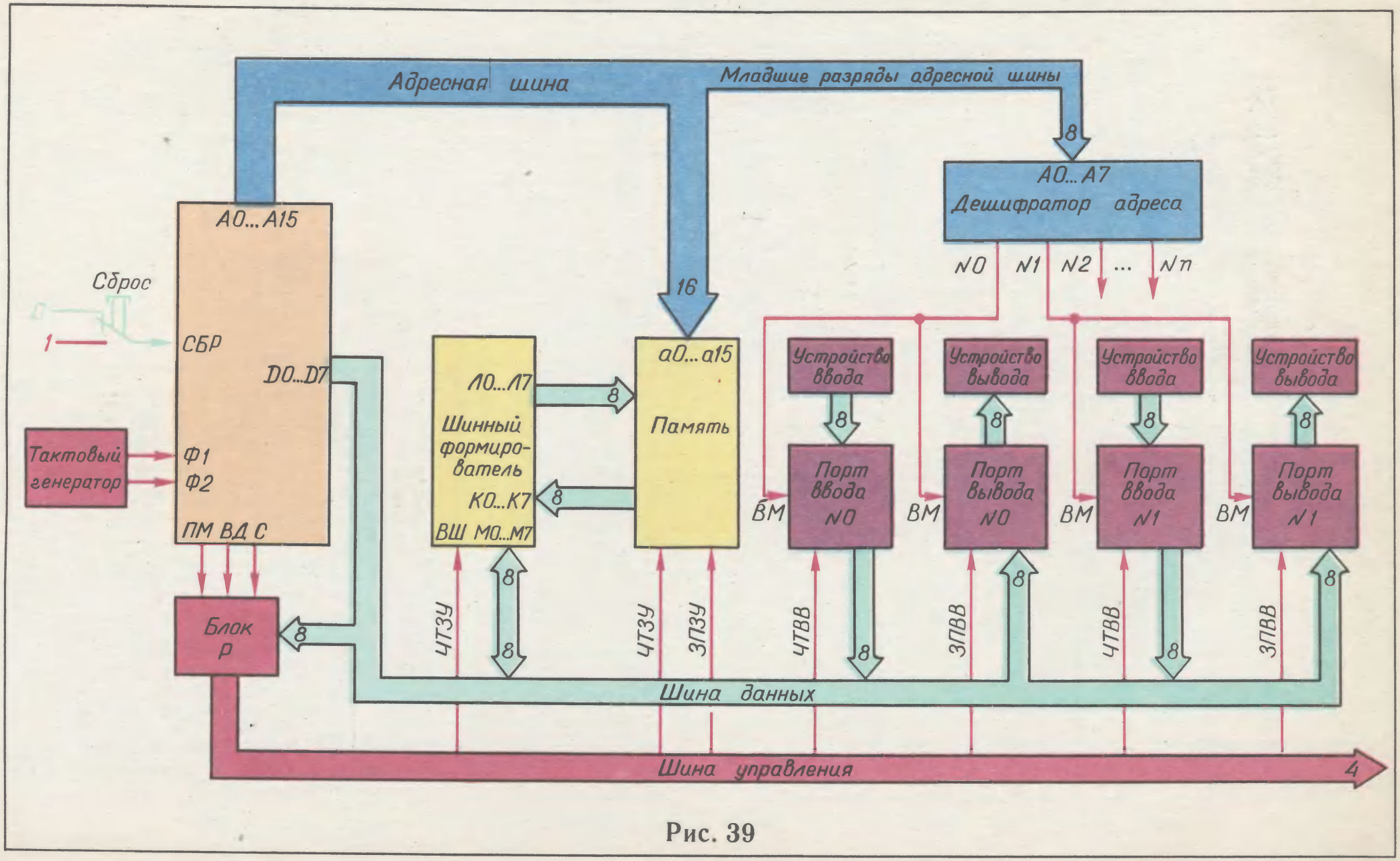

Адресная шина
Младшие разряды адресной шины
8
A0...A7
Дешифратор адреса
N0
N1
N2
...
Nn
16
A0...A15
Сброс
1
СБР
D0...D7
Тактовый генератор
Ф1
Ф2
ПМ ВД С
Блок р
8
Л0...Л7
Шинный формирователь
КО...К7
ВШ М0...М7
8
8
а0...а15
Память
ЧТЗУ
ЗПЗУ
Устройство ввода
Устройство вывода
Устройство ввода
Устройство вывода
8
8
8
8
ВМ
ВМ
ВМ
ВМ
Порт ввода N0
Порт вывода N0
Порт ввода N1
Порт вывода N1
ЧТВВ
ЗПВВ
ЧТВВ
ЗПВВ
8
8
8
8
Шина данных
Шина управления
4

Рис. 39

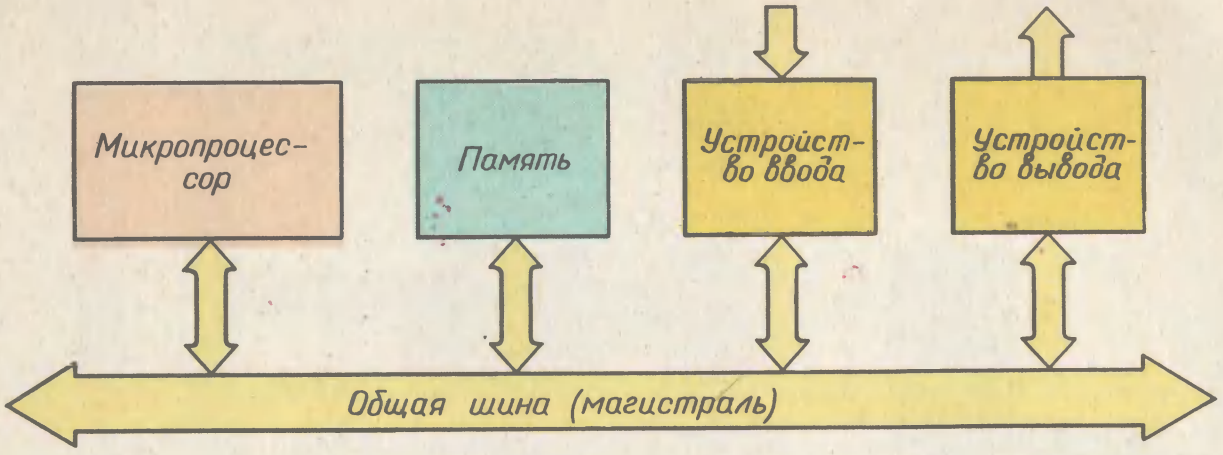


Рис. 40

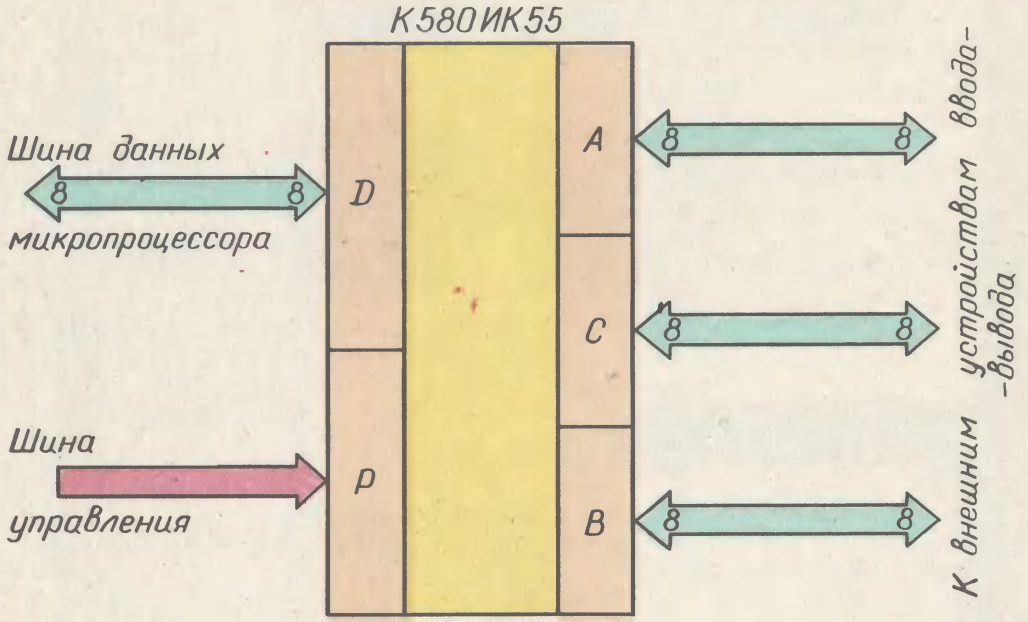


Рис. 41

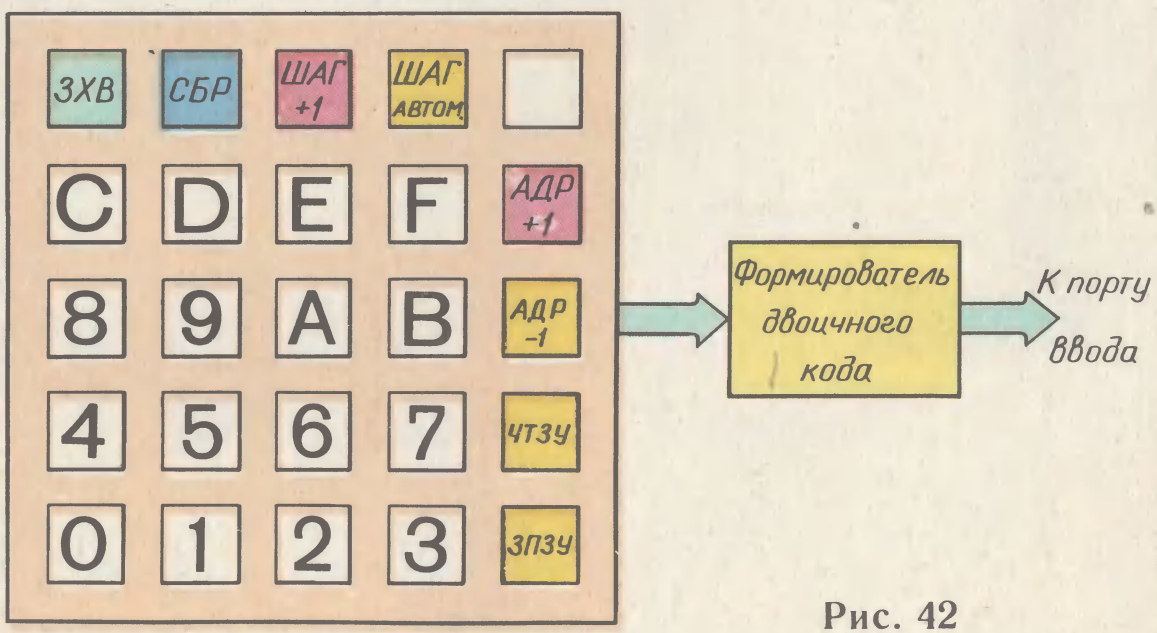


Рис. 42

Таким образом, комбинируя двоичным адресным кодом на адресной шине и сигналами ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ на шине управления, можно независимо друг от друга вводить и выводить данные из любой ячейки памяти и любого порта ввода и вывода.

Пользуясь схемой (рис. 39 вклейки), составим вариант программы сложения двух чисел, отличающийся тем, что числа не нужно предварительно размещать в памяти, а можно непосредственно ввести в регистры микропроцессора при помощи устройств ввода (переключателей), подключенных к портам ввода. С этой целью переключателями порта ввода № 0, изображенных на рис. 39 вклейки в виде устройства ввода, набираем первое слагаемое (число 3), а переключателями порта ввода № 1 — второе слагаемое (число 2). Программу будем составлять так же, как и прежде, пользуясь формой табл. 2.14 (с. 58) и выпиской из системы команд МП—К580, приведенной в табл. 2.13. (с. 56).

Приступая к составлению табл. 2.15, прежде всего заполняем ее колонки 3 и 4. В колонке 3 записываем начальную команду программы: «Переслать в регистр А данные (десятичную тройку) из порта ввода, адрес которого указан во 2-м байте команды». Так как первое слагаемое (тройку) мы набрали переключателями порта ввода № 0, то в первой строке колонки 4 записываем код операции этой команды, взятый из табл. 2.13, а во второй строке — адрес порта ввода № 0, выраженный в двоичной форме (00000000).

Ранее упоминалось, что ввод и вывод данных из портов МП—К580 осуществляет только через аккумулятор (регистр А). Поэтому нужно освободить регистр А от размещенного там числа (три), чтобы можно было переслать в регистр А второе слагаемое — двойку. Для этого пересылаем тройку из регистра А в регистр В. Пересылку выполняем при помощи однобайтной команды, имеющей код 01000111. Этот код записываем в колонке 4. Затем вводим второе число (двойку), набранное переключателями порта ввода № 1, в освободившийся регистр А. В колонке 3 записываем словесное название команды, а в колонке 4 — двоичные коды двух байтов этой команды. Код операции (11011011) будет прежним, а адрес изменится (00000001). Он будет соответствовать порту ввода № 1. После выполнения написанных команд числа 3 и 2 будут размещены в регистрах В и А. Теперь их нужно сложить. Для этого воспользуемся той же командой, что и в первом варианте программы (см. табл. 2.14). Результат сложения МП—К580 автоматически размещается в регистре А. Выведем этот результат в порт вывода № 0, пользуясь командой (см. табл. 2.13): «Переслать в порт, адрес которого указан во 2-м байте команды, данные из регистра А». Код этой операции 11010011, а двоичный адрес порта вывода — 00000000.

Все команды программы написаны. После этого в колонку 2 необходимо занести адреса всех ячеек памяти для размещения байтов команд, начиная с нулевого адреса, подобно тому, как это делалось при заполнении табл. 2.14. После написания этих адресов ячеек памяти, а также комментариев к каждому байту команды

Таблица 2.15

Вариант программы сложения чисел 3 и 2 при помощи МП—К580

Порядковый номер байтов программы	Двоичный код адреса ячейки памяти, в которой размещается байт команды или данных	Описание команды	Двоичный код байтов команды или данных	Комментарии
1	0000000000000000	Переслать в регистр А данные из порта ввода, адрес которого указан во 2-м байте команды	11011011	1-й байт (код операции)
2	0000000000000001		00000000	2-й байт (адрес порта ввода 0)
3	0000000000000010	Переслать данные из регистра А в регистр В	01000111	Однобайтная команда (коды операции и регистров)
4	0000000000000011	Переслать в регистр А данные из порта, адрес которого указан во 2-м байте команды	11011011	1-й байт (код операции)
5	0000000000000100		00000001	2-й байт (адрес порта ввода 1)
6	0000000000000101	Сложить содержимое регистров А и В	10000000	Однобайтная команда (коды операции и регистра)
7	0000000000000110	Переслать в порт, адрес которого указан во 2-м байте команды, данные из регистра А	11010011	1-й байт (код операции)
8	0000000000000111		00000000	2-й байт (адрес порта вывода 0)

программа будет составленной. Ее можно ввести в память, а затем включить МП—К580 и он выполнит эту программу, в результате чего на светодиодах, подключенных к порту вывода № 0 (светодиоды на рис. 39 вклейки изображены в виде устройства вывода), появится двоичное число 00000101, обозначающее десятичную пятерку (результат сложения тройки и двойки).

2.3. МОДУЛИ МИКРОЭВМ И ОСОБЕННОСТИ ИХ РАБОТЫ

Как было уже сказано, микропроцессор обязательно должен работать совместно с памятью и устройствами ввода—вывода данных. При этом комплект, состоящий из микропроцессора, памяти и устройств ввода—вывода, называют микроЭВМ, а составляющие части этого комплекта — **модулями.**

Комбинируя тип модулей и их количество, можно создавать разнообразные микроЭВМ, начиная от простейших контроллеров для управления режимом стиральных машин и кончая сложнейшими персональными компьютерами.

Несмотря на обилие видов микроЭВМ, почти все они имеют единую, типовую структуру, напоминающую структуру обычных ЭВМ. Такая структура изображена на рис. 40 вклейки. Она представляет собой схему соединений модулей микроЭВМ при помощи многопроводных шин адреса, данных и управления. Часто совокупность проводов этих шин выполняют в виде одной общей шины, называемой коммутирующим устройством или магистралью. К общей шине могут быть подсоединены и другие модули внешних устройств, например, **клавишный пульт,** телевизионный **дисплей,** печатающее устройство-принтер и др.

Рассмотрим особенности работы модулей микроЭВМ.

Микропроцессорный модуль

Промышленность выпускает более 40 видов различных микропроцессоров. Помимо четырехразрядного микропроцессора серии К584 и восьмиразрядного серии К580, освоен выпуск нескольких шестнадцатиразрядных микропроцессоров серий К1801, К1803, К1810 и К1883 с улучшенными параметрами, позволяющими создавать современные высокопроизводительные микроЭВМ.

Большой популярностью пользуются также микропроцессоры серии К145, специально разработанные для бытовых нужд (микрокалькуляторы, сервисная автомобильная электроника, управление стиральными машинами, телевизорами, магнитофонами и др.).

Работу микроЭВМ будем рассматривать с учетом того, что микропроцессорный модуль этой микроЭВМ собран на базе микропроцессора серии К580.

Модуль памяти

Память, описанная в разд. 2.2, получила название **оперативного запоминающего устройства** (сокращенно **ОЗУ**). Этот вид памяти является основным для микроЭВМ.

Выполняется ОЗУ на электронных элементах, чаще всего на триггерах, позволяющих быстро и оперативно запомнить 0 или 1 каждого бита данных путем установки триггера в сброшенное (0) или во взведенное (1) состояние. Несмотря на большую скорость и гибкость в работе, ОЗУ имеет один существенный недостаток: при выключении питания триггеры теряют свое устойчивое состояние и переходят в новое (случайное) произвольное положение, при котором данные, хранимые в ОЗУ, теряются. Чтобы избежать этого, применяют **постоянное запоминающее устройство (ПЗУ)**, выполненное так, что при выключении питания содержимое ячеек памяти не изменяется. Данные в ПЗУ записывают однократно при его изготовлении и в процессе эксплуатации ПЗУ (в отличие от ОЗУ) данные можно только считывать, а вносить новые, стирать и заменять другими нельзя.

Существуют разновидности ПЗУ, называемые **программируемым постоянным запоминающим устройством (ППЗУ)** и **репрограммируемым постоянным запоминающим устройством (РППЗУ).**

ППЗУ отличается от ПЗУ тем, что программируется не на заводе-изготовителе, а самим пользователем с помощью специального прибора—программатора. Причем программируется только один раз. После ввода программы данные, находящиеся в ППЗУ, уже нельзя изменить.

РППЗУ, называемое также стираемым ПЗУ, допускает многократное изменение данных. В качестве примера укажем на РППЗУ типа К573РФ1 с ультрафиолетовым стиранием. Если микросхему К573РФ1 извлечь из микроЭВМ и подвергнуть ультрафиолетовому облучению, то все ячейки памяти К573РФ1 станут выдавать на выходе единицы, и это приводит к тому, что содержащиеся там данные в виде комбинации нулей и единиц будут стерты. Устанавливая после облучения К573РФ1 в программатор и задавая адрес и данные, можно снова запрограммировать К573РФ1 по своему усмотрению.

Для сохранения данных при выключении питания, кроме ПЗУ, применяют также ОЗУ, потребляющее мало энергии. Если параллельно проводам питания такого ОЗУ подключить батарейку, то ее энергии будет достаточно для того, чтобы триггеры памяти ОЗУ сохраняли свое состояние при отключении основного питания в течение нескольких часов и даже дней. Такое ОЗУ называют **энергонезависимым** и иногда применяют при реализации памяти микроЭВМ.

Во время работы микроЭВМ часто используют разновидность ОЗУ, называемую **сверхоперативным запоминающим устройством (СОЗУ)**, которое позволяет кратковременно запомнить и в нужный момент выдать цифровые данные. В качестве СОЗУ употребляют регистры РОН микропроцессора или регистры, установленные в других модулях микроЭВМ.

Чтобы закрепить понятия СОЗУ, ОЗУ и ПЗУ, рассмотрим несколько примеров аналогичного использования запоминающих

устройств в повседневной жизни человека. Выполняя, например, умножение в столбик, мы часто произносим: «пятью пять — двадцать пять. Пять пишем, два в уме». Временное, сверхоперативное запоминание двойки в уме и является примером работы СОЗУ. Регистр микроЭВМ, как и человек, временно запоминает какой-то промежуточный итог вычислений.

Как известно, феноменальной памятью обладают немногие. Чаще всего, желая что-то запомнить, мы пользуемся записной книжкой, занося туда адреса, фамилии, номера телефонов и другие данные, а затем, заглядывая в нее, вычитываем нужные сведения, заменяем их, исправляем и записываем новые. Такой вид оперативного запоминающего устройства — записная книжка — соответствует машинному ОЗУ. И наконец, мы часто пользуемся постоянной памятью (ПЗУ) в виде книг, журналов, таблиц лотерейных выигрышей и др. Сведения, имеющиеся в этих источниках данных, можно только читать. Нельзя исправлять или вносить новые данные в книгу или таблицу выигрышей аналогично тому, как нельзя изменять данные в ПЗУ.

Несколько слов о размерах памяти, вернее о количестве запоминающих ячеек, содержащихся в ней. В простейших микроЭВМ применяют ОЗУ, позволяющее разместить около тысячи байтов данных, что соответствует 1 килобайту (1 кбайт). Точнее 1 кбайт = = 1024 байт (округляя, принято говорить, что килобайт равен тысяче байтов).

Современные микроЭВМ снабжаются ОЗУ емкостью в десятки и сотни килобайтов. Размеры ПЗУ бывают примерно в 2—8 раз меньше ОЗУ.

СОЗУ в виде регистра содержит всего 8 или 16 элементов ячейки памяти, где запоминается 1 или 2 байта данных.

Все рассмотренные виды памяти находятся внутри микроЭВМ и предназначены для размещения команд и данных выполняемой программы. Отметим, что решение некоторых сложных задач требует выполнения сотен тысяч, а то и миллионов команд. Разместить такое количество команд в ОЗУ или ПЗУ не представляется возможным. Поэтому их помещают во **внешние запоминающие устройства (ВЗУ)**, не входящие в состав комплекта микроЭВМ, а потом частями, по мере необходимости, переписывают в ОЗУ для выполнения. ВЗУ реализуют с помощью различных технических средств. Чаще всего для этой цели используют магнитную ленту, магнитные диски и бумажную ленту. Намагниченный участок магнитной ленты или пробитое отверстие в бумажной ленте соответствует единице (1), а размагниченный участок или отсутствие отверстия — нулю (0). В таких ВЗУ можно размещать миллионы байт данных (1 млн. байт = 1 мегабайт = = 1 Мбайт).

Недостатком ВЗУ является малая скорость работы. Приходится затрачивать много времени (несколько секунд, а то и минут) для того, чтобы отыскать на ленте тот участок, где записаны нужные данные.

Последние годы стало весьма популярным в качестве ВЗУ использовать бытовые магнитофоны. С этой целью к магнитофону добавляют специальную электронную приставку, подключаемую к шинам адреса, данных и управления микроЭВМ, что позволяет записывать нужные данные на магнитофонную ленту или считывать их с ленты в ОЗУ. На одном миллиметре магнитофонной ленты удается записать 60—80 бит данных, а в магнитофонной кассете разместить примерно 200—300 кбайт данных. Часто применяют также дисковод — устройство, состоящее из вращающегося магнитного диска и электромагнитной головки, считывающей намагниченные участки дорожек, нанесенных на диске.

Модуль ввода—вывода

Существует большое количество типов **устройств ввода— вывода данных,** применяемых в микроЭВМ. Все они заимствованы из комплекта технических средств больших ЭВМ. Своих устройств, присущих только микроЭВМ, пока нет.

Использование того или иного типа устройств ввода—вывода зависит от задач, поставленных перед микроЭВМ. Эти устройства в технической литературе часто называют **периферийными** или **внешними устройствами.**

В простейших случаях работы микроЭВМ устройства ввода—вывода подключаются к шинам через порты, а в более сложных — с помощью специальных устройств, называемых **интерфейсом.** В узком смысле слова под интерфейсом понимают устройство сопряжения (согласования) модулей микроЭВМ, а в более широком толковании — совокупность электрических, механических и программных средств, предназначенных для организации обмена данными между модулями микроЭВМ.

Поясним сказанное об интерфейсе на таком примере. Приобретая бытовой электрический прибор (электробритву, магнитофон, электроутюг), мы прежде всего интересуемся его электрическим согласованием с питающей сетью. Если в сети напряжение 127 В, а прибор рассчитан на питание от сети 220 В, то при его включении в сеть он работать не будет. Еще хуже, если рабочее напряжение прибора окажется меньше напряжения в сети (прибор может выйти из строя). Для нормальной работы требуется электрическое согласование бытового прибора и питающей сети. Если такого согласования не будет, то его можно достигнуть, установив повышающий (понижающий) трансформатор, который будет выполнять роль интерфейса между прибором и питающей сетью. Кроме электрического согласования, должна выполняться и механическая совместимость. Например, штепсельная вилка прибора и розетка питающей сети должны соответствовать друг другу конструктивно.

При использовании сложных периферийных устройств, имеющих свою программу работы, свой язык программирования и т. д., интерфейс выполняет роль переводчика с одного языка програм-

мирования на другой, согласовывает форматы данных, осуществляет синхронизацию обмена данными и выполняет целый ряд других программных согласований работы периферийного устройства с модулями микроЭВМ. Помимо этого, интерфейс выполняет функции портов — производит дешифрацию адреса, обеспечивает двунаправленность работы шины данных и др.

Если периферийное устройство выполняет многочисленные разнообразные и очень сложные операции, то с модулями микроЭВМ его согласовывают с помощью усложненного интерфейса, называемого **программируемым интерфейсом.** Промышленность серийно выпускает такой интерфейс типа К580ИК55, допускающий совмещенную работу с микроЭВМ, собранной на МП—К580. Этот интерфейс (рис. 41 вклейки) имеет три канала (A, B и C) для подключения внешних периферийных устройств и один канал D для подключения шины данных микроЭВМ. Подавая на управляющий вход P двоичный код, можно по-разному запрограммировать работу К580ИК55, а именно: в зависимости от кода каналы A, B и C могут работать каждый порознь или все вместе на ввод или вывод данных. Допускается также работа двух каналов (A и B) на ввод или вывод, а третьего (C) для включения управляющих сигналов.

К580ИК55 считается универсальным интерфейсом широкого применения, поскольку его можно сочетать почти со всеми имеющимися периферийными устройствами, получая при этом программным путем около ста различных схемных комбинаций обмена данными между устройствами ввода—вывода и микроЭВМ.

Рассмотрим некоторые устройства ввода—вывода данных.

1. **Устройства ввода.** В простейших случаях для ввода данных используют известные нам переключатели в виде тумблеров, клавиш, кнопок или движковых выключателей, обеспечивающих ввод единицы или нуля двоичного числа, в зависимости от положения ручки тумблера, клавиши, кнопки или движка. Достоинством такого устройства ввода является простота конструкции и то обстоятельство, что ввод осуществляется непосредственно в двоичном машинном коде.

Если программа написана в шестнадцатеричном виде, то для удобства ее ввода в микроЭВМ пользуются шестнадцатеричной клавиатурой (рис. 42 вклейки), содержащей 16 клавиш, позволяющих вводить шестнадцатеричные цифры от 0 до F. При нажатии любой клавиши на выходе формирователя двоичного кода появляется четырехразрядный двоичный код, соответствующий цифре, указанной на клавише. Этот код поступает в регистр порта ввода и далее по шине данных в микроЭВМ. Часто шестнадцатеричную клавиатуру снабжают рядом дополнительных клавиш, позволяющих управлять режимом работы микроЭВМ (клавиши ЗХВ, СБР, АДР + 1, АДР — 1 и др. на рис. 42 вклейки).

В более совершенных микроЭВМ, использующих телевизионный экран для отображения данных, применяется **клавиатурный пульт,** имеющий десятки клавиш. Кроме букв и цифр, клавиатура

имеет клавиши со всякого рода знаками, удобными для программирования, редактирования текста и управления режимом работы микроЭВМ. Как и в шестнадцатеричной клавиатуре, нажатие каждой клавиши преобразуется (с помощью сложной электронной схемы или специально составленной программы) в двоичный код, соответствующий символу, обозначенному на клавише. Этот код поступает в микроЭВМ для исполнения и одновременно отображается на экране телевизора для контроля. Серийно выпускаемый промышленностью клавиатурный пульт типа МС7004 имеет 96 клавиш.

При использовании микроЭВМ в качестве контроллера, управляющего токарным станком, стиральной машиной или светофором, устройство ввода выполняется в виде преобразователя, называемого датчиком, который превращает данные соответственно о положении резца, температуре воды или количестве автомашин в цифровой двоичный код. Этот код поступает в микроЭВМ. Часто между датчиком и микроЭВМ устанавливают интерфейс.

2. **Устройства вывода.** Наиболее простыми и часто применяемыми устройствами вывода являются знакомые нам светодиоды, по свечению которых можно определить логический уровень (1 или 0) на каждом проводе шины данных и адреса. Светодиоды бывают красного, зеленого и желтого свечения. Для индикации обычно используют светодиоды красного цвета, так как другие цвета трудно различимы при дневном ярком свете.

Если микроЭВМ имеет шестнадцатеричную клавиатуру ввода, то вывод осуществляется при помощи семисегментных индикаторов. Все сегменты могут включаться каждый отдельно или несколько в нужном сочетании. Комбинацией сегментов могут быть отображены все цифры от 0 до 9, а также некоторые буквы. На рис. 35 вклейки показано, как с помощью семисегментного индикатора отобразить цифры шестнадцатеричной системы счисления.

Обычно в микроЭВМ устанавливают шесть семисегментных индикаторов (рис. 36 вклейки): четыре из них подключают к адресной шине и два — к шине данных. Подключение выполняют через специальные дешифраторы, преобразующие двоичный код **тетрады** (4 разряда) шины в семисегментный код засветки индикатора.

Из всех устройств вывода микроЭВМ самым распространенным и совершенным является телевизионный экран, на котором отображается текст, состоящий из слов и цифр, а также графики и рисунки в черно-белом или цветном изображении. Такое устройство вывода (отображения) называют **дисплеем.** Помимо электронно-лучевой трубки, применяемой в телевизионном дисплее, его выполняют также на специальных газоразрядных панелях, жидкокристаллических индикаторах, электрохромных и других элементах.

Для вывода алфавитно-цифровых данных на печать применяют устройство, называемое **принтер.** Иногда принтер именуют **алфавитно-цифровым печатающим устройством** (сокращенно **АЦПУ**).

Принтер выполняет построчную печать на бумаге от 32 до 128 символов в строке. Скорость печатания составляет десятки символов в секунду.

При использовании микроЭВМ для автоматизации проектных работ в качестве выводного устройства применяют прибор, называемый **графопостроителем.** Он представляет собой автоматическую чертежную доску, управляемую двоичным кодом микроЭВМ.

Под воздействием кодовых сигналов включаются управляющие двигатели, передвигающие рапидограф — чертежный инструмент, вычерчивающий на бумаге линии чертежа, графики или схемы.

При работе микроЭВМ в качестве контроллера для управления станками или бытовыми приборами выводное устройство выполняется в виде исполнительного механизма, преобразующего выходной двоичный код микроЭВМ в электрическое напряжение, включающее реле, двигатель или иное электромеханическое устройство, обеспечивающее нужное положение управляющего органа (резца, водяного вентиля, регулятора громкости, выключателя питания и др.).

3. **Устройства ввода—вывода.** К универсальным устройствам, позволяющим во время работы микроЭВМ вводить и выводить данные, относится сочетание клавишного пульта и телевизионного дисплея. С помощью клавиш данные вводятся в микроЭВМ, а на экране телевизора осуществляется контроль данных. Такое устройство в литературе иногда называют дисплеем, хотя, строго говоря, дисплей служит только для отображения данных. Поэтому комплект, состоящий из клавишного пульта и дисплея, правильнее называть **экранным пультом** или **видеотерминалом.**

К универсальным устройствам ввода—вывода относится также телетайп, который был первоначально разработан для телеграфных сообщений. Он имеет буквенно-цифровую клавиатуру и устройство, печатающее текст на бумажной ленте. В комплекте с микроЭВМ телетайп применяется редко. Чаще используется обыкновенная электрическая пишущая машинка типа «Консул». При нажатии клавиши такой машинки образуется двоичный код, который через преобразователь поступает в микроЭВМ для исполнения. И наоборот, двоичный код микроЭВМ с помощью преобразователя превращается в код, управляющий механизмом печатания пишущей машинки, и на бумаге воспроизводится текст, выдаваемый микроЭВМ.

В последние годы во многих странах уделяется большое внимание разработке устройств ввода — вывода данных на человеческом языке (в речевой форме), а также путем обмена графическими данными — чертежами, схемами и рисунками. Общение с микроЭВМ, наделенной таким устройством, становится очень простым и естественным. Уже сейчас имеются разнообразные говорящие устройства в виде микрокалькуляторов, бытовых и измерительных приборов, игрушек, электронных игр, читающих машин и речераспознающих автоматов. Правда, успехи в этих разработках пока весьма скромные. Эти устройства еще очень громоздкие и дорогие.

Они не выпускаются промышленностью серийно. Однако микропроцессорная техника развивается так стремительно и многолико, что нельзя определенно предсказать ее дальнейший ход. Возможно в самом ближайшем будущем появятся простые, дешевые синтезаторы речи и понимающие человеческий голос компьютеры.

Обмен данными в микроЭВМ

В зависимости от вида взаимодействия микропроцессора, памяти и устройства ввода — вывода различают три способа обмена данными: программный, с прерыванием программы, с помощью прямого доступа к памяти.

Программный способ рассмотрен в разделе 2.2. Для обмена данными по этому способу используются специальные управляющие сигналы ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ и различные команды программы для реализации четырех режимов работы микропроцессора: чтение данных из памяти, запись данных в память, чтение (получение) данных из устройств ввода и запись (выдача) данных в устройство вывода.

Второй способ обмена данными (с прерыванием программы) может возникнуть при эксплуатации микроЭВМ в условиях повышенного риска (медицина, авиация, атомная энергетика), когда требуется немедленная передача данных в микропроцессор для их внеочередной обработки и выработки управляющих сигналов, нормализующих обстановку. Выполнение режима прерывания происходит следующим образом. Внешнее устройство, зафиксировав аварийное состояние объекта, посылает по шине управления на вывод ЗПР микропроцессора сигнал (1) запроса на прерывание. Получив этот сигнал, микропроцессор заканчивает выполнение текущей команды и пересылает в специально отведенное место в памяти, называемое **стековой памятью,** данные о состоянии всех своих задействованных регистров, чтобы потом, отработав программу внешнего устройства, можно было снова вернуть эти данные в регистры и продолжить выполнение основной, прерванной программы. Выполнив все свои действия, микропроцессор выдает на вывод РПР сигнал (1) разрешения на обслуживание внешнего устройства, разрешая этим прерывания.

Различают три вида прерывания: простое, векторное и приоритетное.

Простое прерывание включает единственное устройство ввода и одну подпрограмму, обслуживающую это прерывание. Например, внешнее устройство в виде датчика температуры, установленное на электродвигателе, зафиксировало перегрев обмотки. В этом случае требуется, чтобы микропроцессор прервал выполнение своей основной программы и включил служебную программу сигнала тревоги, извещающего обслуживающий персонал.

Векторное прерывание позволяет включать не одну, а несколько подпрограмм прерывания. С этой целью, помимо сигнала РПР, внешнее устройство посылает по трем проводам шины данных

трехразрядный двоичный код (называемый вектором), позволяющий микропроцессору включить одну из восьми подпрограмм прерывания, расположенных в памяти микроЭВМ.

При наличии нескольких устройств ввода, способных вызвать прерывание, может возникнуть ситуация, при которой во время обслуживания запроса прерывания от одного устройства, поступает запрос прерывания от другого устройства. В таком случае порядок работы микропроцессора решается путем системы приоритетного (первоочередного) прерывания. Распределение приоритетов осуществляется разработчиком или пользователем микроЭВМ, путем формирования специальных кодовых сигналов.

Во время запроса прерывания микропроцессор может выполнять настолько важную и срочную программу, что ее не следует прерывать. Для такого случая в системе команд предусмотрены специальные команды: «Запрещение прерывания» и «Разрешение прерывания». Если пользователь, составляя программу, включит в нужное место программы команду запрета прерывания, то микропроцессор будет игнорировать запрос на прерывание, поступающий от внешнего устройства на вывод РПР до тех пор, пока в программе не будет обозначена команда, разрешающая прерывание.

Третий способ обмена данными осуществляется с помощью режима **прямого доступа к памяти (ПДП)**, при котором данные между внешним устройством и памятью обмениваются напрямую, минуя микропроцессор. Аналогично способу обмена с прерыванием программы, внешнее устройство посылает по шине управления на вывод ЗХ микропроцессора сигнал (1) запроса захвата шин. Получив сигнал ЗХ, микропроцессор заканчивает выполнение текущей команды и посылает на вывод ПЗХ подтверждение, разрешающее внешнему устройству использовать шины адреса и данных, а сам отключается от этих шин. Причем (в отличие от режима прерывания) микропроцессор не очищает свои регистры, не засылает их данные в стековую память, а просто приостанавливает свою работу, сохраняя во всех регистрах имеющиеся данные. Как только обмен ПДП закончится и на выводе ЗХ микропроцессора будет снят сигнал запроса захвата от внешнего устройства, микропроцессор подключится к шинам адреса и данных и продолжит выполнение своей работы.

Более подробно о работе микропроцессора и микроЭВМ читатель может узнать из литературы [9, 11, 12, 16, 17, 37, 38, 47, 49, 63].

Глава 3. КАК СОБРАТЬ, НАЛАДИТЬ И УСОВЕРШЕНСТВОВАТЬ МИКРОЭВМ

3.1. ВЫБОР СХЕМЫ И КОНСТРУКЦИИ ОСНОВНЫХ МОДУЛЕЙ

Схема и конструкция самодельной микроЭВМ всецело определяется наличием деталей для ее сборки, а также знанием и умением того, кто ее собирает.

МикроЭВМ, изготавливаемая в радиокружке или в домашних условиях, должна удовлетворять следующим требованиям:

1) иметь простую схему и конструкцию, максимально доступную для повторения;

2) содержать наименьшее количество деталей, особенно дефицитных;

3) допускать замену отсутствующих деталей другими, имеющимися в наличии;

4) позволять налаживание при помощи простых средств и приборов;

5) наглядно демонстрировать основные режимы работы;

6) производить запись данных с пульта управления в любую ячейку памяти;

7) обеспечивать чтение данных, содержащихся в любой ячейке памяти;

8) разрешать пошаговый и автоматический режимы выполнения команд учебной программы;

9) совершать наглядный обмен данными с внешними устройствами;

10) осуществлять постепенную модернизацию путем замены и добавления новых модулей.

Ниже приводится описание одного из возможных вариантов самодельной сборки микроЭВМ, в какой-то степени отвечающей перечисленным требованиям. В литературе можно встретить и другие варианты, которые коротко будут рассмотрены нами.

МикроЭВМ целесообразно собирать в два этапа. Вначале собирают простейшую учебную микроЭВМ. После ее налаживания, изучения и отработки на ней учебных программ переходят ко второму этапу, который заключается в сборке дополнительных модулей, позволяющих превратить учебную микроЭВМ в персональный компьютер.

Для выполнения требований 5, 6, 7, 8 и 9, обеспечивающих получение простейшей микроЭВМ (которая допускает изучение основных режимов работы и отработку написанных программ), достаточно собрать три модуля, очерченных на рис. 43 вклейки пунктиром, а потом постепенно добавлять другие модули, совер-

шенствуя этим структуру микроЭВМ в соответствии с требованием 10.

Рассмотрим подробно с помощью принципиальных электрических схем, как устроены и как работают основные модули, а также соединяющая их общая шина.

Изучение модулей при помощи принципиальных электрических схем крайне необходимо для сознательного выполнения электрического монтажа, а главное — для приобретения знаний по налаживанию модулей и их дальнейшему совершенствованию.

Рассмотрение начнем с наиболее простой части микроЭВМ — ее общей шины.

Общая шина

Общая шина — это жгут проводов, состоящий из 16 проводов адресной шины, 8 проводов шины данных, 10 проводов шины управления, 6 проводов шины питания и 5 резервных проводов.

Общая шина с модулями микроЭВМ соединяется при помощи многоконтактных разъемов. На модулях устанавливаются штыревые части разъема, а на общей шине — гнездовые. Для соединения используется разъем типа ГРПМ1—45, имеющий 45 контактов. Допускается применение разъемов и другого типа, имеющих не менее 40 контактов.

При развитии структуры микроЭВМ в один из разъемов общей шины может быть подключен размножитель (рис 44 вклейки), позволяющий включение дополнительных модулей.

Гнездовые контакты разъемов, к которым подключаются провода всех шин, соединяются на всех разъемах (кроме разъема XS5 блока питания) параллельно друг с другом. Например, провода адресной шины ША0—ША15 подключаются к контактам А13 А20 и Б14—Б21, установленным на каждом разъеме общей шины, а провода шины данных ШД0—ШД7 подводятся к контактам А9—А12 и Б10—Б13. Свободными от подключения остаются лишь резервные контакты разъемов А3, А6, А21, Б3 и Б22, к которым подсоединяются индивидуальные (для данного модуля) электрические цепи, не вошедшие в состав шин.

Принципиальная электрическая схема общей шины приведена на рис. 11 (с. 79). В связи с тем, что соединения на всех разъемах (кроме разъема блока питания) выполнены одинаково, то на рис. 11 для упрощения показана разводка проводов лишь для трех разъемов: XS1, XS5 и XS7.

Перед началом монтажа читатель должен начертить всю схему соединений общей шины для всех разъемов, чтобы в процессе выполнения монтажа можно было с помощью красного карандаша отмечать на схеме каждое выполненное соединение. Такой способ позволяет устранить ошибки и пропуски при монтаже.

Разъем XS5, служащий для подключения блока питания, лучше выбрать другого типа (например, РП10—7 или РПМ8—8), обеспечивающего более надежное соединение штырей с гнездами.

Это особенно важно для цепей питания +5 В и 0 В, где протекает ток значительной силы. Если же подключение блока питания будет выполняться при помощи разъема ГРПМ1—45, то для коммутации цепей +5 В и 0 В рекомендуется соединить параллельно несколько контактов разъема, как это показано на рис. 11. Чтобы предотвратить возможность ошибочного включения в разъем XS5 блока питания, одного из разъемов (XP1—XP3) модулей микроЭВМ, необходимо несколько гнезд разъема XS5 заглушить при помощи пластмассовых или металлических вставок, а на штыревом разъеме XP5 блока питания изъять соответствующее количество штырей, расположенных напротив этих гнезд. Такой прием позволит включать в гнездовой разъем XS5 только штыревой разъем XP5. Включение других штыревых разъемов станет невозможным из-за установленных в гнезда заглушек.

Для более надежного соединения цепи питания +5 В и 0 В подключаются на всех разъемах при помощи двух параллельно соединяемых контактов. Напряжение +5 В подводится к контактам А1 и Б23, а напряжение 0 В — к контактам Б1 и А22.

Длина проводов общей шины не должна превышать одного метра. Провода питания +5 В и 0 В должны иметь сечение не менее 0,5 мм^2 Сечение остальных проводов не лимитируется.

Модуль микропроцессора

Как видно из структурной схемы рис. 45 вклейки, этот модуль состоит из микропроцессора МП, тактового генератора ТГ, шинных формирователей ШФА и ШФД, устройства формирования управляющих сигналов ФУ и многоконтактного штыревого разъема XP, с помощью которого модуль микропроцессора подключается к разъему общей шины.

Основное назначение микропроцессорного модуля — обеспечение соединений выводов микропроцессора с общей шиной. Микропроцессор устроен так, что его выходные выводы (адресные, данных и управления) нельзя непосредственно подключать к проводам общей шины, так как на всех выходах микропроцессора установлены маломощные транзисторы, которые не могут обеспечить нужную силу тока при значительной нагрузке, подключенной к общей шине. Согласно заводским данным, каждый выходной вывод МП—К580 можно подключать только к одному входному выводу микросхем типа ТТЛ (например, серии 155 или 133). Если же требуется подключение к нескольким входам ТТЛ, то приходится дополнительно устанавливать усилитель мощности, что и делается в микропроцессорном модуле. Помимо этого, модуль микропроцессора осуществляет такие действия:

1) создает тактовые сигналы Ф1 и Ф2 для микропроцессора;
2) выдает управляющие сигналы ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ;
3) отключает выводы адреса, данных и управления микропроцессора от общей шины при запросе внешним устройством использования (захвата) общей шины;

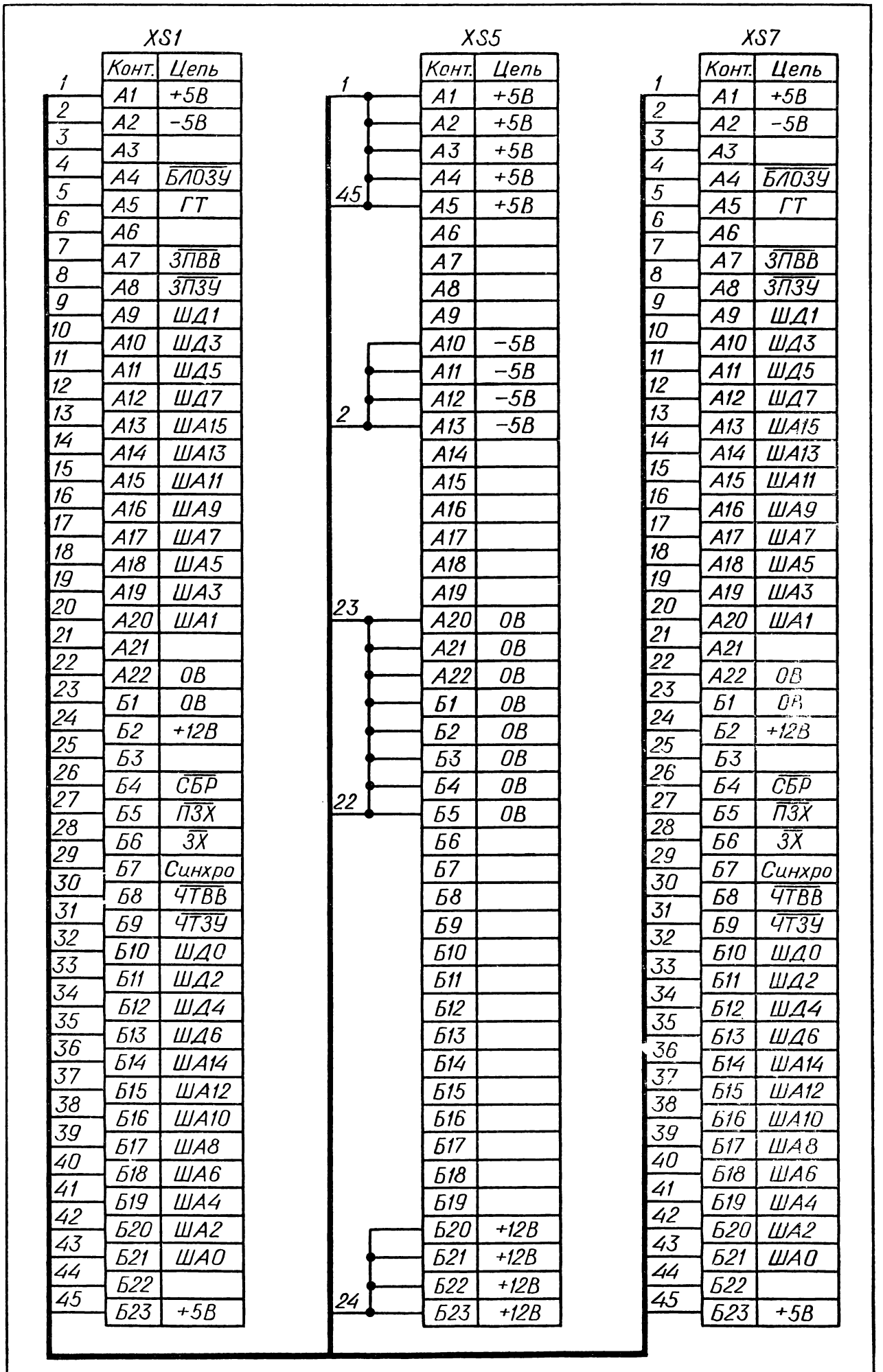

XS1
XS5
XS7
Конт.
Цепь
A1 +5В
A2 −5В
A3
A4 БЛОЗУ
A5 ГТ
A6
A7 ЗПВВ
A8 ЗПЗУ
A9 ШД1
A10 ШД3
A11 ШД5
A12 ШД7
A13 ША15
A14 ША13
A15 ША11
A16 ША9
A17 ША7
A18 ША5
A19 ША3
A20 ША1
A21
A22 ОВ
Б1 ОВ
Б2 +12В
Б3
Б4 СБР
Б5 ПЗХ
Б6 ЗХ
Б7 Синхро
Б8 ЧТВВ
Б9 ЧТЗУ
Б10 ШД0
Б11 ШД2
Б12 ШД4
Б13 ШД6
Б14 ША14
Б15 ША12
Б16 ША10
Б17 ША8
Б18 ША6
Б19 ША4
Б20 ША2
Б21 ША0
Б22
Б23 +5В
1
45
2
23
22
24

Рис. 11

4) обеспечивает двунаправленное действие шины данных при выполнении микропроцессором режима чтение—запись;

5) осуществляет включение в нужный момент времени сигналов ГТ и СБР, поступающих от внешних устройств на одноименные выводы микропроцессора.

В соответствии со структурной схемой рис. 45 вклейки на рис. 12 приведена принципиальная электрическая схема модуля микропроцессора. В этой схеме функции шинных формирователей выполняют микросхемы DD5—DD12 типа К589АП16, работающие по принципу шинных формирователей, описанных в разд. 2.3, с той лишь разницей, что каждая микросхема К589АП16 производит переключение четырех, а не восьми проводов шины.

К589АП16 (рис. 46 вклейки) имеет три группы выводов: А1—А4, В1—В4 и С1—С4, а также два управляющих вывода: ВШ и ВМ. Если на оба управляющих вывода подать уровень логического нуля, то выводы А1—А4 соединяются внутри схемы формирователя с выводами В1—В4 (рис. 46, *б* вклейки). Таким образом, подав данные на выводы А1—А4, их можно получить на выводах В1—В4. Они будут усиленными по мощности, так как внутри схемы К589АП16 установлены усилители (на рис. 46 вклейки их не показано). Если на выводе ВМ оставить нулевой уровень, а на вывод ВШ подать уровень логической единицы, то выводы В1—В4 соединяются с выводами С1—С4 (рис. 46, *в* вклейки). В этом случае данные с выводов В1—В4 будут передаваться через усилители на выводы С1—С4. Если же изменить логический уровень на выводе ВМ с нуля на единицу, то независимо от того, какой логический уровень подан на вывод ВШ, выводы В1—В4 и С1—С4 переходят в высокоимпедансное состояние и отключаются от схемы (рис. 46, *г* вклейки). Для облегчения действий по налаживанию микропроцессорного модуля необходимо запомнить, что изменение логического уровня на управляющем выводе ВШ приводит к смене направления передачи данных, а изменение уровня на ВМ вызывает подключение или отключение выводов В1—В4 и С1—С4 от соответствующих шин. Следует иметь в виду, что в литературе нет единообразия в обозначении выводов микросхем, в частности в обозначении выводов К589АП16. Вывод ВМ часто обозначается буквами ВК или латинскими CS, а вывод ВШ — в виде ВР, УВ или DIEN (подробнее об этом см. табл. 9 Приложения). Чтобы читатели могли привыкнуть к различным обозначениям управляющих выводов микросхем, на рис. 12 и на всех последующих принципиальных схемах выводы ВМ шинных формирователей обозначены буквами ВК, а выводы ВШ — буквами ВР.

Шинный формирователь К589АП16 изготовлен по технологии типа ТТЛШ. Не вникая в подробности этой технологии, отметим, что величина входного тока для ТТЛШ намного меньше, чем для микросхем ТТЛ. Поэтому к каждому выходу МП—580 можно подключать не один, а несколько входов микросхемы К589АП16.

Рассмотрим теперь соединения адресных выводов А0—А15 микропроцессора (рис. 12) с контактами разъема XP1. Эти соеди-

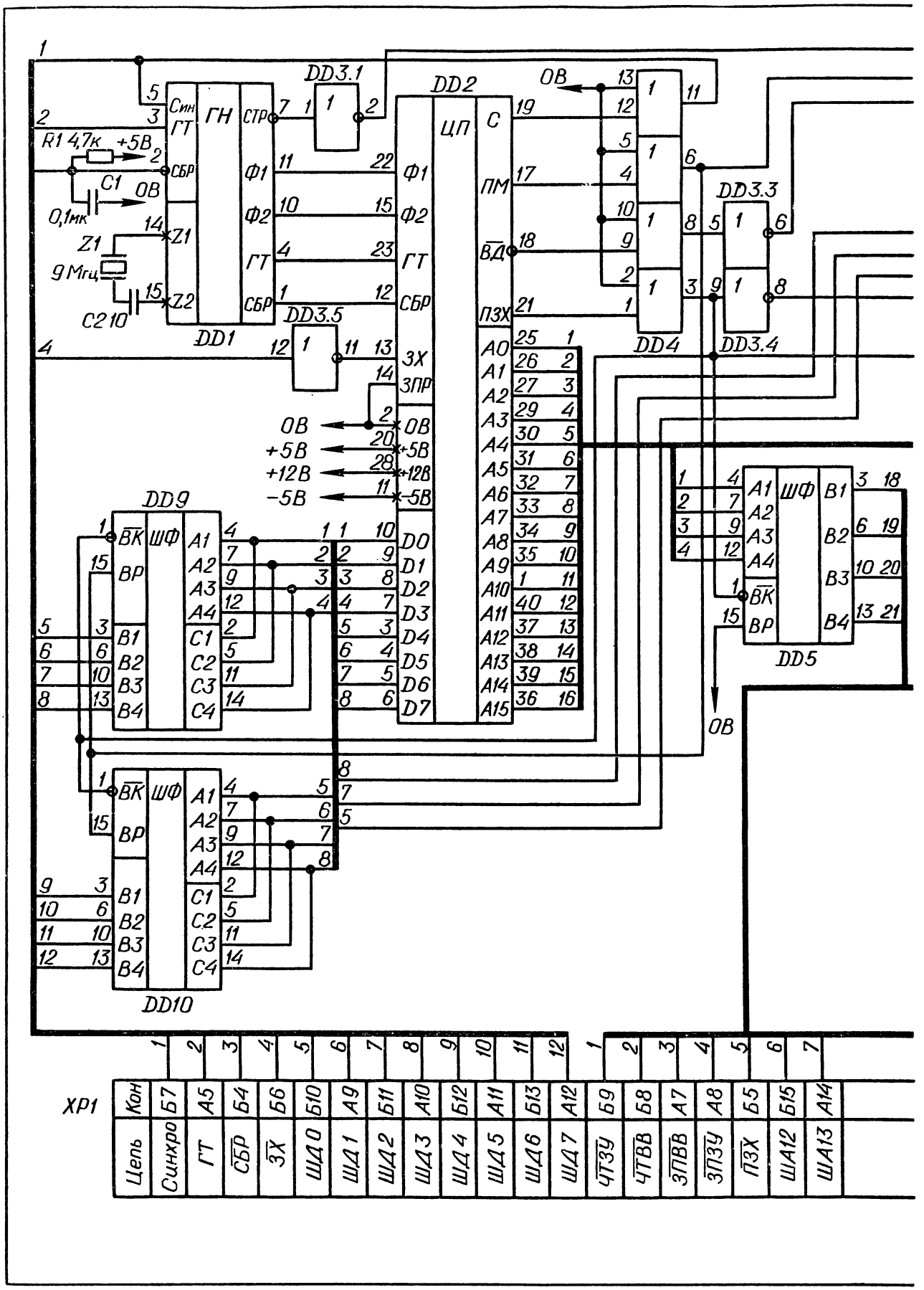
нения выполняются при помощи четырех шинных формирователей DD5—DD8 типа К589АП16, каждый из которых обеспечивает подключение четырех адресных выводов микропроцессора к адресной шине.

Шинные формирователи выполняют здесь двойную роль. Во-первых, они усиливают по мощности сигналы, поступающие от выводов А0—А15 микропроцессора, и, во-вторых, отключают свои выводы В1—В4 от общей шины по сигналу ПЗХ, возникающему на выводе 21 (ПЗХ) микропроцессора в виде уровня логической единицы. Сигнал ПЗХ подается на выводы 1 (ВК) всех четырех шинных формирователей через усилитель-повторитель, выполненный на одном элементе микросхемы DD4. Этот элемент представляет собой логическую схему 2 ИЛИ. Если обратиться к таблице истинности такого элемента, то можно убедиться в том, что сигнал на выводе 3 будет повторять входной сигнал, подаваемый на вывод 1, если на выводе 2 установлен уровень логического нуля. После того как сигнал ПЗХ микропроцессора пройдет через элемент DD4, он усиливается по мощности, и к выводу 3 элемента DD4 разрешается подключать уже до десяти входов микросхем типа ТТЛ. В нашем случае к выводу 3 подключено шесть входов ВК микросхем DD5—DD10 и один вход микросхемы DD.3.4.

На управляющие выводы 15 (ВР) всех четырех шинных формирователей DD5—DD8 постоянно подается уровень логического нуля, путем соединения выводов ВР с выводом ОВ питающего напряжения. Поэтому шинные формирователи DD5—DD8 не могут изменять направление передачи сигналов. У них выводы А1—А4 будут соединены с выводами В1—В4, а выводы С1—С4 — не используются, на них сигналы отсутствуют. По этой причине выводы С1—С4 шинных формирователей DD5—DD8 не показаны на принципиальной схеме рис. 12.

Рассмотрим схему соединения выводов данных D0—D7 микропроцессора с разъемом XP1. Эти соединения выполняются при помощи двух шинных формирователей DD9 и DD10 типа К589 АП16. Здесь шинные формирователи выполняют тройную роль. Во-первых, они, как и в предыдущем случае, усиливают сигналы, поступающие с выводов D0—D7 микропроцессора; во-вторых, отключают свои выводы В1—В4 от общей шины, когда на их управляющие выводы 1 (ВК), как и на аналогичные управляющие выводы 1 (ВК) шинных формирователей DD5—DD8, поступает сигнал ПЗХ, в-третьих, шинные формирователи DD9—DD10 обеспечивают двунаправленную передачу данных при осуществлении режимов чтения или записи.

Во время режима чтения, когда микропроцессор принимает данные, на его выводе 17 (ПМ) образуется уровень логической единицы. Этот уровень поступает в усилитель-повторитель DD4, а из него на входы 15 (ВР) шинных формирователей DD9—DD10. Под воздействием уровня логической единицы, поданного на ВР, выводы В1—В4 соединяются с выводами С1—С4 и данные поступают из общей шины в микропроцессор для чтения.



В режиме записи на выводе 17 (ПМ) микропроцессора образуется уровень логического нуля, передаваемый через DD4 на выводы 15 (ВР) микросхем DD9—DD10. В результате этого выводы А1—А4 соединяются с выводами В1—В4 и данные из микропроцессора поступают в общую шину для последующей записи в память или устройство вывода.

Прежде чем перейти к рассмотрению схемы работы тактового

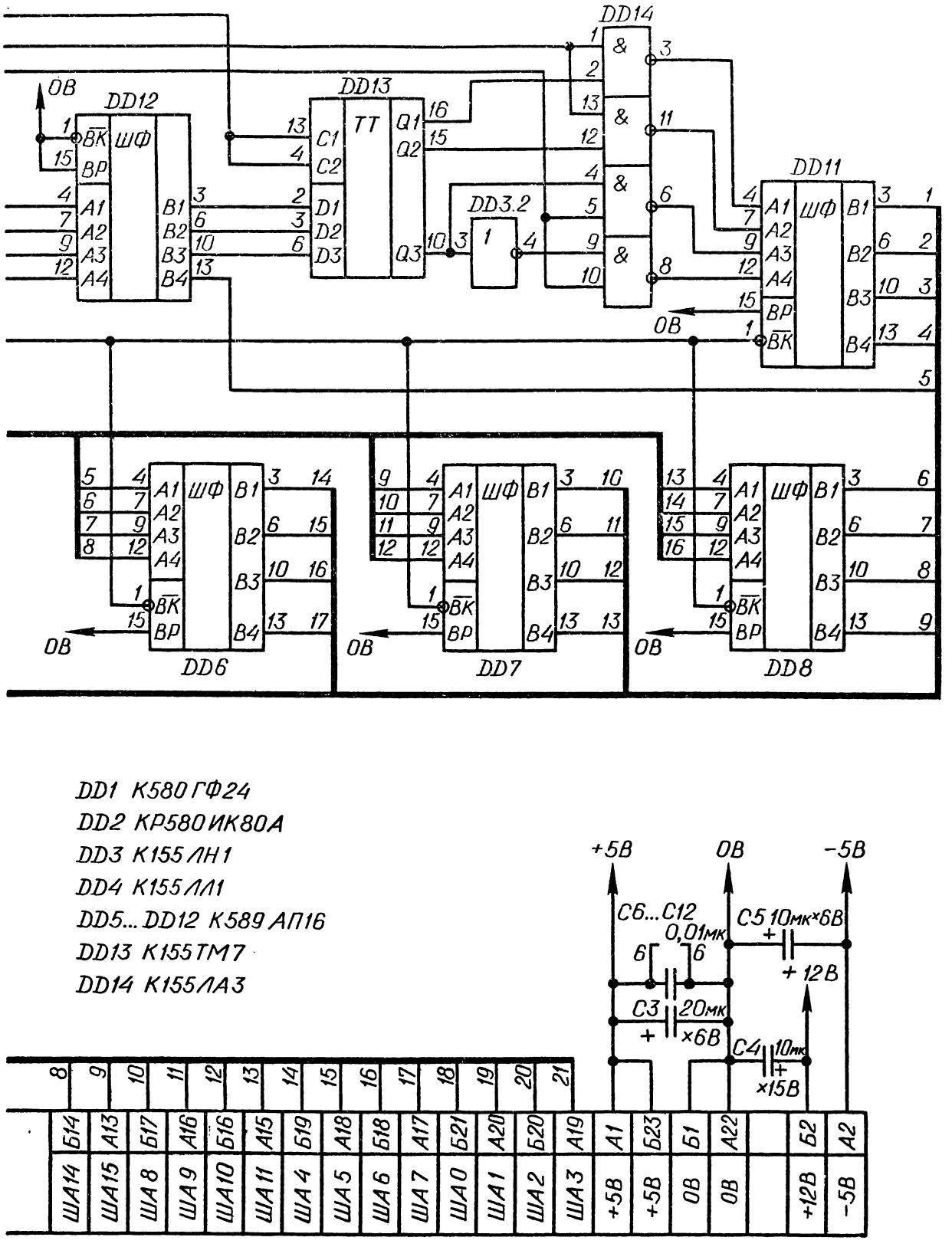


Рис. 12

генератора ТГ и устройства формирования управляющих сигналов ФУ, необходимо ознакомиться со смыслом некоторых понятий, «цикл команды», «машинный цикл» и «машинный такт», а также усвоить некоторые характерные действия, совершаемые микропроцессором при выполнении команд программы.

Циклом команды Тц называют время, в течение которого команда выбирается из памяти и исполняется. Этот цикл реали-

зуется за несколько машинных циклов М (от 1 до 5 для МП—К580), а каждый машинный цикл разделяется на некоторое количество машинных тактов Т (от 3 до 5).

В колонке 6 табл. 1 системы команд МП—К580 (см. Приложение, с. 133) указано количество машинных циклов (первая цифра) и машинных тактов (вторая цифра), затрачиваемых микропроцессором на каждую команду. Для уяснения действий МП—К580 при выполнении машинных циклов и тактов полезно обратиться к временным диаграммам.

На рис. 48 вклейки приведена временная диаграмма выполнения микропроцессором простейшей команды, состоящей из одного машинного цикла М1 и пяти машинных тактов Т1—Т5, а на рис. 49 вклейки — выполнение более сложной команды, состоящей из трех машинных циклов М1—М3 и десяти машинных тактов (Т1—Т4, Т1—Т3 и Т1—Т3). Из этих диаграмм видно, что длительность машинного такта Т равна периоду следования тактовых сигналов Ф1 и Ф2.

В каждом машинном цикле М микропроцессор выполняет определенные действия. За время первого такта Т1 он устанавливает на своих адресных выводах А0—А15 адрес ячейки памяти, порта или стека в зависимости от того, куда он будет обращаться в данном машинном цикле. В этом же такте (Т1) микропроцессор посылает на выводы данных D0—D7 сигналы о своем состоянии и о тех действиях, которые он будет совершать в текущем машинном цикле. Эти сигналы будут описаны далее, а сейчас отметим, что в машинном такте Т1 одновременно с сигналами состояния микропроцессор выдает на вывод С синхронизирующий сигнал, указывающий на начало машинного цикла. Сигнал С (синхро) используется для управления работой отдельных устройств и узлов микроЭВМ. По времени сигналы состояния и С выдаются в течение одного машинного такта. В машинном такте Т2 микропроцессор проверяет состояние сигналов на своих входах ГТ и ЗХ. Если на входе ГТ нет логической единицы, то микропроцессор приостанавливает свою работу и переходит в состояние ожидания. Наличие единицы на входе ЗХ заставляет микропроцессор отключиться от общей шины и выдать сигнал на вывод ПЗХ, разрешающий внешним устройствам занять общую шину. В машинном такте Т3 микропроцессор производит обмен одним байтом данных с памятью или портами. На временной диаграмме рис. 48 вклейки сигналы ПМ (прием) и ВД (выдача) условно показаны одновременно. На самом деле может быть выдан только один из этих сигналов, потому что микропроцессор за один машинный цикл может только передать или принять один байт данных. В последующих тактах Т4 и Т5 микропроцессор выполняет различные внутренние межрегистровые пересылки и обработку данных в соответствии с командой. В некоторых машинных циклах эти два такта могут отсутствовать или иметь место только один машинный такт Т4.

В связи с тем, что логические уровни на выводах адреса А0—А15 и данных D0—D7 имеют в каждом разряде нулевое или еди-

ничное значение, временные диаграммы этих уровней условно изображают в виде фигуры, приведенной на рис. 50 вклейки, а высокоимпедансное состояние отображают пунктирной линией.

Продолжим изучение принципиальной схемы микропроцессорного модуля (рис. 12). Рассмотрим, как образуются тактовые сигналы Ф1 и Ф2. Их формирование осуществляется при помощи специальной микросхемы DD1 типа К580ГФ24. Если к выводам 14 и 15 этой микросхемы подсоединить кварцевый резонатор Z1, то на выводах 10 и 11 образуются тактовые сигналы Ф1 и Ф2, частота следования которых будет в 9 раз меньше собственной частоты колебаний кварца Z1 (деление частоты именно в 9 раз обусловлено особенностями временных соотношений сигналов внутри схемы DD1). Для нормальной работы МП—К580 частота тактовых сигналов Ф1 и Ф2 должна находиться в пределах от 0,5 до 2 МГц. Таким образом, частота кварца Z1 должна быть выбрана от 4,5 до 18 МГц. В рассматриваемой микроЭВМ установлен кварц Z1, имеющий резонансную частоту 9 МГц. Следовательно, частота сигналов Ф1 и Ф2 составляет 1 МГц, а период повторения — 1 мксек. Последовательно с кварцем Z1 включен конденсатор С2, обеспечивающий более устойчивую работу кварцевого генератора. Тактовые сигналы Ф1 и Ф2 с выводов 11 и 10 DD1 поступают на соответствующие выводы 22 и 15 микропроцессора DD2.

Микросхема DD1, помимо основного назначения — генерирования сигналов Ф1 и Ф2, выполняет ряд дополнительных действий. В частности, она обеспечивает выдачу микропроцессору DD2 на его выводы 23 и 12 сигналов ГТ и СБР в нужный момент времени (в начале такта Т1), несмотря на то, что эти сигналы приходят на выводы 2 и 3 микросхемы DD1 в произвольное время.

Элементы R1 и C1 служат для автоматического сброса в исходное состояние счетчика команд микропроцессора при каждом включении питающего напряжения. Происходит это так. Конденсатор С1 при включении питания разряжен, поэтому на выводе 2 (СБР) микросхемы DD1 имеется нулевой уровень. Этот уровень передается через внутреннюю схему DD1 на ее вывод 1 (СБР) и далее на вывод 12 (СБР) микропроцессора DD2, в результате чего счетчик команд DD2 устанавливается в нулевое состояние и микропроцессор может начать выполнение программы с нулевой команды. Затем конденсатор С1 заряжается через резистор R1 и на выводе 2 микросхемы DD1 образуется логическая единица,— сигнал СБР автоматически прекращает свое действие. Микросхема DD1 формирует также специальный сигнал СТРОБ, используемый для управления работой различных узлов микроЭВМ. Сигнал СТРОБ совпадает по времени с сигналами С и Ф2.

Перейдем к рассмотрению наиболее сложной части микропроцессорного модуля, выполняющего формирование сигналов ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ.

Промышленность выпускает специальные БИС типа КР580ВК28 и КР580ВК38, позволяющие получить эти четыре управляющих сигнала. Учитывая дефицитность указанных БИС, на рис. 12

приведен один из вариантов схемы формирования сигналов ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ, составленный при помощи набора более доступных типов микросхем.

Обычно формирование управляющих сигналов ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ осуществляют путем использования **сигналов состояния,** кратковременно выдаваемых микропроцессором на выводы данных в начале каждого машинного цикла. Сигналы состояния для МП—К580 сведены в табл. 3.1. В левой колонке таблицы эти сигналы описаны, а в правых — указаны логические уровни (1 или 0), возникающие на выводах данных D0—D7 микропроцессора для каждого сигнала. В литературе совокупность из вось-

Таблица 3.1

№ п/п	Сообщение о предстоящем действии МП—К580 или его состоянии	Логические уровни на выводах данных МП—К580							
		D0	D1	D2	D3	D4	D5	D6	D7
1	Начало первого машинного цикла. Будет выбираться первый байт команды	0	1	0	0	0	1	0	1
2	Будет выполняться чтение из памяти (ЧТЗУ)	0	1	0	0	0	0	0	1
3	Будет выполняться чтение из порта (ЧТВВ)	0	1	0	0	0	0	1	0
4	Будет выполняться чтение из стека	0	1	1	0	0	0	0	1
5	Будет выполняться запись в память (ЗПЗУ)	0	0	0	0	0	0	0	0
6	Будет выполняться запись в порт (ЗПВВ)	0	0	0	0	1	0	0	0
7	Будет выполняться запись в стек	0	0	1	0	0	0	0	0
8	Подтверждение режима прерывания	1	1	0	0	0	1	0	0
9	Подтверждение режима останова	0	1	0	1	0	0	0	1
10	Подтверждение прерывания при останове	1	1	0	1	0	1	0	0

ми логических уровней сигнала состояния называют **словом состояния.**

Все десять слов состояния, приведенных в табл. 3.1, используют лишь в микроЭВМ достаточно сложной структуры. В более простых микроЭВМ применяют только некоторые слова состояния, а в словах часто используют не восемь, а меньшее количество разрядов.

В описываемой микроЭВМ для формирования сигналов ЧТЗУ, ЧТВВ, ЗПЗУ и ЗПВВ используют четыре слова состояния (2, 3, 5 и 6 из табл. 3.1.), а в этих словах — четвертые, шестые и седьмые разряды (сигналы состояния, возникающие на выводах D4, D6 и D7 микропроцессора DD2).

Помимо этих сигналов, дополнительно применяют сигналы ПМ и ВД, выдаваемые микропроцессором. Поступают так потому, что сигналы состояния нельзя непосредственно использовать в качестве управляющих режимом чтения или записи. Эти сигналы весьма кратковременные и появляются они не тогда, когда микропроцессор должен выполнять чтение или запись. Сигналы состояния появляются в первом машинном такте (Т1), а чтение — запись выполняется в третьем (Т3). Поэтому сигналы состояния приходится улавливать в такте Т1 и где-то запоминать, чтобы в такте Т3 использовать для управления. Запоминание сигналов выполняется при помощи триггеров (DD13 на рис. 12), а их выдача в такте Т3 путем дополнительного применения сигналов ПМ и ВД, действующих (как это видно из временной диаграммы рис. 48 вклейки) в третьем машинном такте.

Используя принципиальную схему рис. 12, рассмотрим подробнее, как это происходит.

Сигналы состояния, возникающие на выводах D4, D6 и D7 микропроцессора DD2, нельзя непосредственно подавать на входы D1, D2 и D3 триггеров DD13, так как ко входам D0—D7 разрешается подключать только один вход микросхемы типа ТТЛ, а к ним уже подключены входы микросхем DD9 и DD10. Поэтому на пути прохождения сигналов состояния устанавливают микросхему DD12 типа ТТЛШ, выполняющую роль усилителя.

Выводы ВК и ВР этой микросхемы постоянно соединены с нулевым уровнем питающего напряжения и DD12 работает только как усилитель сигналов состояния, поступающих с выводов D4, D6 и D7 микропроцессора на ее входы А1—А3, и сигналов с выходов В1—В3, передаваемых на входы D1—D3 микросхемы DD13. Эта микросхема (типа К155ТМ7) содержит четыре триггера. В схеме рис. 12 используется три триггера. Согласно таблице истинности для К155ТМ7, каждый триггер запоминает и выдает на выход Q сигнал, поступающий на его вход только тогда, когда на вывод С подается разрешающий уровень в виде логической единицы. Таким уровнем является сигнал СТРОБ, формируемый микросхемой DD1 в такте Т1 (во время выдачи микропроцессором сигналов состояния). Так как сигнал СТРОБ выдается из DD1 в виде уровня логического нуля (о чем свидетельствует кружочек,

изображенный у вывода 7), то включается инвертор DD3.1., преобразующий этот уровень в логическую единицу. При поступлении на входы С1—С2 триггера DD13 положительного сигнала СТРОБ (в виде уровня 1) первый триггер DD13 запоминает и выдает на выход Q1 сигнал состояния, сообщающий о чтении данных из памяти (ЧТЗУ), второй триггер запоминает и выдает на Q2 сигнал о чтении данных из порта (ЧТВВ), третий триггер выдает на Q3 сигнал, извещающий о записи данных в порт (ЗПВВ). Однако эти сигналы тоже нельзя использовать в качестве управляющих режимом чтение—запись, так как они, запомнившись в триггерах, действуют на выходах Q1, Q2 и Q3 на протяжении всего машинного цикла, а нам они нужны только в такте Т3. Поэтому используется сочетание этих сигналов состояния и сигналов ПМ и ВД, выдаваемых микропроцессором в такте Т3. Для этого используют четыре элемента 2И—НЕ микросхемы DD14.

Согласно таблице истинности, элемент 2И разрешает прохождение сигнала, поступающего на его вход 2, и выдает этот сигнал на выход 3 только тогда, когда на входе 1 имеется уровень логической единицы. Поэтому, подавая на вход 2 элемента DD14 сигнал состояния из Q1, извещающий о чтении данных из памяти, а на вход 1 — уровень логической единицы сигнала ПМ, мы на выходе 3 элемента DD14 получим нужный нам сигнал управления $\overline{\text{ЧТЗУ}}$. Причем получим его именно в такте Т3, когда на схеме 2И элемента DD14 совпадут по времени действия сигнал ПМ и сигнал состояния, поступающий из Q1.

Кроме того, получим сигнал ЧТЗУ в инверсном виде, о чем свидетельствует черта над его обозначением.

Аналогично формируется сигнал чтения из порта ($\overline{\text{ЧТВВ}}$). Для этого на вход 12 элемента DD14 подают сигнал состояния из Q2 («Чтение данных из порта»), а на вход 13 сигнал ПМ. Тогда на выходе 11 образуется управляющий сигнал $\overline{\text{ЧТВВ}}$. Для формирования управляющих сигналов записи используют два нижних элемента микросхемы DD14. На входы 4 и 5 подают соответственно сигналы состояния из Q3 («Запись данных в порт») и сигнал ВД. При этом на выходе 6 выдается управляющий сигнал $\overline{\text{ЗПВВ}}$. Так как на вход 5 должен подаваться сигнал в виде логической единицы, а из микропроцессора сигнал ВД поступает в виде логического нуля, то установлен инвертор DD3.3.

Для формирования сигнала, управляющего записью в память, следовало бы использовать пятое слово состояния (ЗПЗУ). Однако, как это видно из табл. 3.1, оно отличается от шестого (ЗПВВ) только тем, что в разряде D4 содержится нуль вместо единицы. Поэтому, инвертируя при помощи микросхемы DD3.2 сигнал, находящийся на Q3 (ЗПВВ), получаем пятое слово состояния (ЗПЗУ), подав которое на вход 9 элемента DD14, можно получить на выходе 8 управляющий сигнал $\overline{\text{ЗПЗУ}}$, если на вход 10 подать сигнал ВД.

Сформированные сигналы $\overline{\text{ЧТЗУ}}$, $\overline{\text{ЗПЗУ}}$, $\overline{\text{ЧТВВ}}$ и $\overline{\text{ЗПВВ}}$ с инверсных выходов 3, 8, 6 и 11 микросхемы DD14 поступают

в виде уровня логического нуля на входы А1—А4 шинного формирователя DD11, а с его выходов В1—В4 на контакты разъема XP1 общей шины. Управляющие выводы $\overline{ВК}$ DD11 соединены с управляющими выводами $\overline{ВК}$ всех остальных шинных формирователей (DD5—DD10) микропроцессорного модуля, что позволяет с приходом сигнала ПЗХ отключать от общей шины не только выводы данных и адреса микропроцессора, но и управляющие сигналы $\overline{ЧТЗУ}$, $\overline{ЗПЗУ}$, $\overline{ЧТВВ}$ и $\overline{ЗПВВ}$.

Сигнал ПЗХ, помимо управления шинными формирователями микропроцессорного модуля, применяется для управления и другими узлами микроЭВМ. С этой целью он инвертируется микросхемой DD3.4, усиливается микросхемой DD12 (вход А4—выход В4) и поступает в виде уровня логического нуля (сигнал $\overline{ПЗХ}$) на контакт Б5 разъема XP1 общей шины.

В заключение отметим, что из общей шины (из контакта Б6 разъема XP1) приходит сигнал $\overline{ЗХ}$, запрашивающий у микропроцессора разрешения на захват шин. Этот сигнал приходит в виде уровня логического нуля, а в микропроцессор DD2 (на его вывод 13) он должен поступать в виде уровня логической единицы. Для инвертирования сигнала $\overline{ЗХ}$ установлена микросхема DD3.5.

Рассмотренный микропроцессорный модуль имеет некоторые схемные излишества. Как мы узнаем из раздела 3.2, выполнение простейших учебных действий микроЭВМ можно достичь при помощи более простой схемы микропроцессорного модуля. Вместе с тем, если учитывать дальнейшее совершенствование собираемой микроЭВМ, то следует стремиться к изготовлению микропроцессорного модуля по схеме рис. 12. Это избавит от необходимости переделывать схему и добавлять новые элементы при последующей модернизации.

Модуль памяти

Из структурной схемы рис. 47 вклейки видно, что модуль памяти состоит из оперативного запоминающего устройства ОЗУ, служащего для записи, хранения и чтения данных, шинного формирователя ШФД, обеспечивающего двунаправленное действие шины данных, и логической схемы ЛС, использующей управляющие сигналы ЧТЗУ и ЗПЗУ, поступающие по шине управления ШУ, для создания необходимых логических уровней (1 или 0), подаваемых на управляющие выводы ОЗУ и ШФД при чтении или записи. Подключение модуля памяти к общей шине выполняется с помощью многоконтактного разъема XP.

Обычно простейшие микроЭВМ комплектуются памятью, состоящей только из ОЗУ. Объем такой памяти насчитывает 1—4 Кбайт. При дальнейшей модернизации микроЭВМ объем ОЗУ может быть увеличен до 16—64 Кбайт, а также добавлено ПЗУ емкостью 2—8 Кбайт. Размещение дополнительных микросхем ОЗУ и ПЗУ можно предусмотреть на плате первоначального

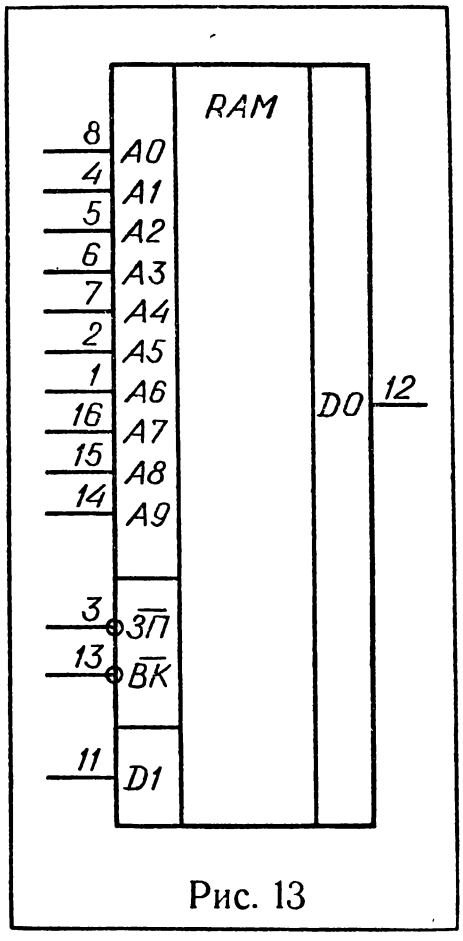


Рис. 13

модуля памяти или же собрать их в виде отдельного, добавочного модуля, включаемого в резервный разъем общей шины.

Промышленность выпускает большое количество типов микросхем для ОЗУ и ПЗУ, что облегчает их выбор и приобретение. Исходя из требования наименьшей дефицитности, для ОЗУ описываемой микроЭВМ выбрана самая распространенная (хотя и несколько устаревшая) микросхема типа К565РУ2, содержащая 1024 запоминающих элемента. Расположение выводов этой микросхемы приведено на рис. 13. К565РУ2 имеет десять адресных выводов А0—А9. На эти выводы из микропроцессора подают десятиразрядный двоичный адресный код.

Нормальная работа К565РУ2 обеспечивается при уровне логического нуля на выводе ВК. Если же на вывод ВК подать уровень логической единицы, то выход D0 микросхемы К565РУ2 переходит в высокоимпедансное состояние. Для чтения бита данных, выдаваемого на выход D0 микросхемы, на вывод ЗП, необходимо подать уровень логической единицы, проследив при этом, чтобы на выводе ВК сохранялся уровень логического нуля. Запись бита данных в элемент памяти производится путем подключения логического уровня (1 или 0) этого бита ко входу D1 микросхемы и подачи на ее вывод ЗП уровня логического нуля. При этом выполнение режима записи имеет свои особенности. Заключаются они в том, что нулевые уровни логических напряжений на выводы ЗП и ВК должны подаваться на 0,5 мксек позже, чем сигналы А0—А9, поступающие на адресные выводы. Режим чтения данных выполняется без временных ограничений. На выводы ЗП и ВК достаточно подать уровень логической единицы, чтобы на выводе D0 появилось содержимое элемента памяти, адрес которого установлен на выводах А0—А9. При изменении адреса на вывод D0 будет выдано содержимое нового элемента памяти, соответствующее новому адресу.

На рис. 14 приведена принципиальная электрическая схема модуля памяти. В связи с тем, что МП—К580 выполняет действие с восьмиразрядными данными, а К565РУ2 запоминает только один разряд (бит) данных, то приходится устанавливать восемь корпусов (DD1—DD8) этих микросхем, что позволяет записывать, запоминать или считывать 1024 байта данных.

Адресные выводы А0—А9 на всех восьми корпусах К565РУ2 соединяются параллельно и подключаются непосредственно к выводам ША0—ША9 адресной шины разъема ХР1. Входы D1 всех

восьми микросхем DD1—DD8 подключаются к выводам С1—С4 шинных формирователей DD9—DD10, а выходы D0 — к выводам А1—А4. Выводы В1—В4 подсоединяются к выводам ШД0—ШД7 шины данных при помощи разъема XP1.

Шинные формирователи DD9 и DD10 подобно шинным формирователям DD9 и DD10 микропроцессорного модуля (см. рис. 12) обеспечивают двунаправленное действие шины данных, а также осуществляют отключение ячеек памяти от общей шины.

Восемь управляющих выводов $\overline{ВК}$ на микросхемах памяти DD1—DD8 и два управляющих вывода $\overline{ВК}$ на шинных формирователях DD9—DD10 соединяются друг с другом и подключаются к выходу 6 логического элемента DD11.2. В рабочем состоянии модуля памяти на этом выходе 6 должен быть уровень логического нуля. С появлением уровня логической единицы все выходы D0 микросхем DD1—DD8 переходят в высокоимпедансное состояние и отключаются от общей шины.

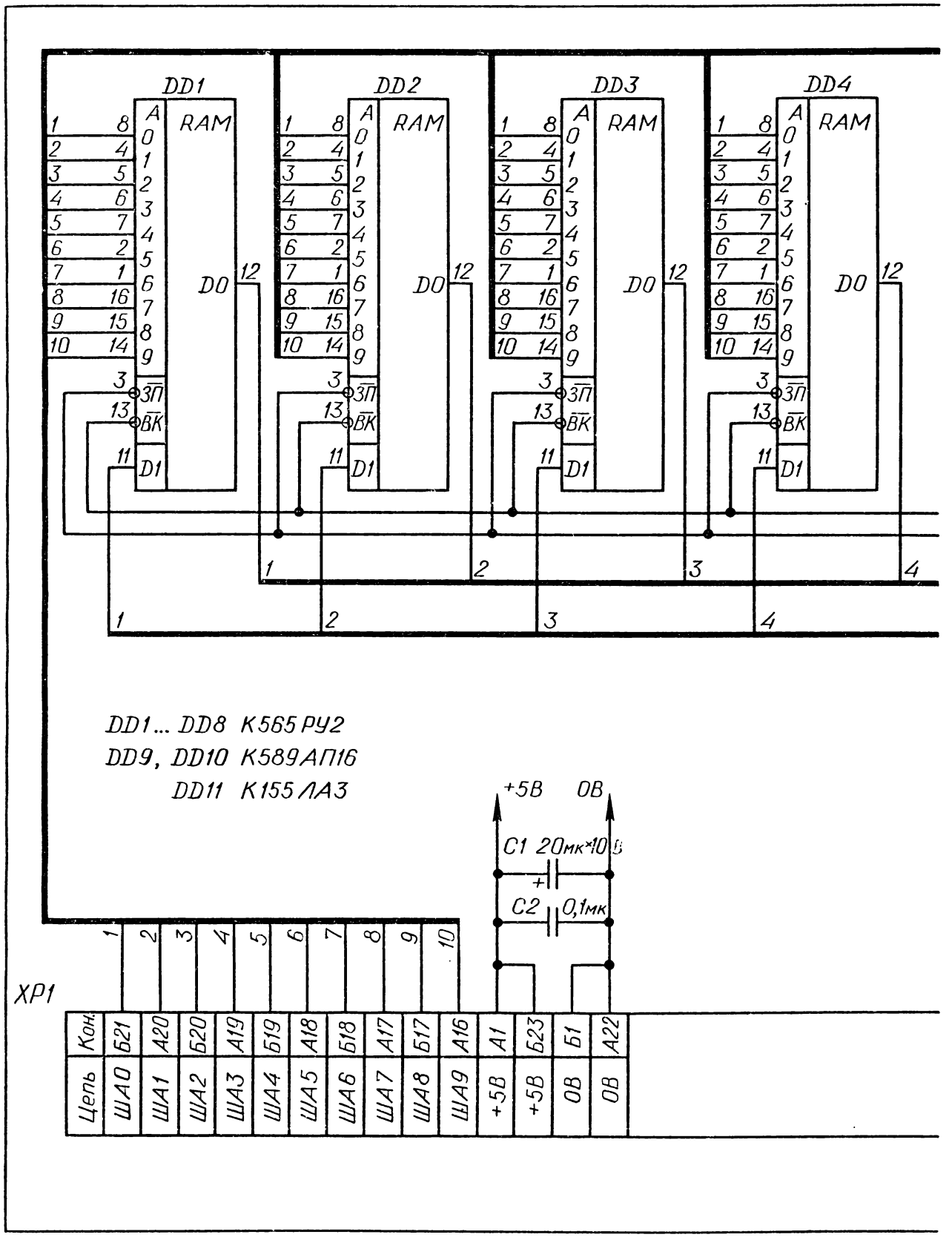
Восемь управляющих выводов $\overline{ЗП}$ микросхем DD1—DD8 соединяются параллельно друг с другом и подключаются к выводу $\overline{ЗПЗУ}$ общей шины, что позволяет осуществлять режим чтения (когда $\overline{ЗПЗУ} = 1$) или записи (когда $\overline{ЗПЗУ} = 0$). Кроме того, сигналы $\overline{ЗПЗУ}$ и $\overline{ЧТЗУ}$ поступают на входы логического элемента DD11.1, который совместно с элементом DD11.2 вырабатывает управляющий сигнал, подаваемый на десять выводов $\overline{ВК}$ для отключения модуля памяти от общей шины в тот момент, когда микропроцессор не обращается к памяти.

Сигнал «Блокировка ОЗУ», подаваемый на вывод А4 разъема XP1, используется при совместной работе модуля памяти и дополнительного модуля в виде ПЗУ или внешних запоминающих устройств, подключаемых к общей шине. Подача уровня логического нуля на вывод А4 позволяет выключать модуль памяти и считывать данные с других запоминающих устройств, подсоединенных к общей шине.

Модуль управления

Модуль управления (рис. 51 вклейки) состоит из 16 светодиодов США, подключенных к адресной шине ША и указывающих своим свечением код адреса, установленный на этой шине, а также 8 светодиодов СШД, подключенных к шине данных ШД и указывающих код, содержащийся на этой шине. Помимо светодиодов, в состав модуля управления входит 16 переключателей ПША, позволяющих вручную устанавливать код адреса, а также восемь переключателей ПШД для установки кода данных. Подключение переключателей ПША и ПШД к общей шине выполняется через шинные формирователи ШФА и ШФД, позволяющие включать или отключать переключатели по сигналам управления, поступающим из шины управления.

Модуль управления имеет также узел УШР, позволяющий осуществлять **шаговый режим работы,** заключающийся в исполнении программы отдельными машинными циклами. Выполнение



машинного цикла команды происходит при нажатии кнопки, установленной в УШР. Выполнив один машинный цикл, микропроцессор приостанавливает работу и переходит в режим ожидания следующего нажатия кнопки.

Узел управления УУ содержит ряд кнопок, тумблеров и электронных элементов, служащих для реализации различных режимов работы микроЭВМ и создания ряда управляющих сигналов.

Принципиальная электрическая схема модуля управления приведена на рис. 15. Светодиоды VD1—VD24 типа АЛ307Б

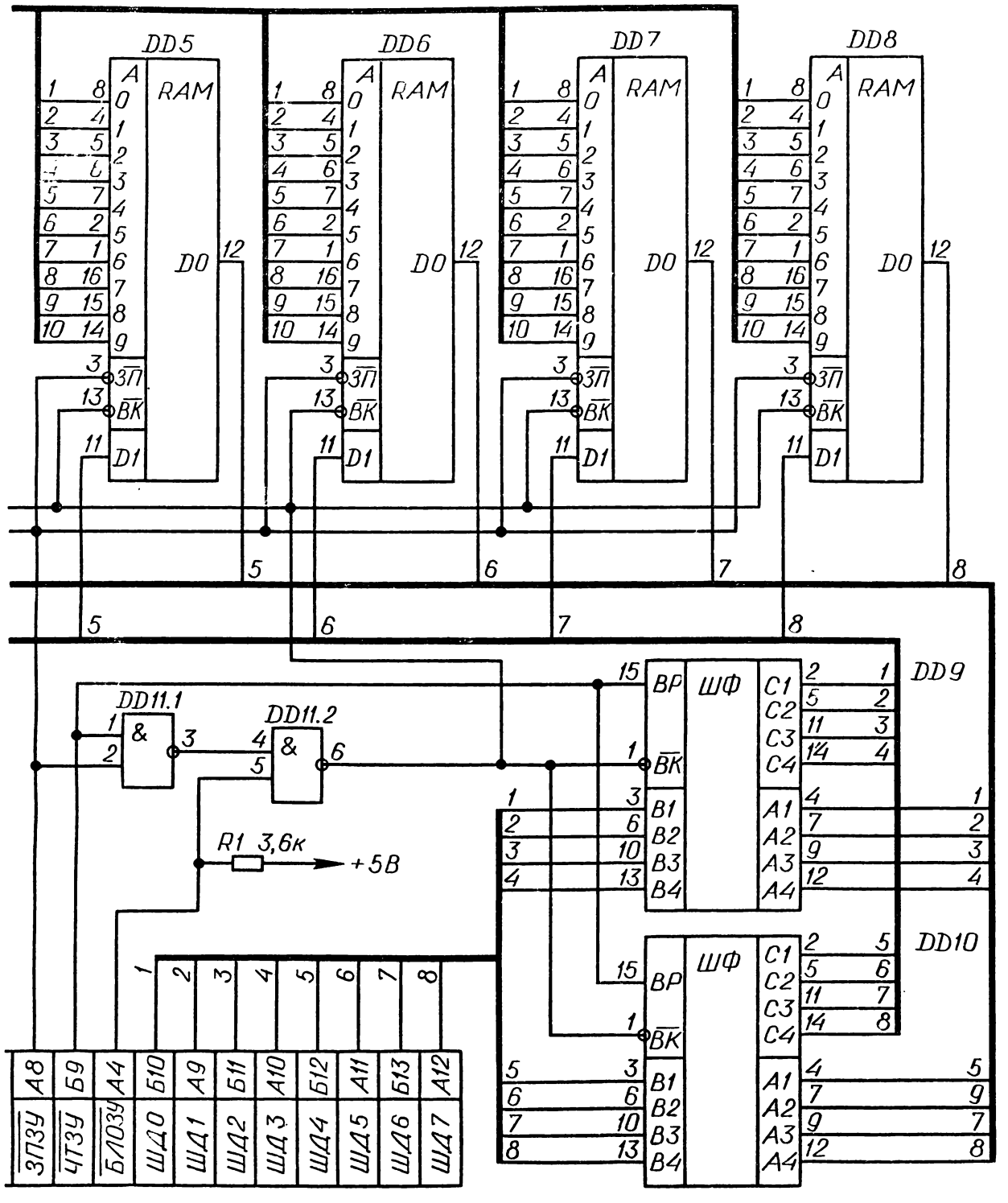


Рис. 14

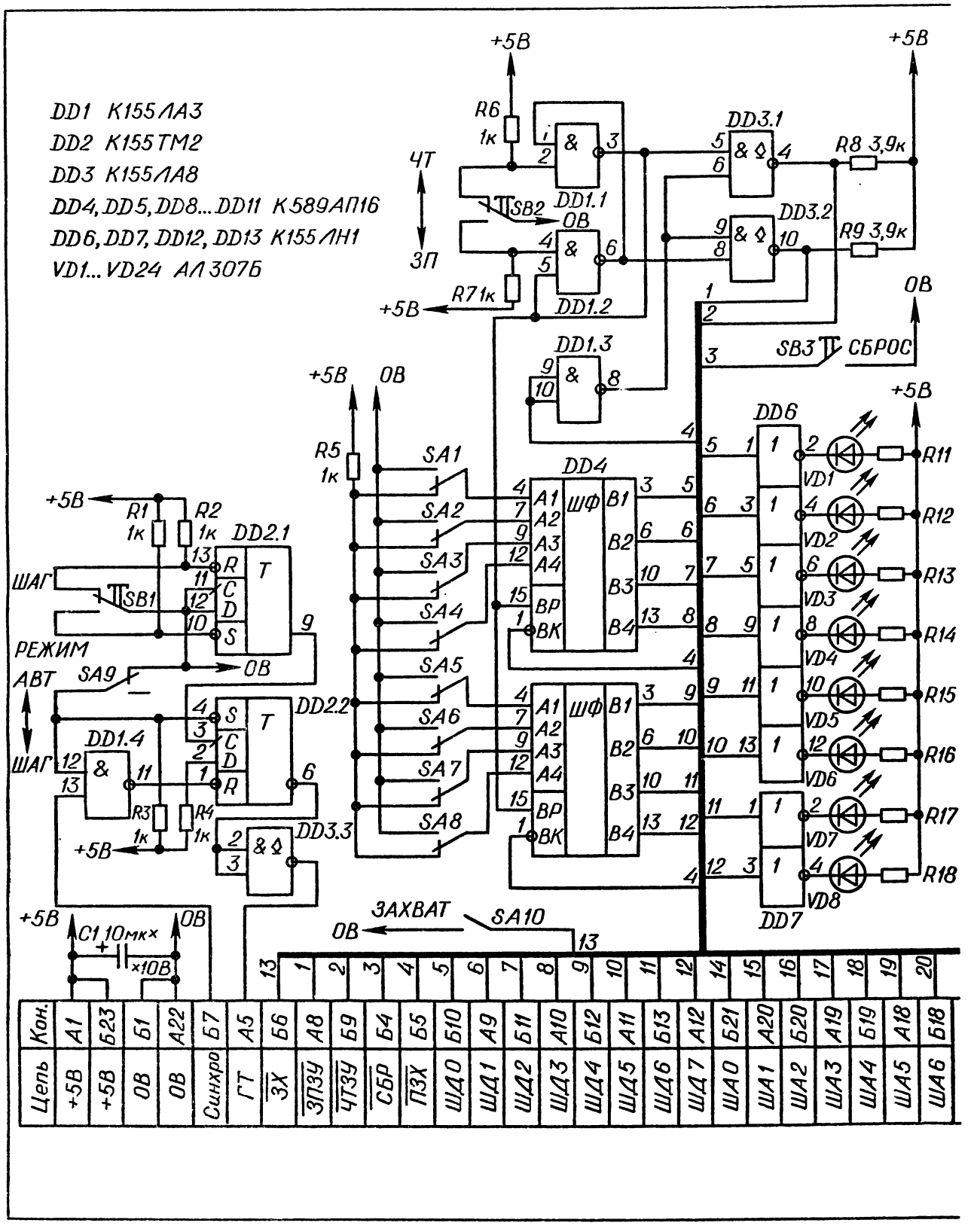
потребляют значительный ток (около 10 мА каждый), поэтому их непосредственное подключение к шинам адреса и данных вызывает повышенную нагрузку на шины. Для уменьшения нагрузки светодиоды подключают к шинам через логические элементы DD6, DD7, DD12 и DD13, выполняющие роль усилителей.

Учитывая, что эти элементы не только усиливают, но и инвертируют входные сигналы, аноды светодиодов подключают к цепи питающего напряжения +5 В, а катоды — к выходам логических элементов. Резисторы R11—R34 ограничивают ток светодиодов

до рабочего уровня. Свечение каждого светодиода сигнализирует о том, что на проводе общей шины, к которому он подключен, имеется сигнал высокого логического уровня (1), а отсутствие свечения сообщает о низком логическом уровне (0).

В качестве переключа́телей SA1—SA8 и S11—S26 использованы тумблеры типа МТ1. Резисторы R5 и R10 понижают напряжение +5 В до уровня напряжения логической единицы.

Для шинных формирователей DD4, DD5 и DD8—DD11 выбраны уже известные нам микросхемы типа К589АП16. Все управляющие выводы ВК этих микросхем подключены к выводу $\overline{\text{ПЗХ}}$



разъема XP1 Если на этот вывод $\overline{\text{ПЗХ}}$ поступает сигнал микропроцессора в виде уровня логического нуля, разрешающий занять шины, то переключатели SA11—SA26 подключаются к адресной шине для набора двоичного кода адреса.

Выводы ВР шинных формирователей данных DD4—DD5 подключены к выводу 3 микросхемы DD1.1. На этом выводе имеется уровень логической единицы, так как вывод 2 с помощью контактов кнопки SB2 подключен к уровню логического нуля. Появление высокого уровня на выводах ВР формирователей DD4—DD5 приводит к тому, что выводы В1—В4 соединяются с выводами

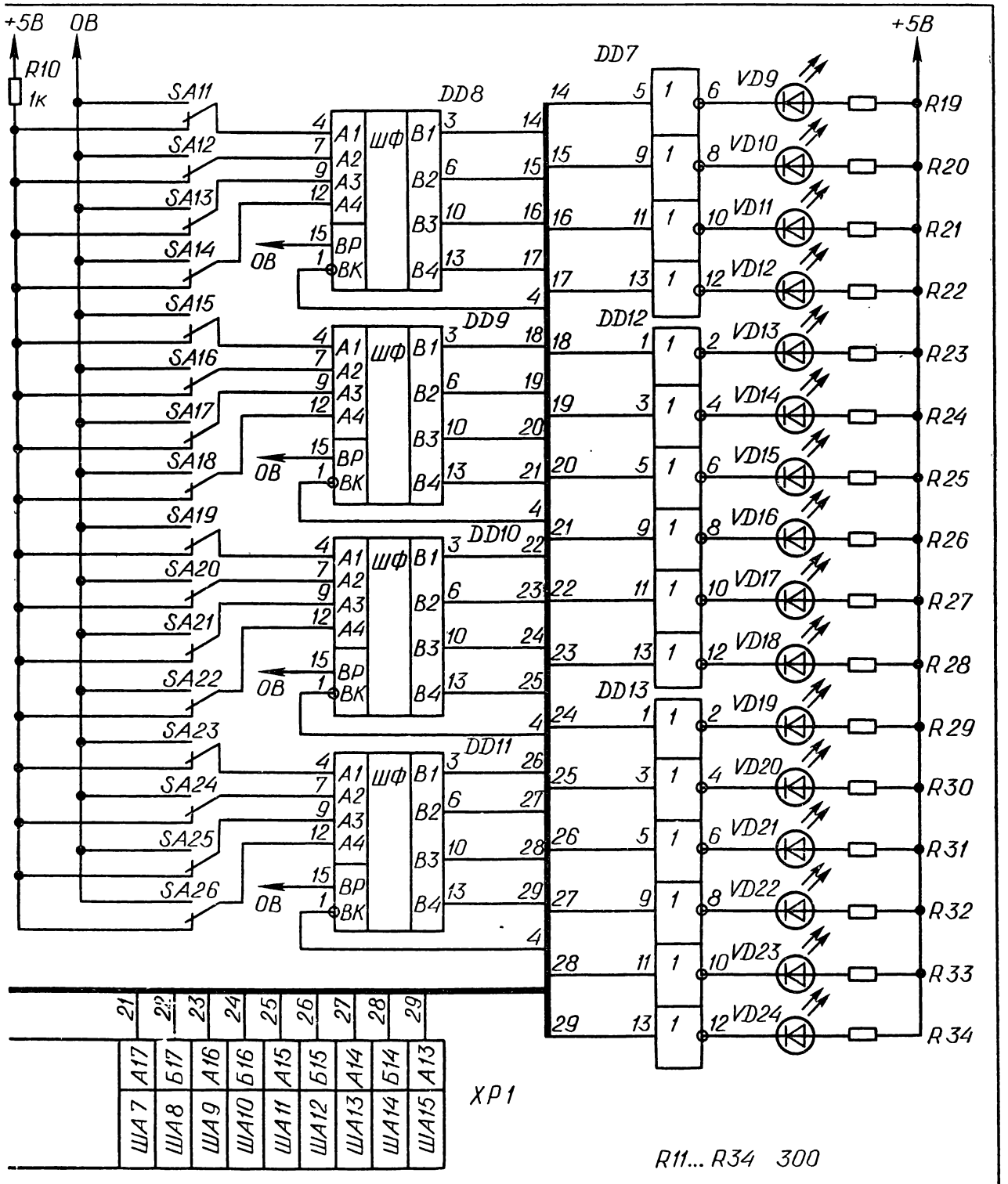


Рис. 15

С1—С4 (выводы С1—С4 на схеме рис. 15 не показаны) и поэтому переключатели SA1—SA8 оказываются отключенными от шины данных. Для их подключения к шине данных на выводы ВР необходимо подать уровень логического нуля. А это выполняется при нажатии на кнопку SB2, когда на выводе 2 низкий логический уровень изменится на высокий. В этом положении SB2 («Запись») переключатели SA1—SA8 подключаются к шине данных и логический уровень напряжения с переключателей (0 или 1) передается в модуль памяти для записи (но при условии, что на выводы $\overline{ВК}$ шинных формирователей подан низкий логический уровень).

Контроль за данными осуществляется с помощью светодиодов VD1—VD8. При отжатой кнопке SB2 светодиоды VD1—VD8 показывают, что хранится в ячейке памяти, адрес которой установлен переключателями SA11—SA26. При нажатии кнопки SB2 светодиоды VD1—VD8 указывают, какие данные записываются в ячейку памяти, адрес которой установлен переключателями SA11—SA26.

Рассмотрим узел пошагового режима. Этот режим осуществляется путем снятия сигнала готовности (ГТ), подаваемого на микропроцессор, в результате чего микропроцессор приостанавливает свою работу и переходит в режим ожидания сигнала ГТ. С помощью кнопки SB1 включают сигнал ГТ только на время выполнения одного машинного цикла команды. В состав узла пошагового режима (рис. 15) входят: сдвоенный триггер DD2, логические элементы DD1.4 и DD3.3 переключатель SA9 («Режим») и кнопка SB1 («Шаг»).

В исходном состоянии, изображенном на рис. 15, переключатель SA9 установлен в верхнее положение («Автоматический режим»). При этом на вывод 4(S) триггера DD2.2 и на вывод 12 элемента DD1.4. подается низкий логический уровень, а на вывод 1(R) триггера DD2.2 с вывода 11 элемента DD1.4 — высокий логический уровень. Согласно таблице истинности триггера К155ТМ2, при нулевом уровне на выводе R и единичном на выводе S триггер устанавливается в одно из своих устойчивых состояний, при котором на вывод 6 выдается уровень логического нуля. Этот уровень инвертируется микросхемой DD3.3 и передается на контакт А5 разъема XP1 общей шины в виде сигнала готовности (ГТ), разрешающего микропроцессору выполнять программу в обычном, автоматическом режиме последовательного исполнения команд.

Если же переключатель SA9 перевести в нижнее положение («Пошаговый режим»), то на вывод 4(S) триггера DD2.2 и вывод 12 элемента DD1.4 будет подан уровень логической единицы. Когда на вывод 13 этого же элемента придет положительный сигнал СИНХРО, выдаваемый микропроцессором в начале каждого машинного цикла, то на выводах 13 и 12 будут действовать уровни логических единиц, а следовательно, на выводе 11 появится уровень логического нуля, передаваемый на вывод 1(R) триггера DD2.2. Поскольку уровни на выводах S и R триггера DD2.2 изменились на обратные, то триггер переходит в другое устойчивое

состояние, при котором на его выходе 6 образуется уровень логической единицы. Этот уровень инвертируется элементом DD3.3 и поступает на провод ГТ шины управления (контакт А5 на разъеме XP1 общей шины) в виде уровня логического нуля.

Под воздействием нулевого уровня сигнала $\overline{\text{ГТ}}$ микропроцессор приостанавливает свою работу и переходит в режим ожидания появления на выводе ГТ уровня логической единицы. Такой уровень кратковременно (на время выполнения одного машинного цикла) выдается при нажатии кнопки SB1. Так как вручную невозможно нажать кнопку на время, исчисляемое микросекундами, то применяется схема электронной автоматики. В исходном состоянии (кнопка SB1 отжата) на вывод 13(R) триггера DD2.1 подается нуль, а на вывод 10(S) — единица. Под воздействием этих уровней триггер DD2.1 устанавливается в состояние, при котором на выход *9* выдается нулевой уровень. Нажатие кнопки SB1 вызывает изменение логических уровней на выводах R и S. На R начинает поступать единица, а на S — нуль. Триггер DD2.1 переходит в другое устойчивое состояние и на его выводе *9* логический уровень изменяется с нулевого на единичный. Это изменение логического уровня передается на вывод C(3) триггера DD2.2 и приводит, согласно таблице истинности К155ТМ2, к перебросу триггера DD2.2 в другое устойчивое состояние. На его выводе *6* появляется нуль, а следовательно, на выводе ГТ — единица, разрешающая микропроцессору возобновить приостановленную работу. Микропроцессор выполняет один машинный цикл команды, а затем в начале второго цикла выдает сигнал СИНХРО. Этот сигнал в виде логической единицы поступает на вывод 13 элемента DD1.4 и вызывает появление нулевого уровня на выводе 11 элемента DD1.4 и на выводе 1(R) триггера DD2.2, что заставляет триггер снова изменить свое устойчивое состояние. На выводе 6 триггера DD2.2 появляется единица, а на проводе ГТ шины управления — нуль. Микропроцессор снова приостанавливает свою работу и переходит в режим ожидания последующего нажатия кнопки SB1. При этом с каждым нажатием кнопки SB1 описанный процесс повторяется, микропроцессор выполняет один машинный цикл команды и ждет следующего нажатия кнопки. Как мы пояснили выше, нажатие кнопки SB1 вызывает изменения логического уровня на входе *C* триггера DD2.2 с нулевого на единичный.

В связи с этим у читателей может возникнуть вопрос: Для чего нужен триггер DD2.1? Почему нельзя изменить логический уровень на входе C с помощью кнопки SB1, не применяя триггера? Дело в том, что замыкание и размыкание контактов кнопки сопровождается явлением дребезга, в результате чего на вход C будет подан не единичный переход напряжения с низкого на высокий уровень, а серия переходов, вызывающая сбой в работе.

Явление дребезга контактов часто служит помехой в работе микропроцессорных схем. Нажимая кнопку какого-либо переключателя, мы обычно полагаем, что его контакты замыкаются мгновенно. На самом же деле при механическом замыкании контактов,

кнопок, тумблеров или переключателей из-за неровностей соприкасающихся поверхностей в течение тысячной доли секунды происходят десятки микросекундных соприкосновений контактов друг с другом, а затем уже один контакт плотно соединяется с другим. Таким образом, при нажатии и отжатии кнопки образуется не один, а несколько десятков коротких (микросекундных) сигналов, посылаемых в какой-либо микропроцессорный узел.

Учитывая, что микропроцессорные устройства работают в режиме обработки коротких (микросекундных) сигналов, они воспринимают сигналы дребезга как рабочие. Если это электронный счетчик, то он при единичном нажатии кнопки счета зафиксирует не одно нажатие, а серию сигналов дребезга, показывая их количество на своем цифровом индикаторе.

Для устранения дребезга контактов применяют как технические, так и программные средства. Из технических средств наиболее часто пользуются дополнительным включением триггера. Если обратиться к схеме рис. 15, то можно понять, что первое же соприкосновение переключателя кнопки SB1 с нижним контактом (при нажатии кнопки) приводит к подаче нулевого уровня на вывод 10 (S) триггера DD2.1, что вызывает немедленное опрокидывание триггера и его переход в другое устойчивое состояние, при котором он уже не реагирует на последующие сигналы дребезга. Снова опрокинуть триггер, то есть перевести его в прежнее устойчивое состояние, сможет только первое соприкосновение переключателя SB1 с верхним контактом (при отжатии SB1), когда нулевой уровень будет подан на вывод 13(R).

Перейдем к рассмотрению элементов узла управления. В состав этого узла входят: кнопка SB3 («Сброс»), переключатель SA10 («Захват»), кнопка SB2 («Чтение—запись») и логические элементы DD1.1.— DD1.3, DD3.1 и DD3.2.

Кнопка SB3 служит для установки счетчика команд микропроцессора в исходное, нулевое состояние, позволяющее микропроцессору выполнять программу с начальной, нулевой команды. Кнопка SB3 не нуждается в противодребезговом устройстве, так как сброс счетчика происходит при первом же соприкасании контактов кнопки, а если касание будет весьма коротким, то при последующем более длинным по времени соединении контактов друг с другом. Сигналы дребезга не влияют на режим работы счетчика, уже установившегося в нулевое состояние.

С помощью переключателя SA10 выполняют запрос микропроцессора о возможности захвата шин адреса и данных для непосредственного обращения к памяти при ее программировании. Переключатель SA10 посылает нулевой уровень сигнала $\overline{ЗХ}$ в микропроцессорный модуль, где сигнал инвертируется и поступает на вывод ЗХ микропроцессора в виде уровня логической единицы. В ответ на этот сигнал микропроцессор выдает сигнал ПЗХ, который тоже инвертируется и приходит на контакт Б5 разъема XP1 общей шины. С этого контакта Б5 сигнал $\overline{ПЗХ}$ передается на выводы $\overline{ВК}$ шинных формирователей DD4, DD5 и DD8—DD11.

Кнопка SB2 служит для формирования сигналов управления чтением ($\overline{\text{ЧТЗУ}}$) и записью ($\overline{\text{ЗПЗУ}}$). Дело в том, что с появлением сигнала ПЗХ шинный формирователь DD11 микропроцессорного модуля (см. рис. 12) отключает управляющие сигналы $\overline{\text{ЧТЗУ}}$, $\overline{\text{ЗПЗУ}}$, $\overline{\text{ЧТВВ}}$ и $\overline{\text{ЗПВВ}}$ от общей шины. Поэтому для управления режимом «Чтение—запись» при программировании памяти приходится формировать свои сигналы $\overline{\text{ЧТЗУ}}$ и $\overline{\text{ЗПЗУ}}$. Это формирование выполняется при помощи кнопки SB2, RS — триггера, собранного на элементах DD1.1 и DD1.2 для устранения дребезга контактов SB2, а также двух логических микросхем DD3.1 и DD3.2 с открытым коллектором (типа К155ЛА8).

Рассмотрим, как формируются сигналы управления чтением ($\overline{\text{ЧТЗУ}}$) и записью ($\overline{\text{ЗПЗУ}}$). В исходном состоянии, когда кнопка SB2 отжата, нулевой логический уровень поступает на вывод 2 элемента DD1.1, а с его вывода 3 в виде уровня логической единицы передается на вывод 5 элемента DD3.1. Так как элемент DD3.1 инвертирует выходной сигнал, то на его выводе 4 образуется сигнал $\overline{\text{ЧТЗУ}}$, передаваемый через контакт Б9 общей шины в модуль памяти для чтения данных. При нажатии кнопки SB2 нулевой уровень подается на вывод 4 элемента DD1.2. Поэтому на его выводе 6 возникает уровень логической единицы, который инвертируется элементом DD3.2 и выдается с вывода 10 в виде сигнала $\overline{\text{ЗПЗУ}}$, имеющего нулевой логический уровень. Микросхема DD3 выбрана с открытым коллектором, что допускает параллельное соединение выходов с аналогичными микросхемами подключенными к общей шине. Элемент DD1.3, подобно шинным формирователям, запрещает или разрешает прохождение через DD3.1 и DD3.2 сигналов $\overline{\text{ЧТЗУ}}$ и $\overline{\text{ЗПЗУ}}$, создаваемых кнопкой SB2 и RS триггером DD1.1—DD1.2. Если на выводе 8 элемента DD1.3 будет логический нуль, то выходы 4 и 10 элементов DD3.1 и DD3.2 перейдут в высокоимпедансное состояние и отключат сигналы кнопки SB2 от общей шины.

Ознакомление с литературой [24, 30, 31, 37, 45, 63, 66] облегчит изучение принципа работы модулей микроЭВМ.

3.2. СБОРКА, НАЛАЖИВАНИЕ И МОДЕРНИЗАЦИЯ

Советы по сборке микроЭВМ предлагаются читателю в нескольких вариантах, позволяющих творчески выбрать наиболее приемлемый (в зависимости от уровня знаний, опыта, наличия деталей, оборудования, рабочего места и др.). Вместе с тем среди таких рекомендаций, рассчитанных на сознательное исполнение, встречаются правила, которые нельзя понимать двояко, а нужно выполнять однозначно и точно. К таким правилам относятся, например, правила по технике безопасности, а также ряд предостережений, позволяющих заранее принять меры, которые исключали бы выход из строя элементов и узлов микроЭВМ из-за неправильных действий.

Поэтому прежде всего кратко ознакомимся именно с такими предупреждающими требованиями. Поговорим о технике безопасности.

Несмотря на то, что источник питания модулей микроЭВМ имеет низкое (5 В) и, следовательно, безопасное напряжение, это не исключает возможности поражения электрическим током. Наличие в блоке питания силового трансформатора, первичная обмотка которого подключена к напряжению 220 В, а также пользование электрифицированным инструментом и измерительными приборами, питающимися от сети 220 В, создают дополнительную опасность электротравматизма. Кроме того, всегда следует помнить о возможности получения ожогов при пайке и случайных ранениях при выполнении слесарных работ.

Поэтому для создания безопасных условий сборки микроЭВМ необходимо соблюдать такие правила по технике безопасности:

1) к работе по сборке и налаживанию микроЭВМ допускаются кружковцы, хорошо знающие правила техники безопасности и получившие дополнительный инструктаж в зависимости от выполняемой работы;

2) в помещении, где работает кружок, на видном месте должны быть вывешены правила техники безопасности;

3) рабочее место должно быть оборудовано вытяжной вентиляцией, пол покрыт резиновым ковриком, а само рабочее место — линолеумом;

4) освещенность рабочего места должна соответствовать установленным нормам;

5) в помещении должна быть аптечка;

6) электрические паяльники должны работать при напряжении не более 36 В и иметь специальную подставку, исключающую возможность случайного касания его нагретой части;

7) в случае использования технологии травления печатных плат необходимо предусмотреть меры, исключающие ожоги и выделения вредных газов от применяемых химических реактивов;

8) категорически запрещается проводить монтаж, ремонт и налаживание аппаратуры, находящейся под напряжением, превышающим 36 В. Корпус такой аппаратуры должен быть надежно заземлен. Все работы с напряжением выше 36 В следует выполнять в резиновых перчатках, а под ногами иметь резиновый коврик.

Радиолюбителям, производящим сборку микроЭВМ в домашних условиях, тоже необходимо соблюдать перечисленные правила.

Теперь о влиянии электростатических зарядов на микросхемы.

Большинство микросхем (особенно типа К—МОП) боятся воздействия электростатических зарядов, попадающих на их выводы. Заряды статического электричества легко возникают при трении многих синтетических материалов, широко используемых в быту, поэтому необходимо принять меры, предупреждающие попадание зарядов на выводы микросхем. С этой целью следует соблюдать следующие правила:

1) хранить микросхемы завернутыми в фольгу;

2) не касаться выводов микросхем металлическими предметами, щупами тестера или пальцами;

3) жало паяльника, корпуса измерительных приборов, рабочий инструмент (пинцет, плоскогубцы), а также полоска шины питания 0 В на плате монтируемого модуля должны быть заземлены при помощи гибкого провода, подключенного к заземлению через резистор 1 Мом;

4) на руки необходимо одевать металлические манжеты, соединенные с заземлением через резистор 1 Мом. Манжеты сначала одевают на руки, а потом соединяют с заземлением;

5) работать с микросхемами нужно в хлопчатобумажной одежде (халате) и обуви на кожаной подошве;

6) монтаж микросхемы, установленной на плате, нужно начинать с монтажа выводов питания (сперва 0 В, а затем +5 В). Особую осторожность следует соблюдать при подключении проводников к выводам микропроцессора.

Помимо электростатических зарядов, для микросхем опасен перегрев. Если при пайке допустить перегрев выводов микросхемы, то это приводит к резкому ухудшению ее параметров, а часто и выходу из строя. Поэтому следует применять миниатюрный паяльник, легкоплавкий припой, а саму пайку выполнять как можно быстрее. Не следует паять два вывода, расположенных рядом. Выполнив пайку одного вывода, нужно переходить к пайке вывода, расположенного на противоположной стороне корпуса микросхемы.

При включении и выключении питающих напряжений микропроцессора К580 и некоторых типов микросхем (например, К565РУ3) требуется соблюдение определенного порядка, иначе микропроцессор может выйти из строя.

Завод-изготовитель устанавливает следующий порядок включения питающих напряжений для МП—К580: первым должно включаться напряжение —5 В, за ним +5 В и, наконец, +12 В. Выключение производится в обратном порядке, а именно — первым выключается напряжение +12 В, потом +5 В и в конце —5 В. Допускается одновременное включение или выключение всех питающих напряжений. Если по какой-либо причине отключилось одно из питающих напряжений, то немедленно должны быть выключены и остальные. Особенно опасно пропадание напряжения —5 В.

Для микросхем недопустимы переполюсовки, в виде ошибочного включения вывода «+» питающего напряжения вместо вывода «—», и наоборот.

Известную опасность создает отклонение величины питающего напряжения. Допускается изменение напряжения в пределах $\pm 5\ \%$. При меньшем напряжении микросхема будет давать сбои в работе, при большем — может выйти из строя. Выводы микросхем уязвимы к механическим воздействиям. Поэтому желательно лишний раз их не перегибать. В случае отлома вывода у основания корпуса подпаять вывод почти невозможно.

Перечисленные меры предосторожности нужно усвоить и учитывать при сборке микроЭВМ.

Советы по сборке модулей

Собрать три модуля простейшей микроЭВМ, описанной в разделе 3.1, сравнительно несложно. Для этого нужно заготовить гетинаксовые пластинки (платы), укрепить на них необходимые детали и соединить выводы деталей между собой так, как это показано на принципиальной схеме. Таким образом, микроЭВМ может собрать даже малоопытный начинающий кружковец или радиолюбитель. А вот наладить собранную микроЭВМ, а потом попытаться ее усовершенствовать — дело довольно сложное. Налаживание всегда связано с поиском неисправностей в работе, которые неизбежно возникают из-за ошибочных соединений при выполнении монтажа. Кроме того, из большого числа деталей, входящих в схему микроЭВМ, несколько деталей могут оказаться неисправными. Поэтому поиск места и определения характера неисправности требуют основательных знаний об устройстве каждой детали и принципе действия всей схемы в целом. Еще больше знаний и умения требуется для модернизации микроЭВМ — превращению трех собранных модулей простейшей микроЭВМ в современный персональный компьютер.

Как было указано, для облегчения сборки и налаживания микроЭВМ рекомендуется собирать в два этапа.

На первом этапе необходимо:

основательно повторить электронику по школьному курсу физики или с помощью популярной литературы, рекомендованной в главе 2;

ознакомиться с двоичным счислением, логическими операциями и принципом работы логических микросхем;

изучить принцип работы микропроцессора и микроЭВМ;

пользуясь сведениями из раздела 3.2, а также учитывая наличие деталей, нужных для сборки модулей микроЭВМ, необходимо уточнить принципиальные схемы модулей;

наметить конструктивное расположение деталей на платах модулей;

укрепить детали на платах и выполнить монтаж.

Одновременно с выполнением этих работ необходимо уделять достаточное внимание программированию. Всегда нужно помнить о том, что в работе микроЭВМ главным является программа. Аппаратная часть (модули, внешние устройства) являются лишь техническим средством, подспорьем для выполнения составленной программы. Поэтому, монтируя модули микроЭВМ, необходимо одновременно изучать систему команд МП—К580 и осваивать язык ассемблера, выполняя достаточное количество примеров написания коротких учебных программ на машинном языке и языке ассемблера. Следует иметь в виду, что сборка микроЭВМ выполняется не только ради того, чтобы иметь собственный компьютер. МикроЭВМ изготавливается прежде всего для того, чтобы познать принцип ее работы и получить практические навыки написания и отработки программ. Опыт показывает, что те, кто

собрал микроЭВМ своими руками, наладил ее, а также написал для нее и отработал достаточное количество программ, приобретают глубокие и прочные знания сложной вычислительной техники и программирования.

Первый этап сборки микроЭВМ сводится к выполнению следующих действий:

1) выбор и творческая доработка принципиальных схем модулей в зависимости от поставленных задач, имеющегося опыта, наличия деталей и оборудования рабочего места;

2) вычерчивание выбранных принципиальных и монтажных схем модулей;

3) составление спецификаций деталей, материалов, инструмента, приборов, справочных данных и др.;

4) решение конструктивных вопросов по размещению и монтажу микросхем на платах;

5) предварительная проверка микросхем и других деталей, входящих в схему модулей;

6) установка микросхем и деталей на плате модулей;

7) монтаж модулей и общей шины;

8) подготовка блока питания;

9) предварительная проверка работоспособности модулей без включения микропроцессора;

10) испытание и налаживание микроЭВМ с включенным микропроцессором, изучение и неукоснительное выполнение правил техники безопасности.

Прежде чем давать советы по выполнению перечисленных работ, укажем варианты схем модулей и возможную замену деталей. Эти сведения будут полезны при творческом выборе принципиальных схем модулей.

Если читатель ограничен количеством и ассортиментом микросхем и поэтому вынужден ставить перед собой задачу-минимум собрать хоть какую-нибудь, пусть даже самую простую, но лишь бы действующую микроЭВМ, позволяющую вводить написанные программы и выполнять их, то схему микропроцессорного модуля можно значительно упростить, а многие типы микросхем, входящих в состав как микропроцессорного, так и других модулей, заменить такими, какие имеются в наличии.

Начнем с узла тактового генератора микропроцессорного модуля. Если нет кварцевого резонатора Z1 (рис. 12), то его можно заменить колебательным контуром, состоящим из индуктивности L1 и конденсатора C1, подключенных к микросхеме DD1 так, как это показано на рис. 16. Катушка L1 наматывается на картонном каркасе диаметром 10 мм и имеет 32 витка провода ПЭВ-1 0,3. Отсутствие кварцевой стабилизации частоты тактового генератора приведет к тому, что микроЭВМ не сможет обеспечить точного выполнения временных задач, то есть не сможет, служить, например, секундомером, таймером или электронными часами. На выполнение других видов задач, не связанных с точным отсчетом времени, отсутствие в схеме кварца Z1 не сказывается.

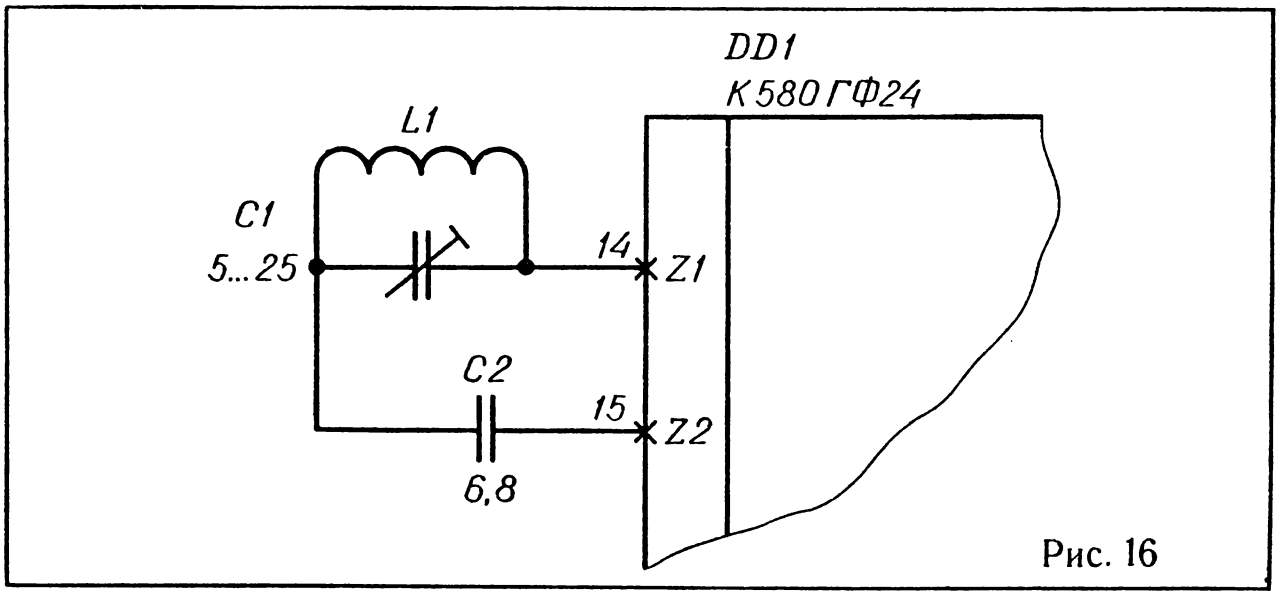


Рис. 16

Специальную микросхему типа К580ГФ24 можно заменить набором широко распространенных микросхем типа К155ЛА3, К155ЛА8, К155ТМ2 и др. При этом вместо одного корпуса микросхемы (К580ГФ24) придется установить пять-шесть корпусов других микросхем, но так как это выполняется не в конструкции для массового тиражирования, а в одном экземпляре личной микроЭВМ, то такая замена допустима.

На рис. 17 приведена схема тактового генератора, собранного из отдельных элементов. Микросхемы DD1.1, DD1.3, DD2.1 совместно с конденсаторами С5—С6 образуют задающий генератор, вырабатывающий прямоугольные импульсы с частотой 9 Мгц. Путем деления этой частоты с помощью элементов DD4, DD1.2, DD2.3, DD1.4 и DD2.4 на выходе 8 элемента DD2.3 и выходе 11 элементе DD2.4 получают тактовые сигналы Ф1 и Ф2, имеющие тактовую частоту, равную 1 Мгц. Так как на входы 22 и 15 микропроцессора (рис. 12) эти сигналы должны быть поданы, имея амплитуду порядка 10—12 В, то установлены транзисторные усилители VT1 и VT2. Цепочки С7—R7 и С8—R8 служат для улучшения формы импульсов, поступающих на базы транзисторов. В связи с тем, что задающий генератор тактового узла не имеет кварцевой стабилизации частоты, следует учесть временные ограничения, о которых сказано выше. Если возникнет необходимость решения задач, связанных с точным отсчетом времени, то в схему тактового генератора можно добавить кварцевый резонатор, так как это сделано в микроЭВМ «Микро-80» [24].

Триггеры DD5 и логические элементы DD3.1—DD3.3 выполняют дополнительную роль, формируя сигнал СТРОБ и управляя временем выдачи сигналов ГТ и СБР, поступающих с общей шины в микропроцессор. Конденсатор С1 и резистор R1 обеспечивают автоматический сброс счетчика команд микропроцессора при включении источника питания.

На схеме рис. 17 возле каждого входа и выхода, изображенного в виде стрелки (СБР, ГТ, СТРОБ, СИНХРО, Ф1, Ф2, +5 В, ОВ и +12 В), рядом со стрелкой в скобках указан номер вывода

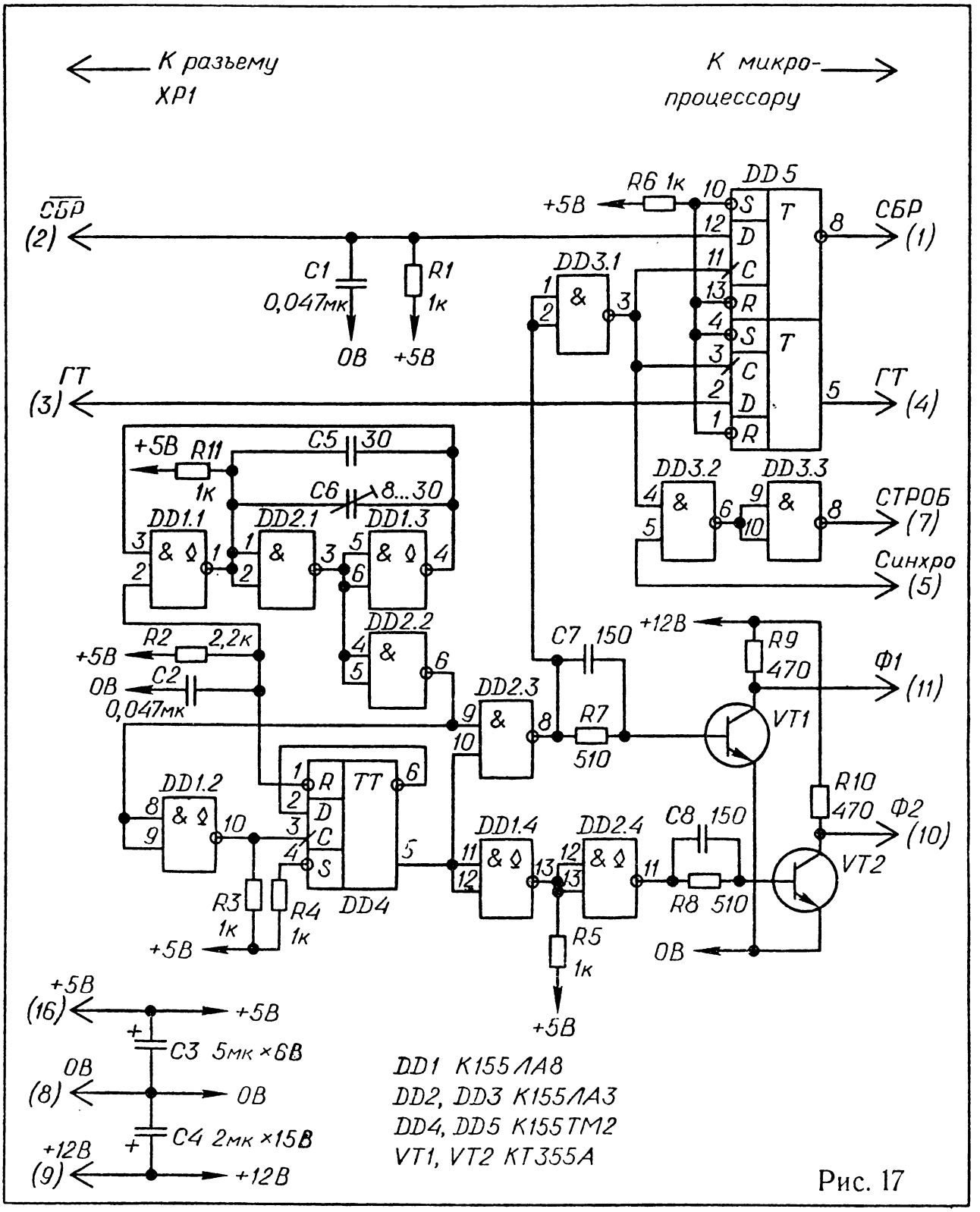


Рис. 17

микросхемы К580ГФ24, который заменяет собой этот выход или вход.

Все выводы со стрелкой, изображенные на рис. 17 слева, подключаются к разъему XP1 общей шины, а те, которые нарисованы справа, к микропроцессору.

Рассмотрим вариант схемы канала формирования управляющих сигналов ЧТЗУ, ЗПЗУ, ЧТВВ и ЗПВВ. Укажем, что имеются микроЭВМ, которые обходятся без формирования упомянутых управляющих сигналов. У этих микроЭВМ устройство ввода или вывода выполняется в виде условной ячейки памяти, номер

(адрес) которой лежит в неиспользуемой области памяти, не соответствующей адресу рабочих ячеек ОЗУ или ПЗУ. Микропроцессор при обращении к устройству ввода (вывода) устанавливает на адресной шине адрес ячейки памяти, условно принятой за устройство ввода (вывода), и по сигналу ЧТЗУ (ЗПЗУ) обменивается с ней данными. При этом в качестве сигналов ЧТЗУ и ЗПЗУ используются сигналы ПМ и ВД, формируемые самим микропроцессором.

Ниже будет подробно описан такой способ ввода—вывода данных. Если воспользоваться им, то можно полностью исключить из схемы микропроцессорного модуля (рис. 12) всю верхнюю правую часть, представляющую собой канал управляющих сигналов $\overline{\text{ЧТЗУ}}$, $\overline{\text{ЗПЗУ}}$, $\overline{\text{ЧТВВ}}$ и $\overline{\text{ЗПВВ}}$.

Временно отказаться от канала четырех управляющих сигналов можно и тогда, когда читатель пожелает досрочно увидеть плоды своего труда и поставит перед собой задачу-минимум: получить микроЭВМ, позволяющую работать пока только с памятью (допускающую ввод с пульта управления в память написанную программу и ее выполнение в пошаговом или автоматическом режимах). В этом случае для управления процессами записи и чтения нужны только два управляющих сигнала ЗПЗУ и ЧТЗУ, получить которые можно из сигналов ВД и ПМ так, как это показано на рис. 18. При этом нужно только не забыть соединить накоротко выводы 12 и 13 шинного формирователя DD12 (рис. 12), чтобы сигнал ПЗХ мог пройти с вывода 8 микросхемы DD3.4 на разъем XP1.

В нижней правой части схемы рис. 12 изображен канал коммутации сигналов адресной шины. В этом канале установлено четыре шинных формирователя (DD5—DD8) типа К589АП16. Эти

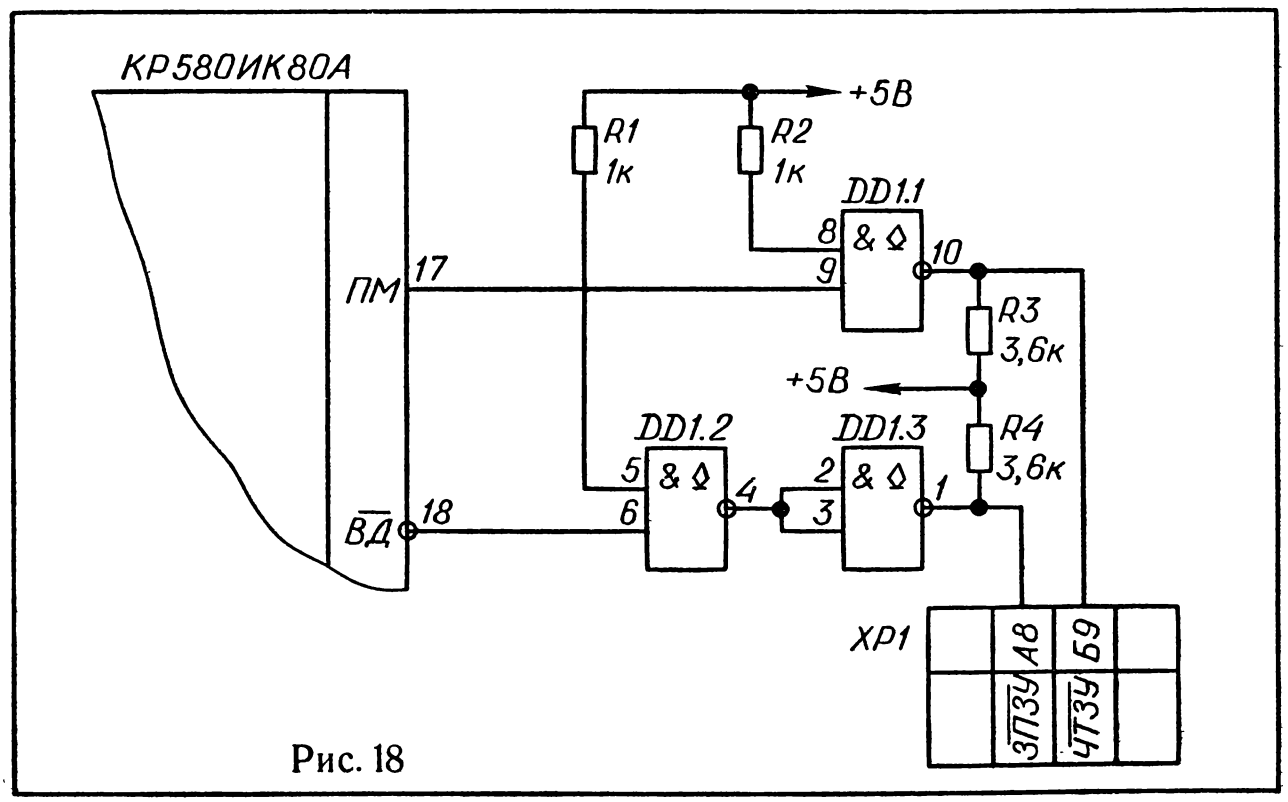


Рис. 18

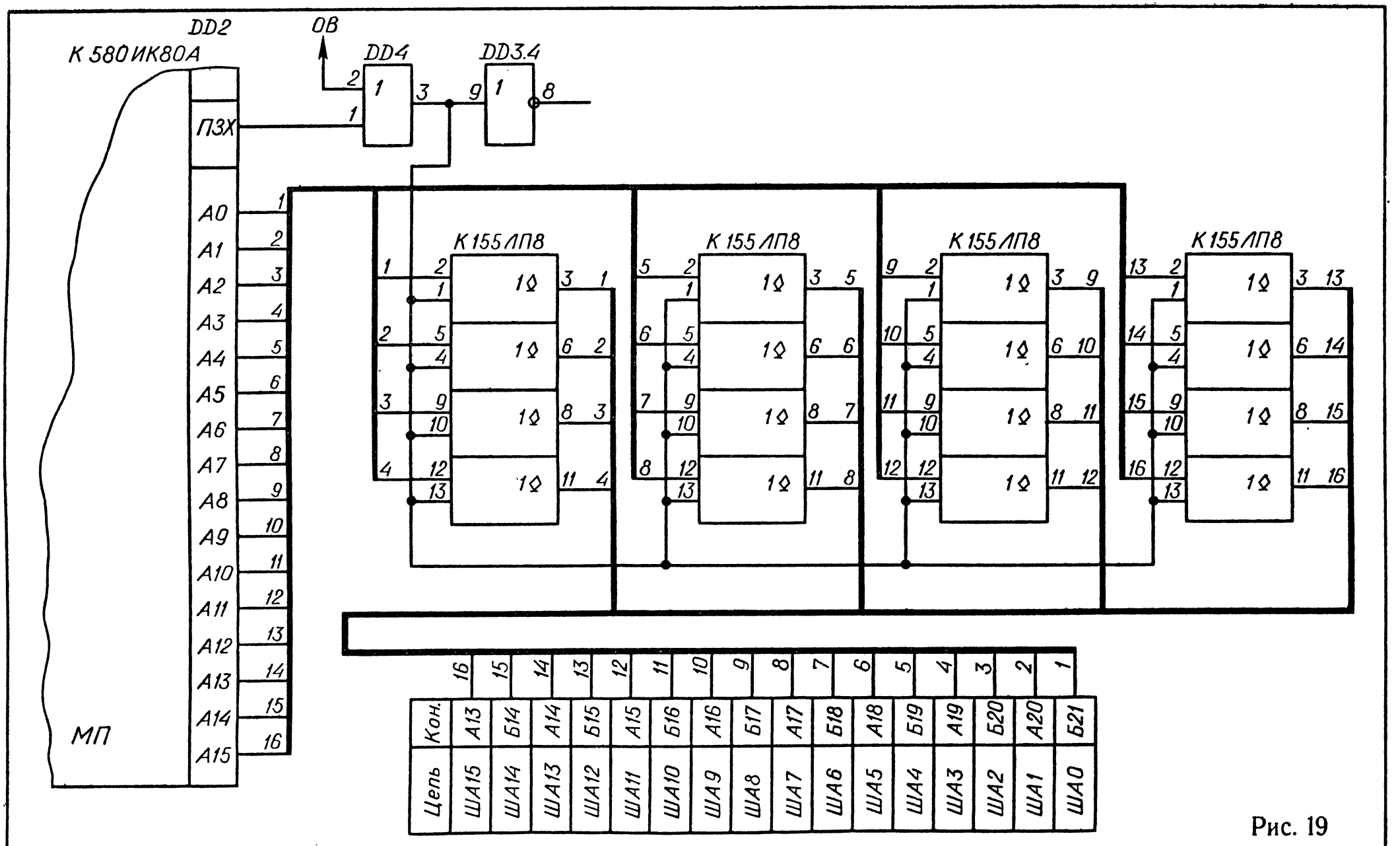

DD2
К 580ИК80А
ОВ
DD4
DD3.4
ПЗХ
А0
А1
А2
А3
А4
А5
А6
А7
А8
А9
А10
А11
А12
А13
А14
А15
МП
К 155ЛП8
К 155ЛП8
К 155ЛП8
К 155ЛП8
Цепь
Кон.
ША15
А13
ША14
Б14
ША13
А14
ША12
Б15
ША11
А15
ША10
Б16
ША9
А16
ША8
Б17
ША7
А17
ША6
Б18
ША5
А18
ША4
Б19
ША3
А19
ША2
Б20
ША1
А20
ША0
Б21

Рис. 19

формирователи можно выполнить на микросхемах типа К155ЛН6, К155ЛП8, К155ЛП10 или К155ЛП11. На рис. 19 приведена схема адресного канала, собранная на микросхемах К155ЛП8. В качестве шинного формирователя очень удобно применять специальную микросхему типа К580ВА86. Однако эта микросхема выпускается в ограниченном количестве и ее трудно приобрести.

Если нет перечисленных выше типов микросхем, то адресный канал можно собрать на популярных логических микросхемах типа К155ЛА8 и К155ЛН1 так, как это показано на рис. 20. Слева от этого рисунка (рис. 20, *а*) изображена микросхема К589АП16,

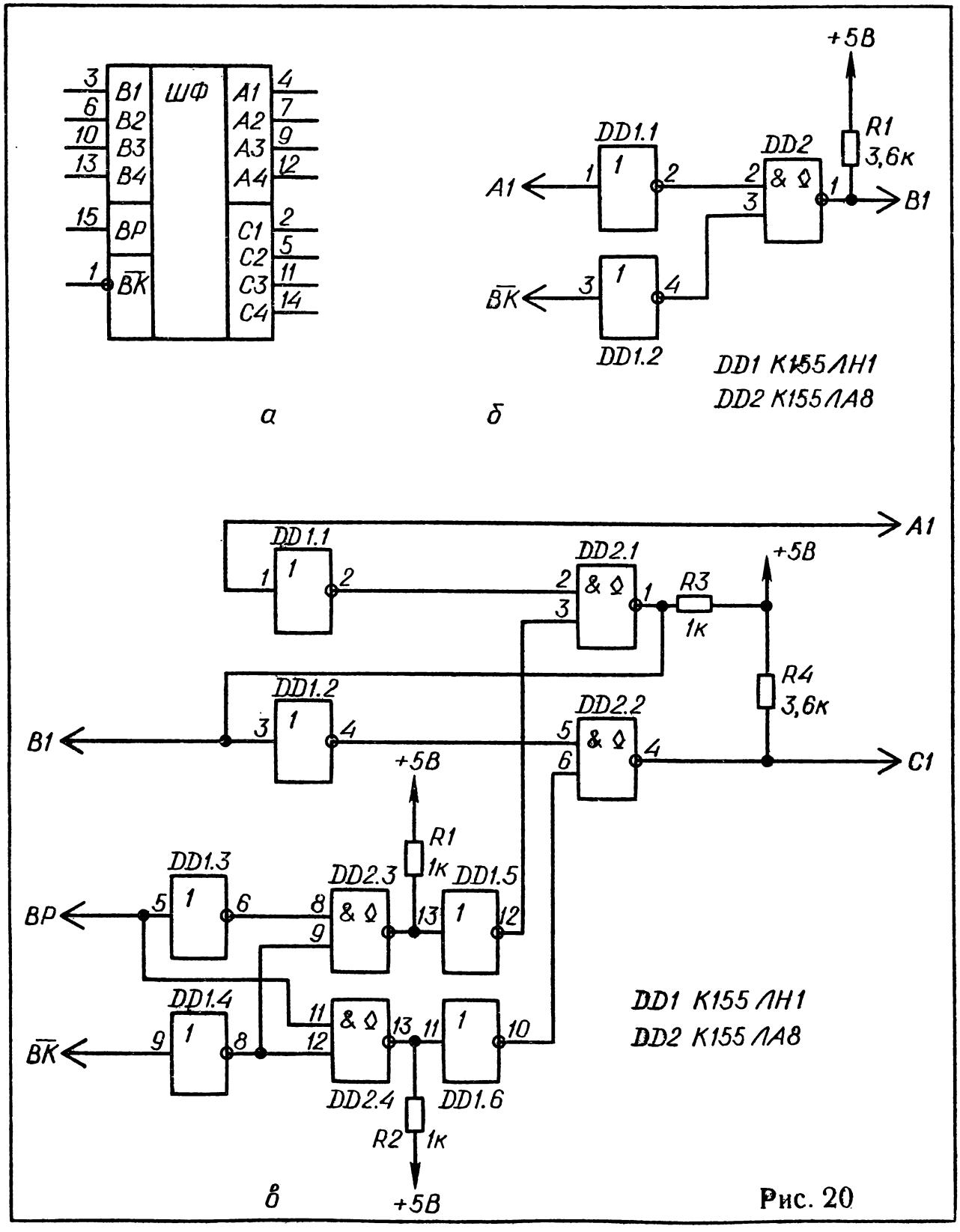


Рис. 20

а справа представлена схема (рис. 20, б) замены одного канала (А1—В1) К589АП16. Следовательно, чтобы заменить всю микросхему К589АП16, нужно собрать четыре схемы, приведенных на рис. 20, *б*. Вместо микросхемы К155ЛН1 можно использовать К155ЛА3 или К155ЛА8. При этом неиспользуемые входы (эти микросхемы двухвходовые, а разрешается подключать только один вход) следует подключить через резистор R = 1 КОм к выводу источника питания +5 В.

На рис. 20, *в* показано, как заменить шинные формирователи DD9 и DD10, установленные в канале шины данных рис. 12. Здесь для замены одного корпуса микросхемы К589АП16 также нужно собрать четыре схемы, изображенные на рис. 20, *в*.

В настоящее время промышленность выпускает микропроцессор типа К580ИК80А в пластмассовом или керамическом корпусе, имеющем 40 выводов. Несколько лет тому назад микропроцессор К580 выпускался в корпусе, имеющем 48 выводов, из которых восемь выводов не использовались.

Если читатель приобретет микропроцессор, имеющий 48 выводов, то следует иметь в виду, что номера его выводов не соответствуют номерам выводов микропроцессора, имеющего 40 выводов. В Приложении приведены справочные данные для обоих типов микропроцессоров, пользуясь которыми можно выполнить правильное подключение выводов своего микропроцессора.

Чтобы предохранить дорогостоящий микропроцессор от возможного повреждения из-за ошибок в монтаже модулей или в результате неправильно поданных сигналов при первоначальном включении и налаживании микроЭВМ, целесообразно включать микропроцессор в схему модуля при помощи переходной панельки, установив на плате микропроцессорного модуля панельку, имеющую 40 гнезд, и включая в эту панельку микропроцессор только после тщательной проверки всех напряжений и сигналов на ее выводах. Хотя переходные контакты панельки могут стать причиной дополнительных неисправностей в работе модуля, такая конструкция все же оправдывает себя при налаживании микроЭВМ.

Собирая сложный микропроцессорный модуль в условиях кружка, иногда бывает целесообразно выполнять конструкцию модуля в виде отдельных съемных узлов, смонтированных на небольших платах, заканчивающихся штыревыми разъемами, а на плате модуля устанавливать гнездовые разъемы для включения узлов в схему. Такая конструкция сборного модуля позволяет производить изготовление узлов параллельно отдельными группами кружковцев, состоящими из 2—3 человек. Таким образом, микропроцессорный модуль изготовляется в кружке во много раз быстрее, а главное — он допускает возможность последующей замены деталей, модификацию и переделку в соответствие с новыми создавшимися условиями.

Разъемная конструкция модуля может оказаться полезной и для начинающего радиолюбителя. Смонтировав отдельный узел, он сможет понести его в местный радиоклуб и там наладить.

Рассмотрим, как можно заменить микросхемы в модулях памяти и управления другими типами. Шинные формирователи этих модулей допускают замену микросхем К589АП16 на микросхемы, типы которых указаны для шинных формирователей микропроцессорного модуля.

Вместо микросхем К565РУ2 модуля памяти можно устанавливать другие типы, рекомендуемые заводом-изготовителем для ОЗУ статического типа. Особенности их включения в схему приводятся в заводском паспорте и в справочнике по полупроводниковым приборам [30].

Вместо переключателей типа МТ1, установленных в модуле управления, можно использовать любые другие переключающие тумблеры или кнопки, позволяющие включать в схему одно из двух напряжений (высокого или низкого уровня). Светодиоды также можно заменить светодиодами любого типа или, в крайнем случае, обычными лампочками для карманного фонаря. При этом лампочки лучше подключать к шинам с помощью транзисторных ключей, а не логических элементов. Светодиоды, лампочки, логические микросхемы и транзисторные ключи цепей индикации можно питать отдельно нестабилизированным напряжением +5 В.

В заключение заметим, что замена типов микросхем может привести к значительному изменению величины тока, потребляемого модулями. Поэтому, пользуясь табл. 8 Приложения, необходимо подсчитать общий расход тока питания вновь установленных микросхем, определить суммарно потребляемый ток каждым модулем и внести соответствующие коррективы в схему блока питания, если этот ток будет превышать данные, указанные в табл. 3.2.

Выбор блока питания

Приступая к сборке микроЭВМ, следует выяснить возможность полного или частичного использования имеющихся (в кружке или дома) источников питания в качестве блока питания микроЭВМ. Для этого нужно знать, какие требования предъявляются к блоку питания микроЭВМ с тем, чтобы определить, смогут ли имеющиеся источники питания их удовлетворить.

Помимо специфических требований включения и выключения питающих напряжений в определенном порядке (о чем говорилось в начале этого раздела), следует знать, какие напряжения и какой ток должен выдавать блок питания. В табл. 3.2 указано потребление тока модулями и всей микроЭВМ в целом. При этом имеется в виду, что микроЭВМ будет состоять из минимального количества модулей. Если же предполагается использовать дополнительные модули, расширяющие возможности микроЭВМ, то при выборе блока питания необходимо учесть ток, потребляемый этими модулями.

Напряжения источников питания должны быть стабилизированы, в противном случае будут иметь место значительные колебания

Таблица 3.2

Модуль микроЭВМ	Потребление тока (в мА) источниками		
	+5 В	+12 В	−5 В
Микропроцессорный	600	100	10
Памяти	450	—	—
Управления	650	—	—
Всего:	1700	100	10

напряжения при изменении тока нагрузки и напряжения питающей сети, а это может вызвать сбои в работе микроЭВМ. Если имеющийся блок питания не полностью удовлетворяет предъявленным требованиям, то его, пользуясь описанием блоков питания, которые часто публикуются в журналах «Радио» или «Моделист-конструктор» (см., например [45, 52]), следует доработать или смонтировать заново нужный блок питания.

Блок питания, удовлетворяющий требованиям современных микроЭВМ, устройство довольно сложное и дорогостоящее. Объясняется это тем, что он имеет очень высокую надежность, снабжен защитой от коротких замыканий и превышения максимально допустимого тока, выдает хорошо стабилизированные напряжения, автоматически выполняет установленный порядок включения и выключения источников напряжения, а также автоматически выключает все источники при аварийном выключении одного из них.

Не будем предъявлять к выбору блока питания для нашей микроЭВМ жесткие требования. Важно только, чтобы блок питания, согласно табл. 3.2, выдавал нужные стабилизированные напряжения и чтобы они включались и выключались одновременно. Желательно иметь простейшее устройство, обеспечивающее выключение питающих напряжений +5 В и +12 В в том случае, если почему-либо выключилось напряжение −5 В. Это можно достигнуть, смонтировав обычное реле, питающееся напряжением −5 В, которое выключит напряжение +5 В и +12 В или же выключит сеть переменного тока напряжением 220 В, если исчезнет напряжение −5 В. Малогабаритное реле желательно разместить на плате микропроцессорного модуля, а его обмотку подключить непосредственно к соответствующим выводам панельки микропроцессора, чтобы исключить возможность нефиксируемого пропадания напряжения −5 В на переходных контактах разъемов и пайках схемы. Выводы контактов реле следует подключить к резервным контактам (А3, Б3, А2 и А6) разъема XP1 микропроцессорного модуля и скоммутировать их на свободные гнезда (А6, А7, Б6 и Б7) разъема XS5 блока питания для отключения напряжения +5 В и +12 В в случае выключения напряжения −5 В. Выходы всех напряжений блока питания должны быть подключены к разъему общей шины так, как это показано на рис. 11.

Во избежание наводок импульсных помех провода, соединяющие блок питания с общей шиной, нужно делать как можно короче. Для уменьшения воздействия помех между проводом ОВ и проводами питающих напряжений (+5 В, —5 В, +12 В) необходимо устанавливать низкочастотные и высокочастотные блокировочные конденсаторы. Такие конденсаторы устанавливают на выходе блока питания, а также на входе шины питания каждого модуля микроЭВМ. Емкость низкочастотных электролитических конденсаторов должна быть от 10 до 50 мкФ, а высокочастотных керамических порядка 0,05—0,1 мкФ.

Если для питания микроЭВМ использовано несколько отдельных источников, то нужно предусмотреть установку общего выключателя, позволяющего отключать и включать все напряжения одновременно.

Порядок сборки модулей

Если модули учебной микроЭВМ собираются в кружке, то кружок рекомендуется разделить на 4 группы по 2—3 человека в каждой. Первая группа, состоящая из наиболее подготовленных кружковцев, собирает микропроцессорный модуль, вторая — модуль памяти, третья — модуль управления, а четвертая — монтирует общую шину и подготавливает блок питания. При этом возможны варианты. Если модуль микропроцессора конструктивно выполняется в виде съемных узлов, то первая группа собирает узел тактового генератора, вторая — узел коммутации адресной шины, третья — узел коммутации шины данных и четвертая — узел микропроцессора, куда входит и монтаж гнездовых разъемов, установленных на плате для подключения всех его узлов. Затем группы получают задания по сборке остальных модулей.

Сборка микроЭВМ, выполняемая радиолюбителем в домашних условиях, обычно начинается со сборки того модуля, который полностью обеспечен деталями, указанными на принципиальной схеме. Если будут иметь место непредвиденные задержки в сборке этого модуля, переключаются на монтаж общей шины и подготовку блока питания.

Приступая к сборке модулей в соответствии с перечнем основных этапов, прежде всего следует определить, по каким схемам будут собираться модули. Не нужно ограничиваться простым копированием схем, указанных в настоящей книге. К тому же у читателя может не оказаться в наличии какой-либо микросхемы, приведенной в описании. Поэтому, проявляя творческий подход к выбору схемы, бывает весьма полезно и поучительно прочитать дополнительно описания нескольких аналогичных микроЭВМ [8, 24, 44, 45], а затем, окончательно выбрав принципиальную схему модуля, вычертить эту схему на листе бумаги с изображением всех соединительных цепей (в том числе и цепей питания, не показываемых на принципиальных схемах). Вычерчивать схемы модулей нужно сознательно, ясно представляя себе, какой сигнал

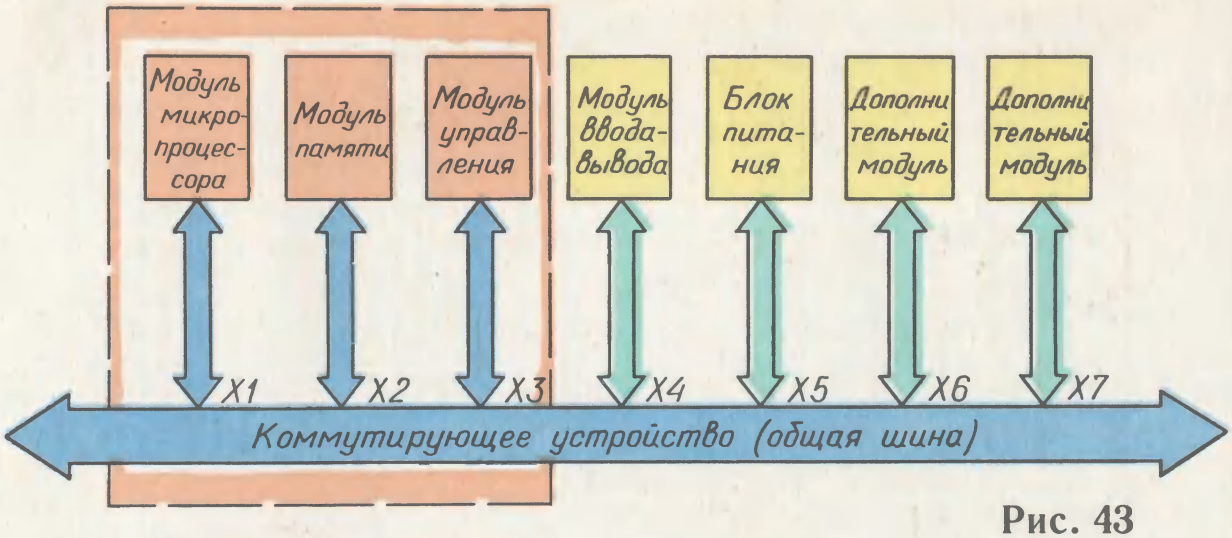


Рис. 43

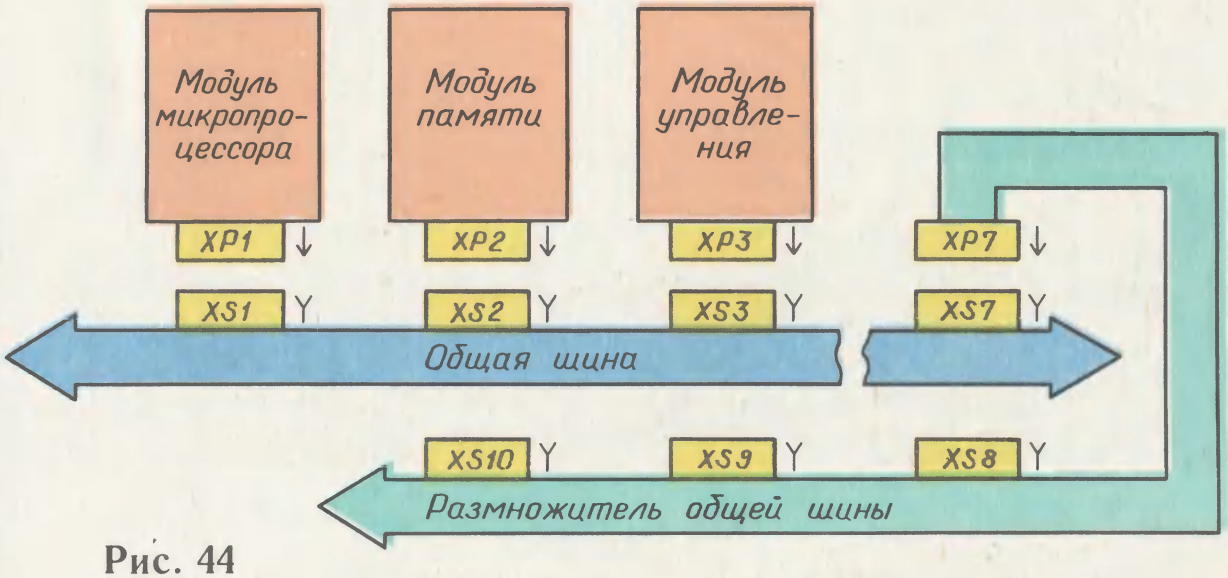


Рис. 44

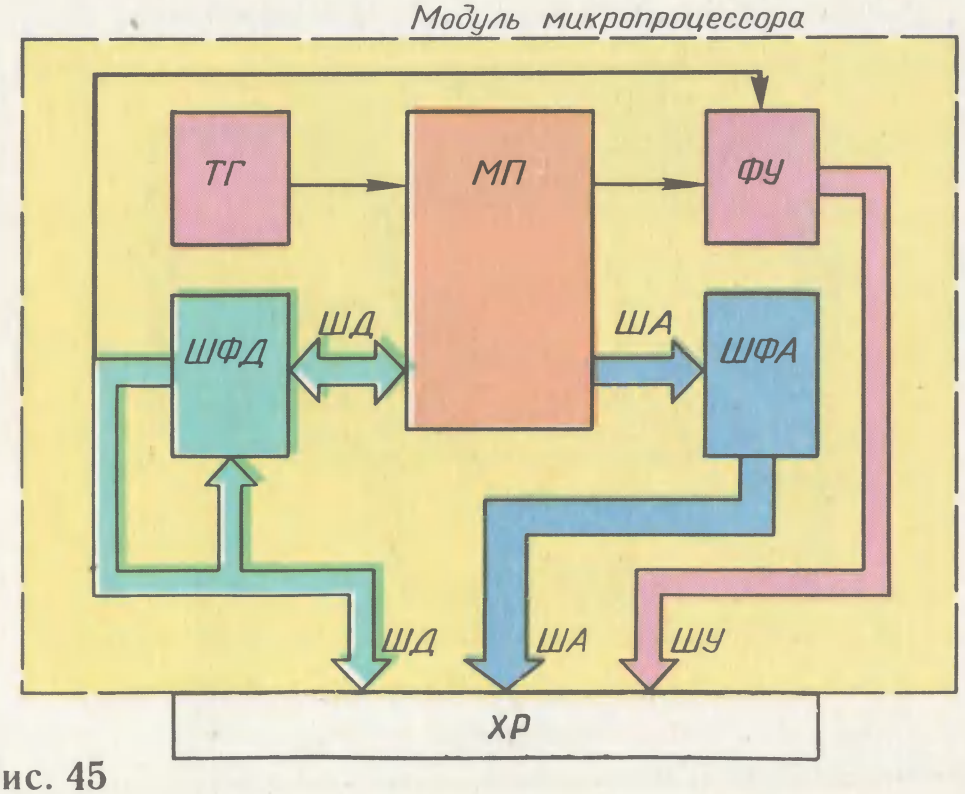


Рис. 45

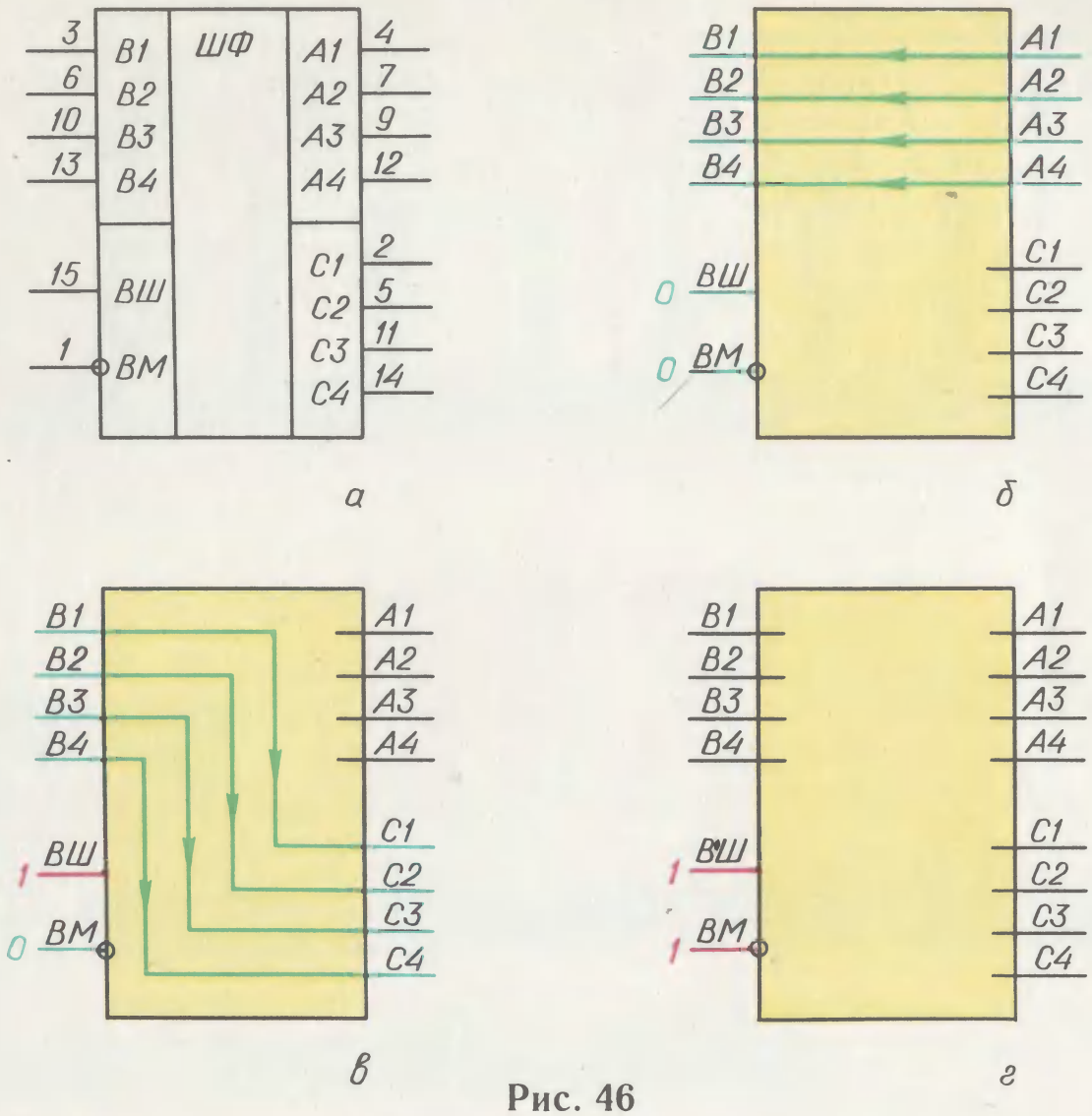

3
6
10
13
15
1
В1
В2
В3
В4
ВШ
ВМ
ШФ
А1
А2
А3
А4
С1
С2
С3
С4
4
7
9
12
2
5
11
14
а
В1
В2
В3
В4
ВШ
ВМ
0
0
А1
А2
А3
А4
С1
С2
С3
С4
б
В1
В2
В3
В4
ВШ
ВМ
1
0
А1
А2
А3
А4
С1
С2
С3
С4
в
В1
В2
В3
В4
ВШ
ВМ
1
1
А1
А2
А3
А4
С1
С2
С3
С4
г

Рис. 46

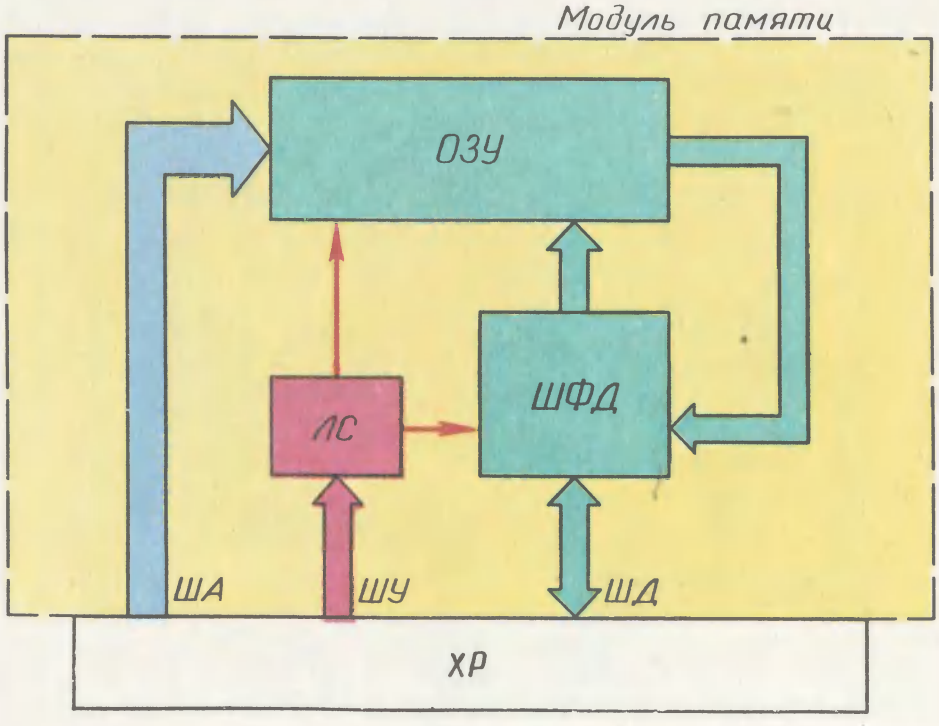
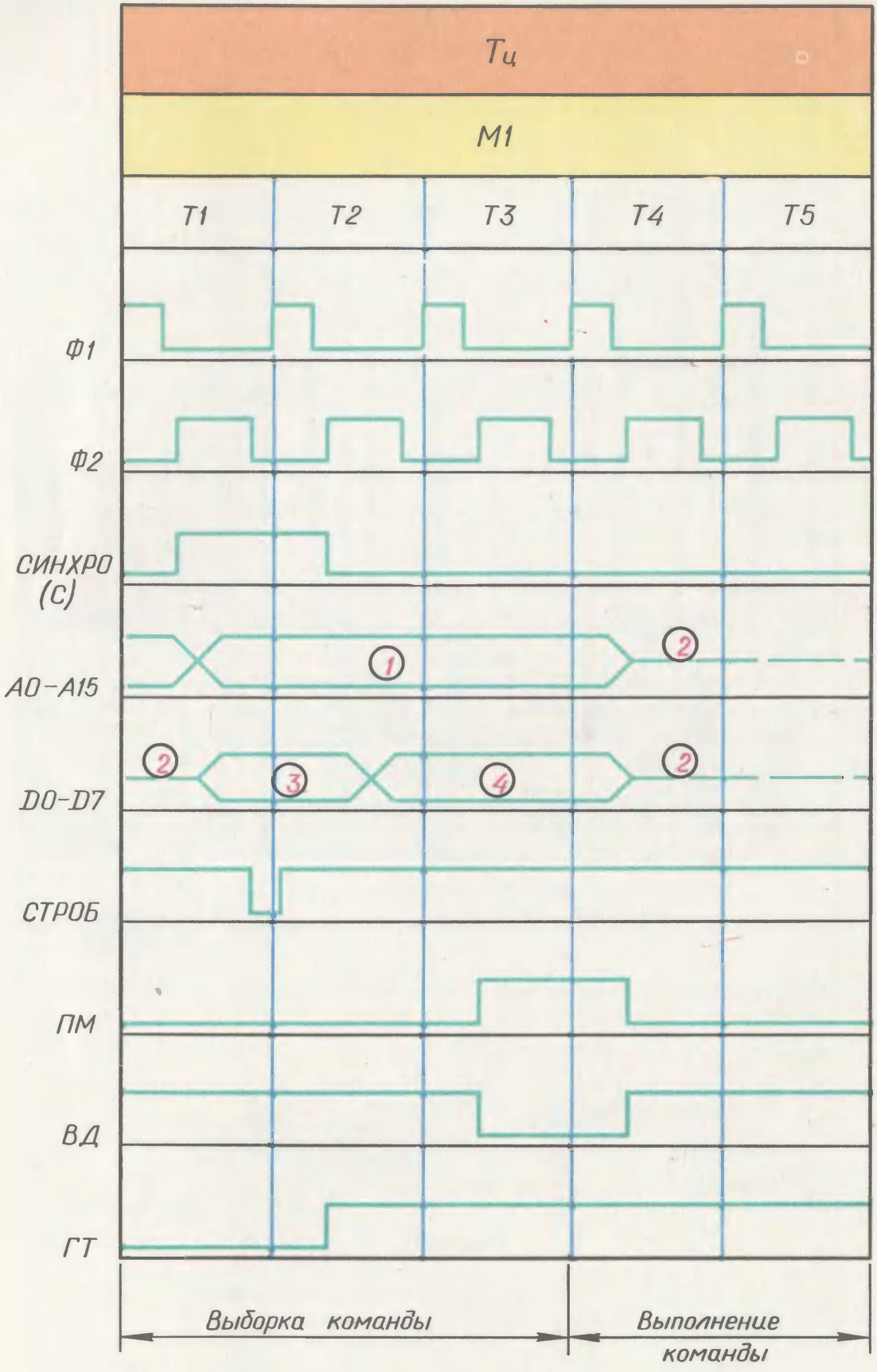
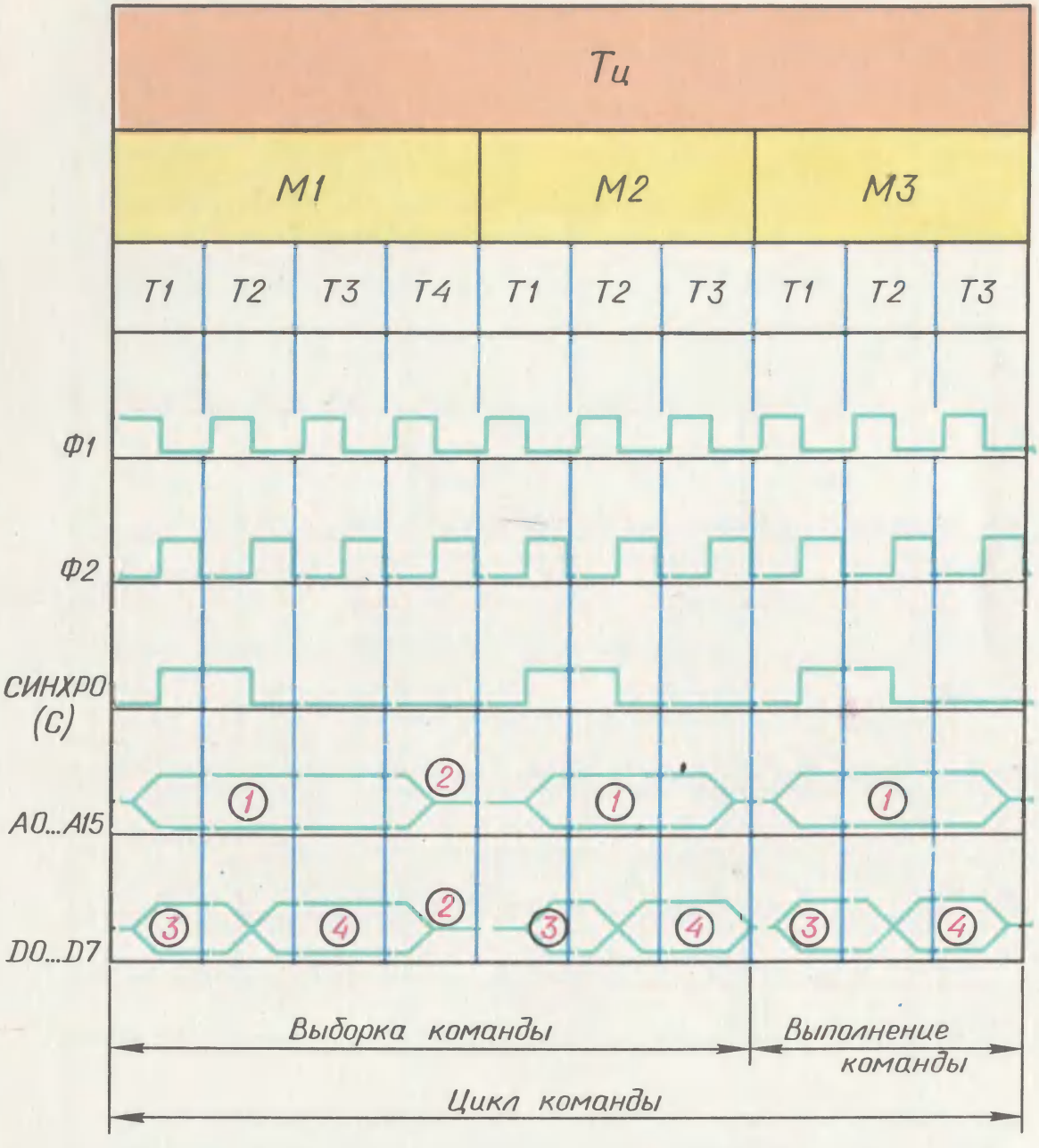

Модуль памяти
ОЗУ
ЛС
ШФД
ША
ШУ
ШД
ХР

Рис. 47



1. Адрес
2. Высокоимпеданское состояние
3. Сигналы о состоянии микропроцессора
4. Данные

Рис. 48



① Адрес
② Высокоимпедансное состояние
③ Сигналы о состоянии микропроцессора
④ Данные

Рис. 49

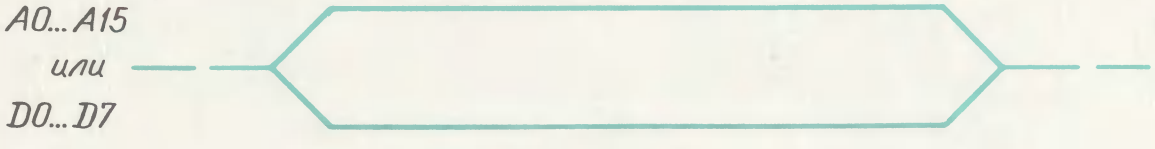


Рис. 50

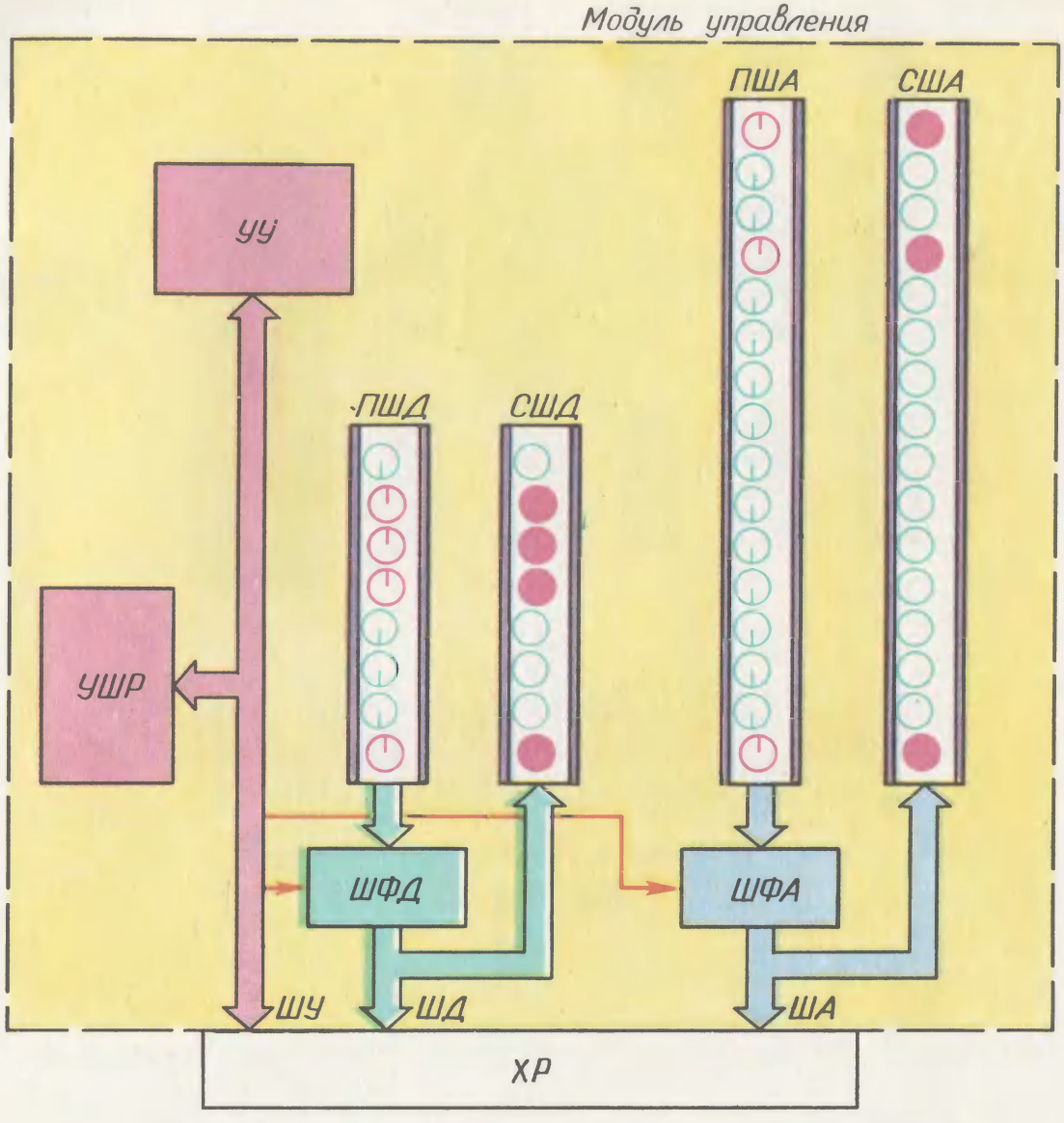


Рис. 51

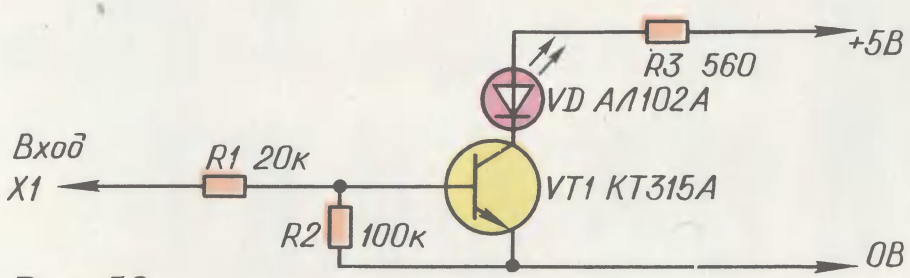


Рис. 52

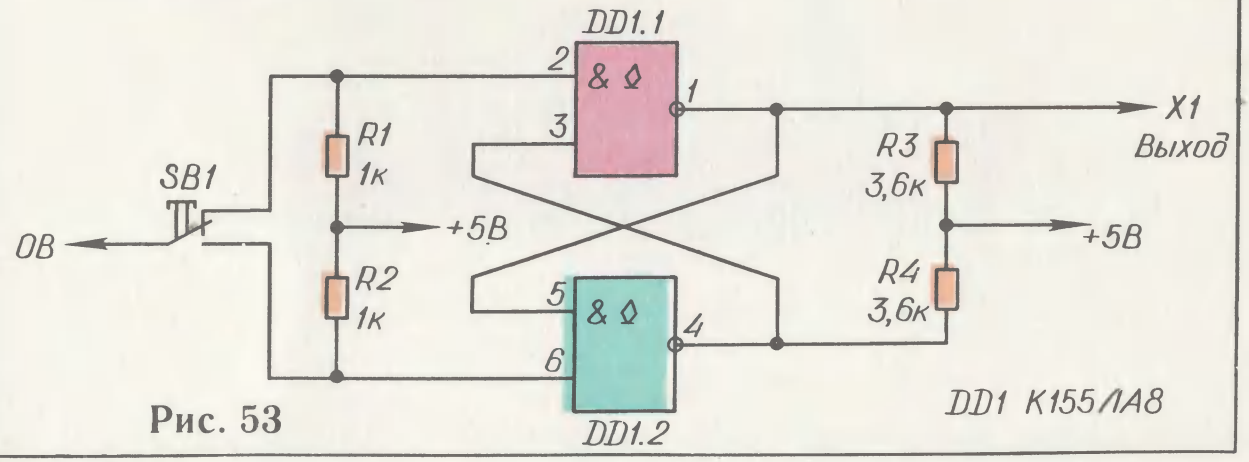


Рис. 53

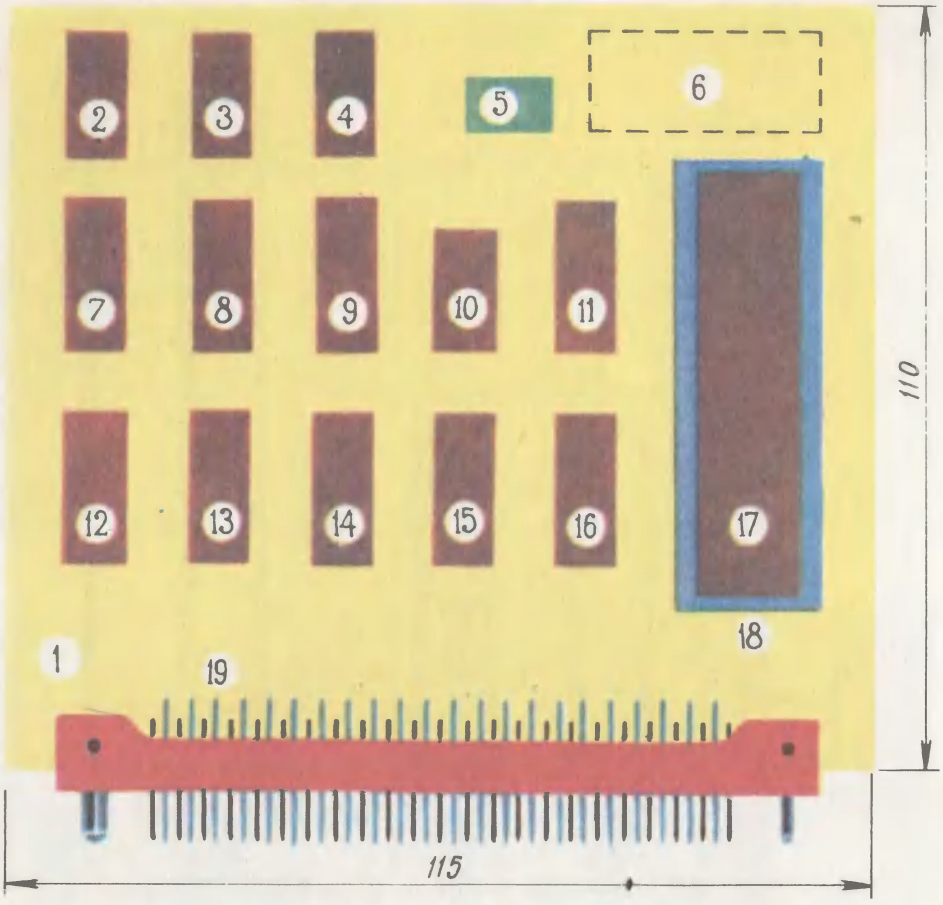

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
1
19
110
115

Рис. 54

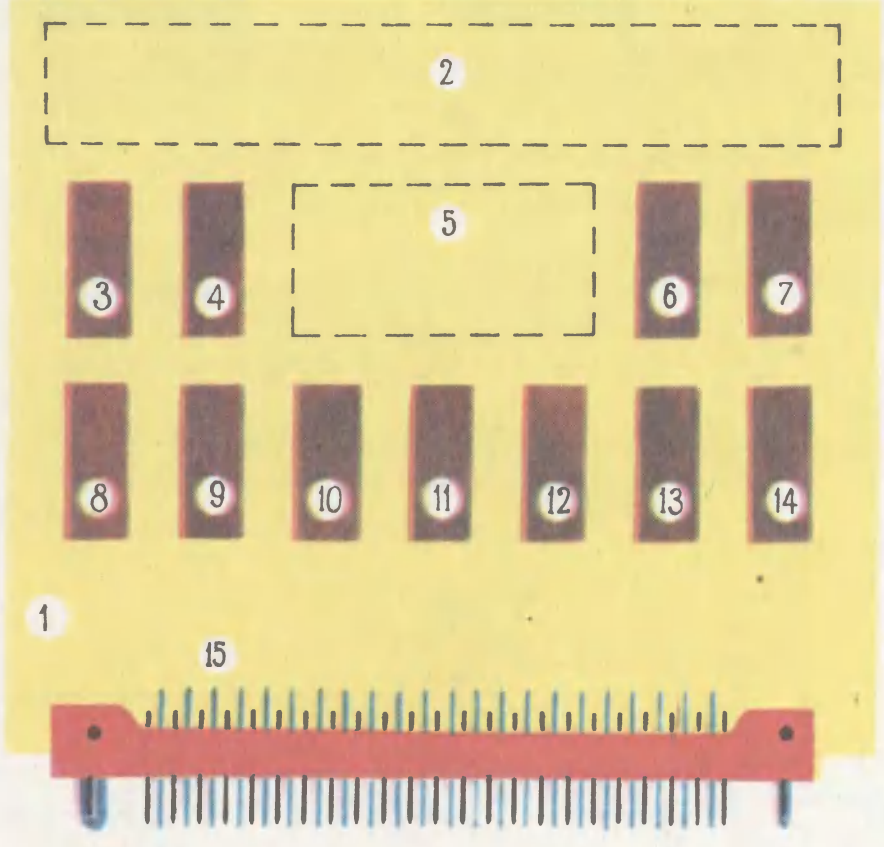

2
5
3
4
6
7
8
9
10
11
12
13
14
1
15

Рис. 55

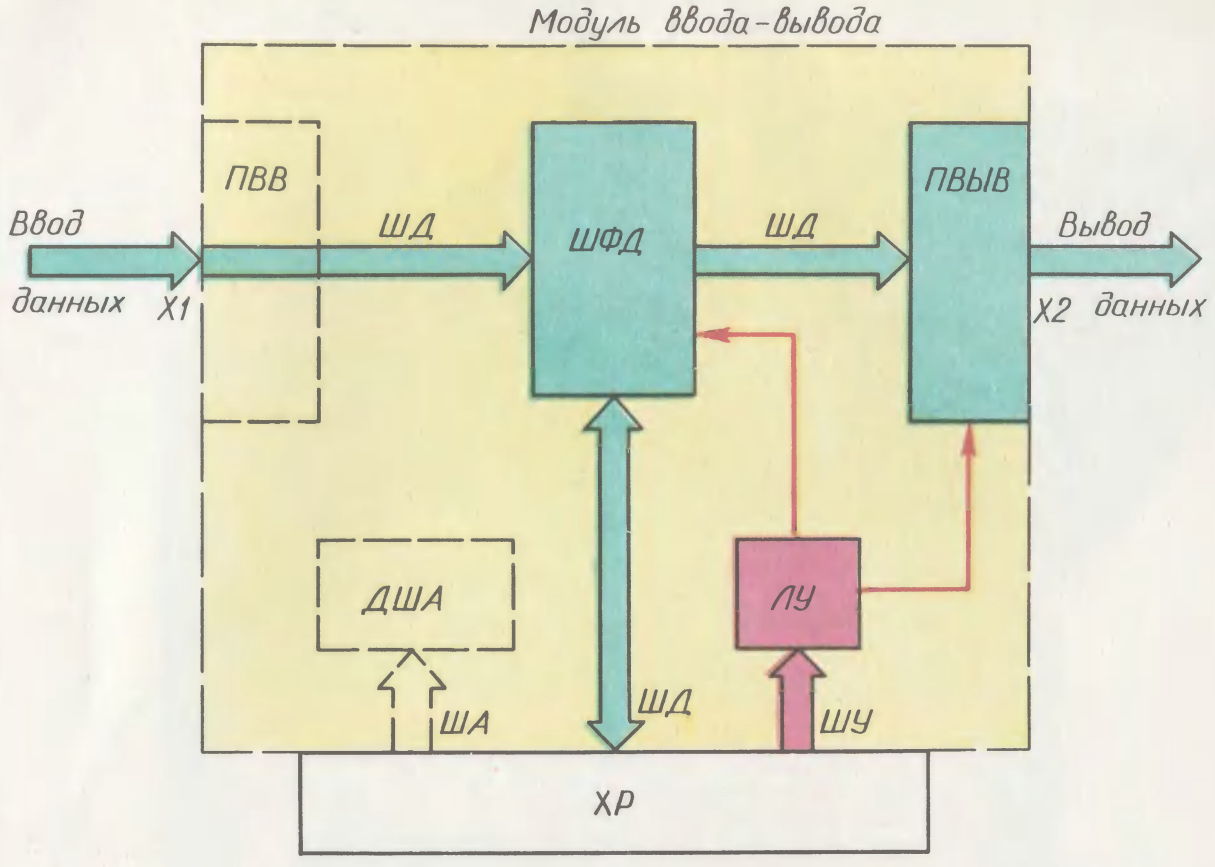

Модуль ввода-вывода
ПВВ
Ввод
данных
X1
ШД
ШФД
ШД
ПВЫВ
Вывод
X2
данных
ДША
ЛУ
ША
ШД
ШУ
ХР

Рис. 56

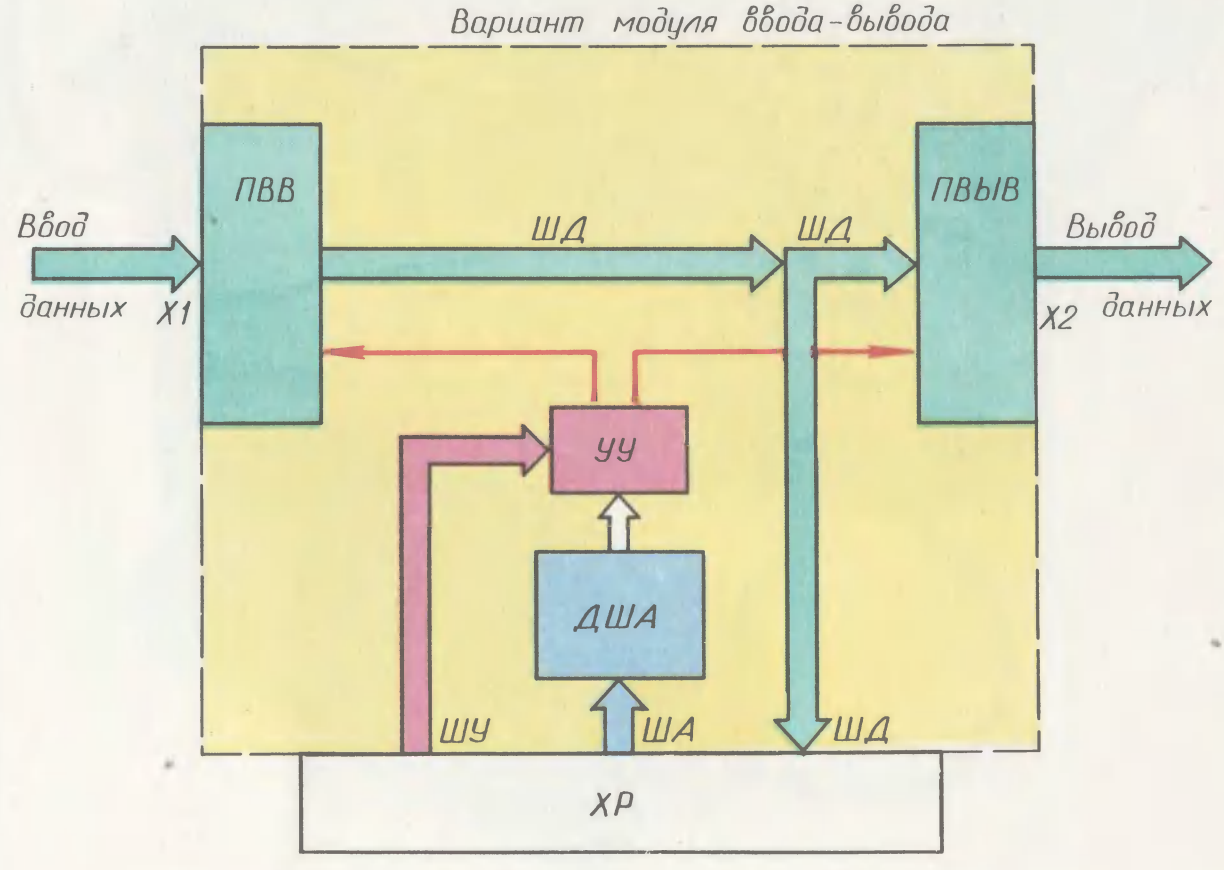

Вариант модуля ввода-вывода
ПВВ
Ввод
данных
X1
ШД
ШД
ПВЫВ
Вывод
X2
данных
УУ
ДША
ШУ
ША
ШД
ХР

Рис. 57

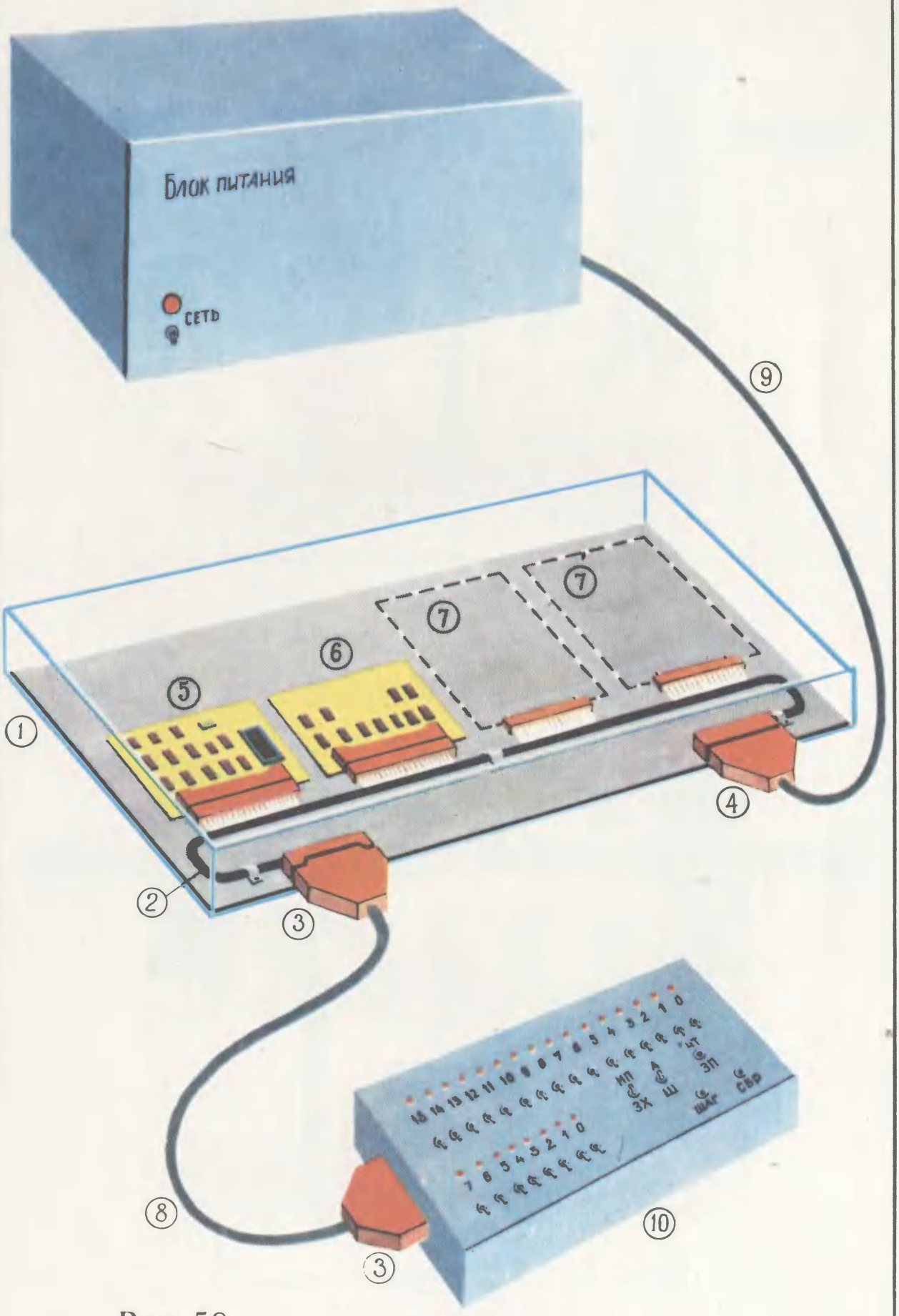


Рис. 58

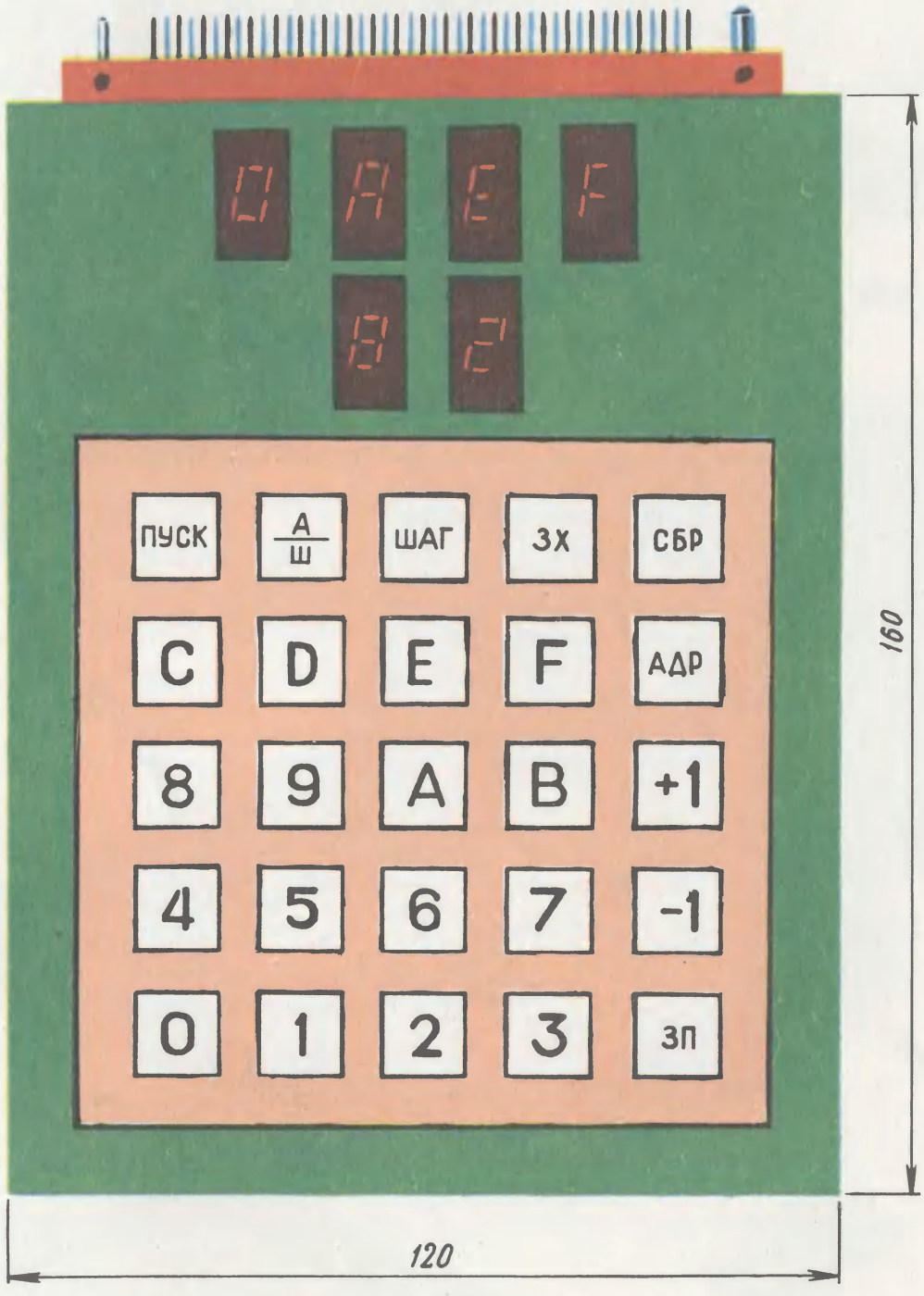


Рис. 59

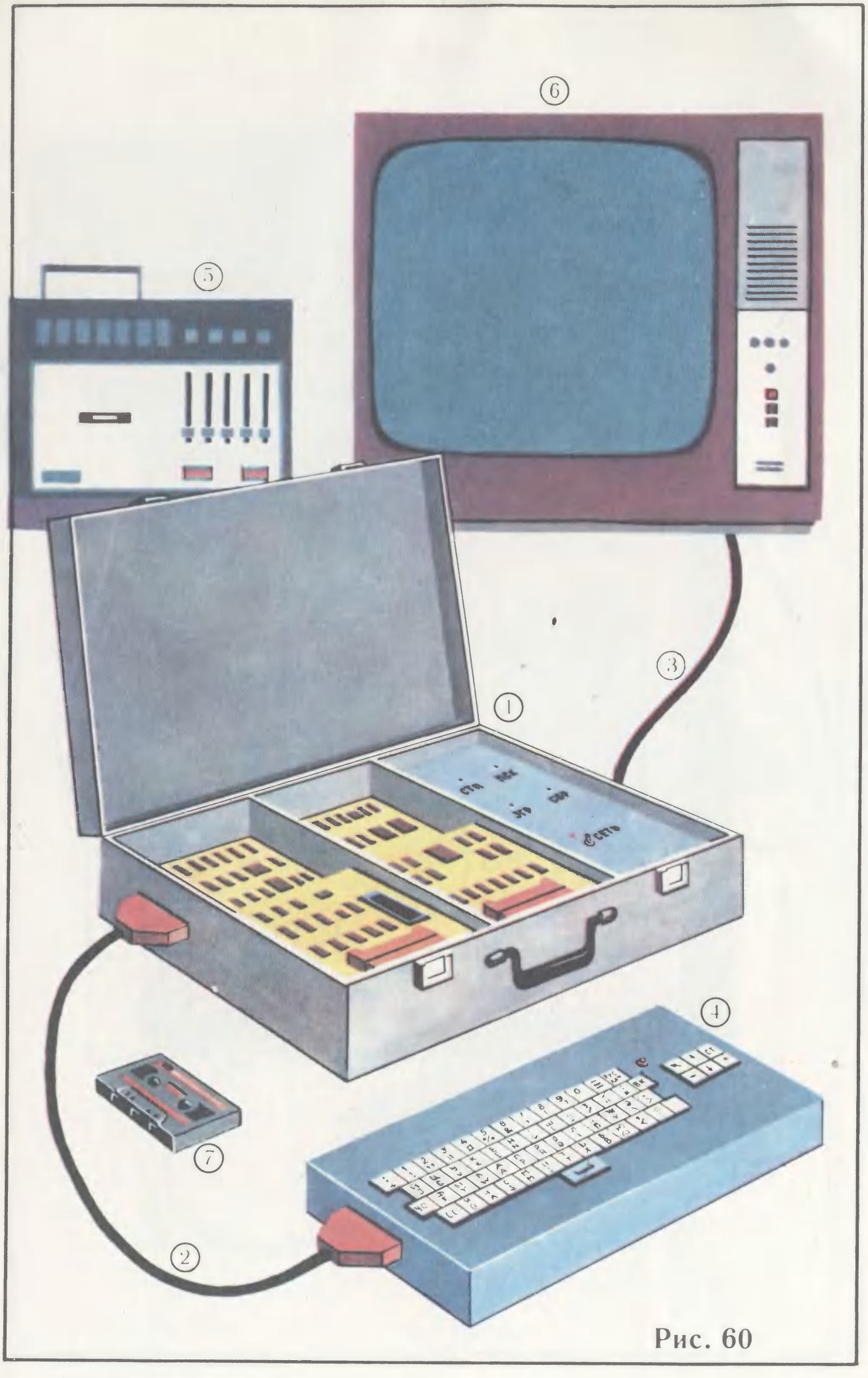


Рис. 60

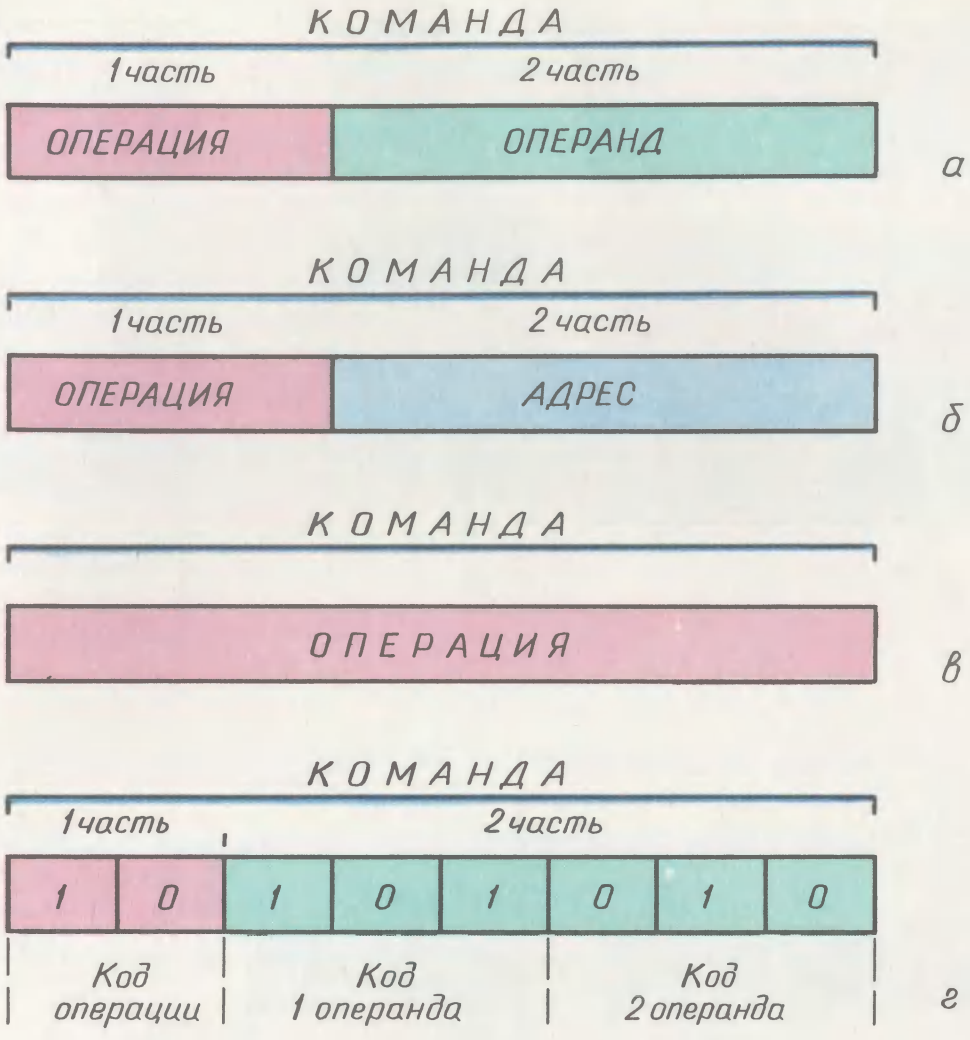

КОМАНДА
1 часть
2 часть
ОПЕРАЦИЯ
ОПЕРАНД
а
КОМАНДА
1 часть
2 часть
ОПЕРАЦИЯ
АДРЕС
б
КОМАНДА
ОПЕРАЦИЯ
в
КОМАНДА
1 часть
2 часть
1 0 1 0 1 0 1 0
Код операции
Код 1 операнда
Код 2 операнда
г

Рис. 61

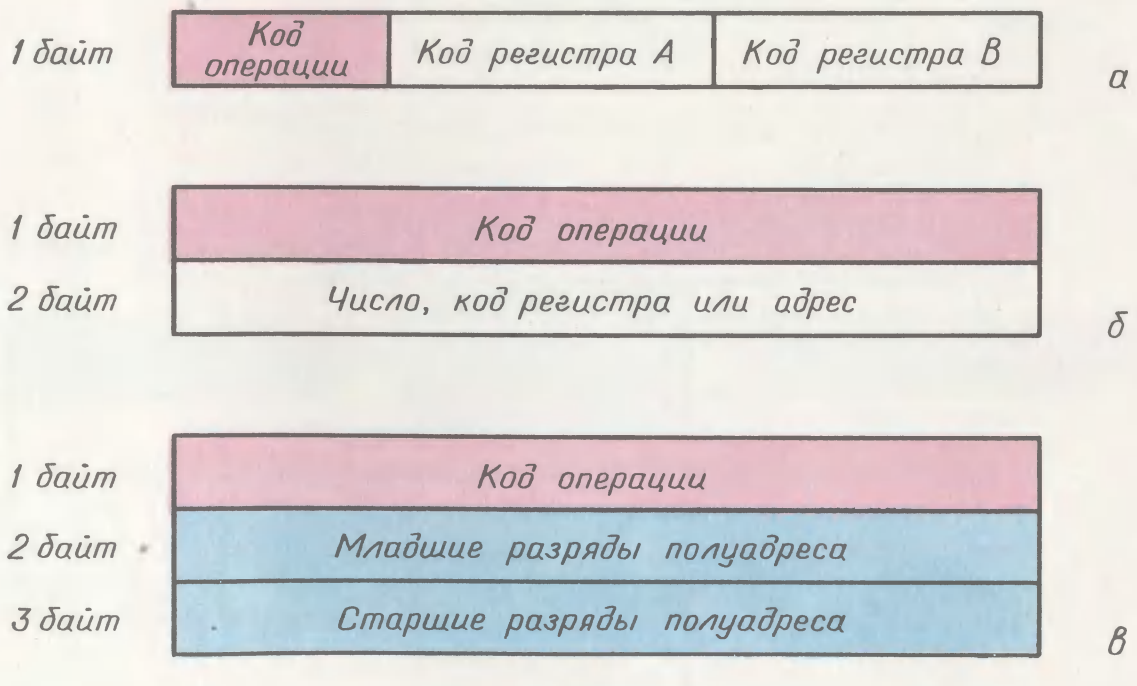
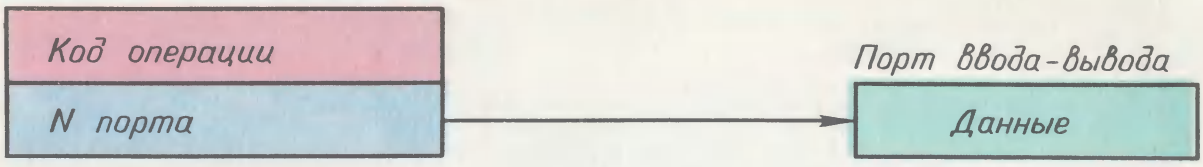
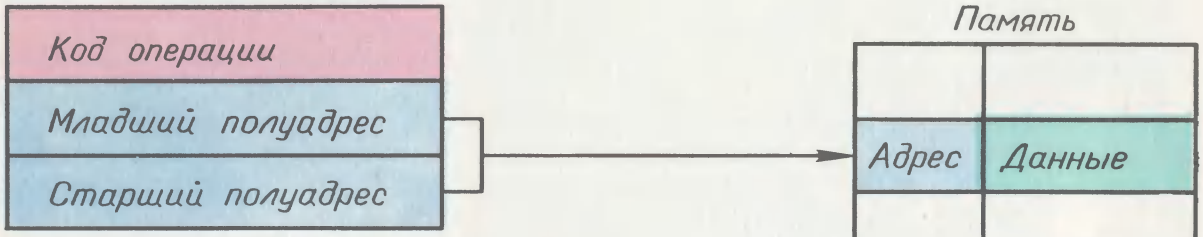

1 байт
Код операции
Код регистра А
Код регистра В
а
1 байт
2 байт
Код операции
Число, код регистра или адрес
б
1 байт
2 байт
3 байт
Код операции
Младшие разряды полуадреса
Старшие разряды полуадреса
в

Рис. 62



а



б

Рис. 63

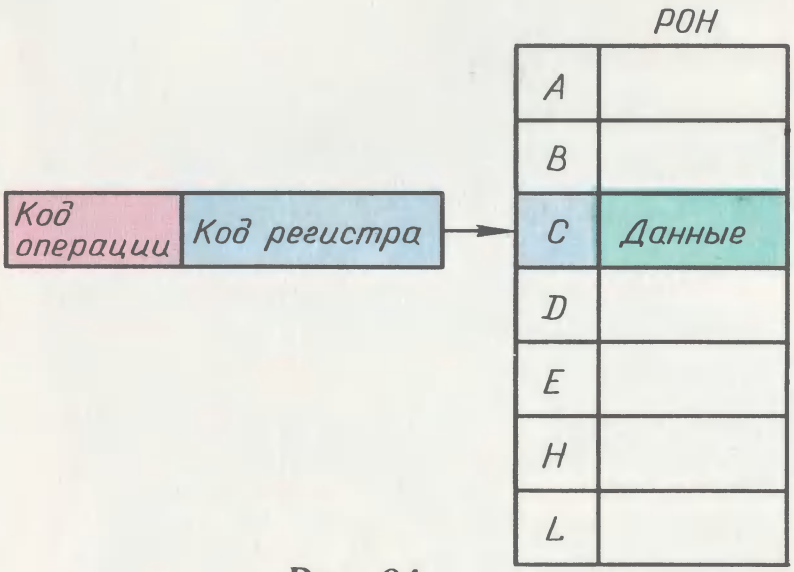


Рис. 64

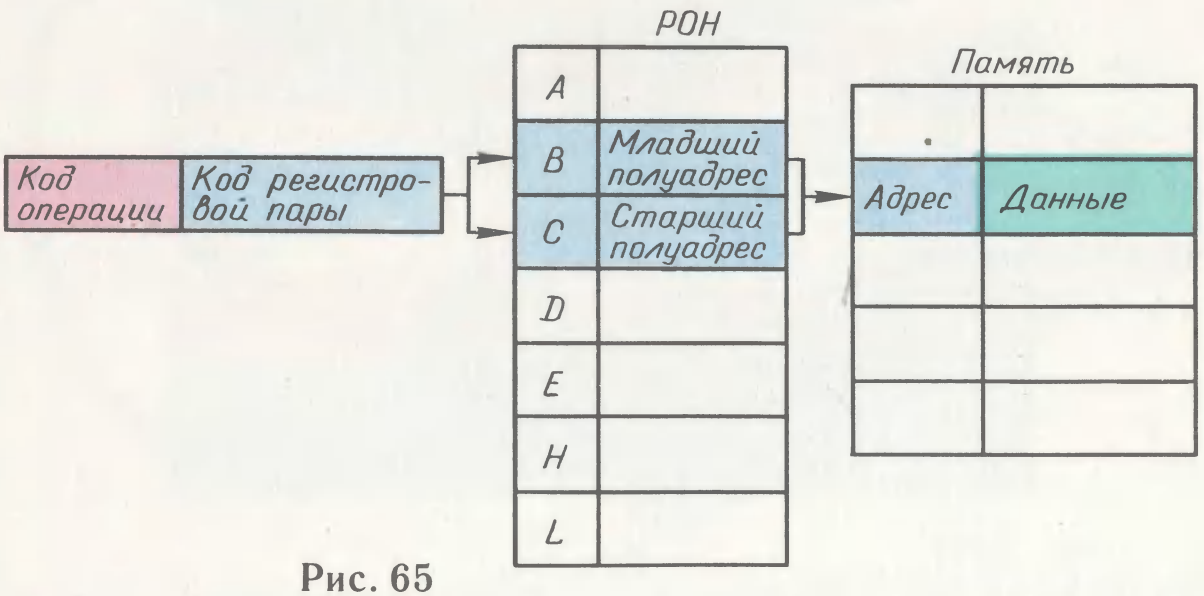


Рис. 65

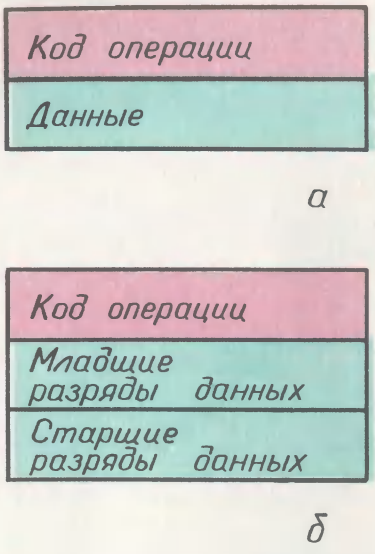


Рис. 66

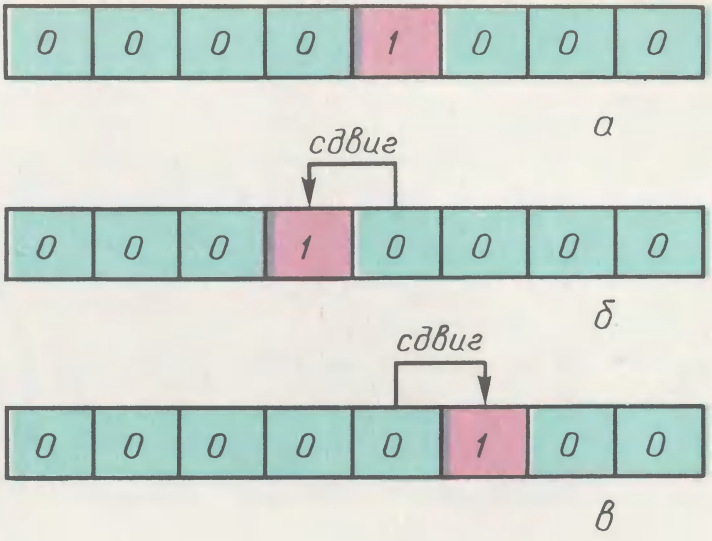


Рис. 67

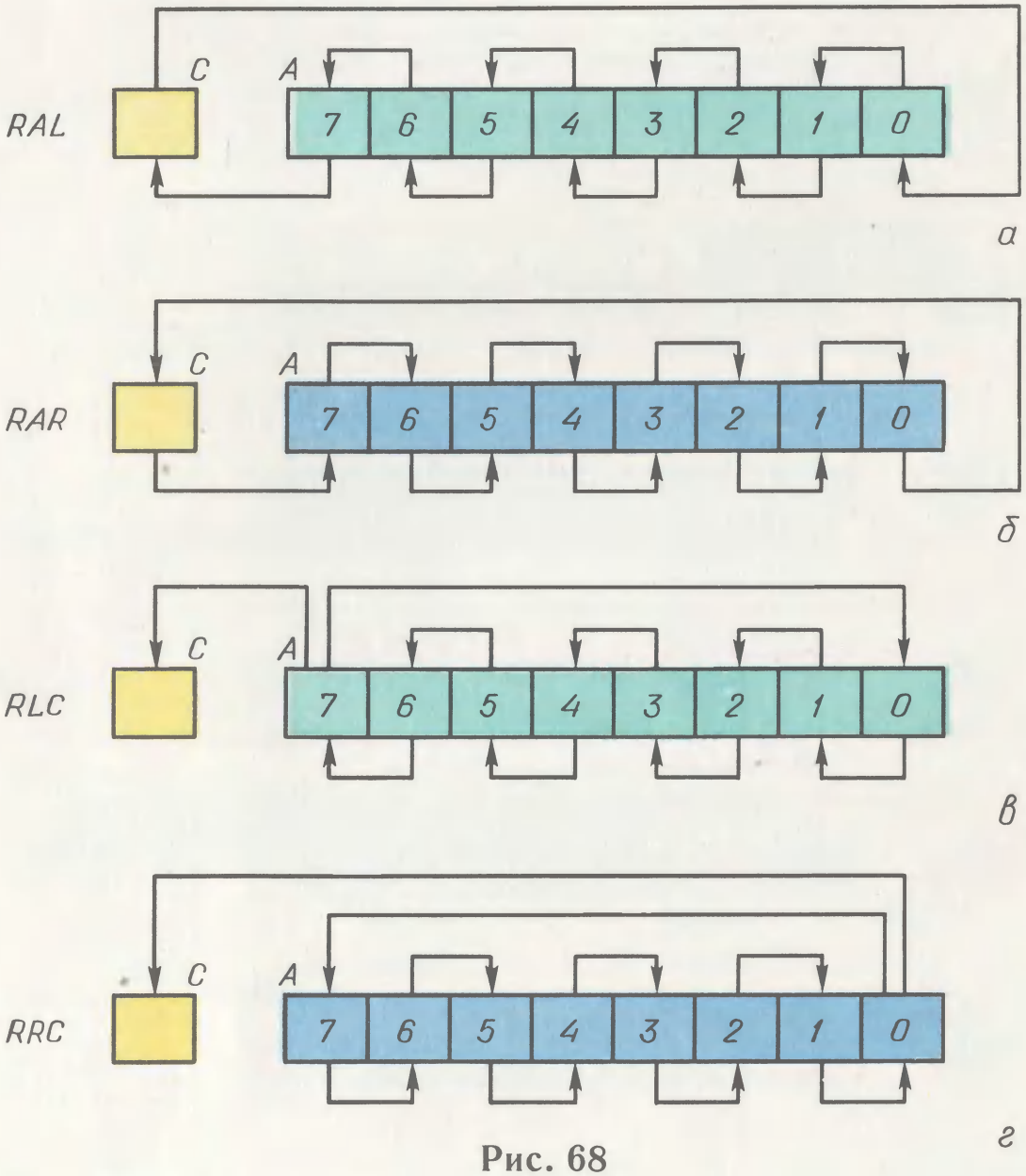


Рис. 68

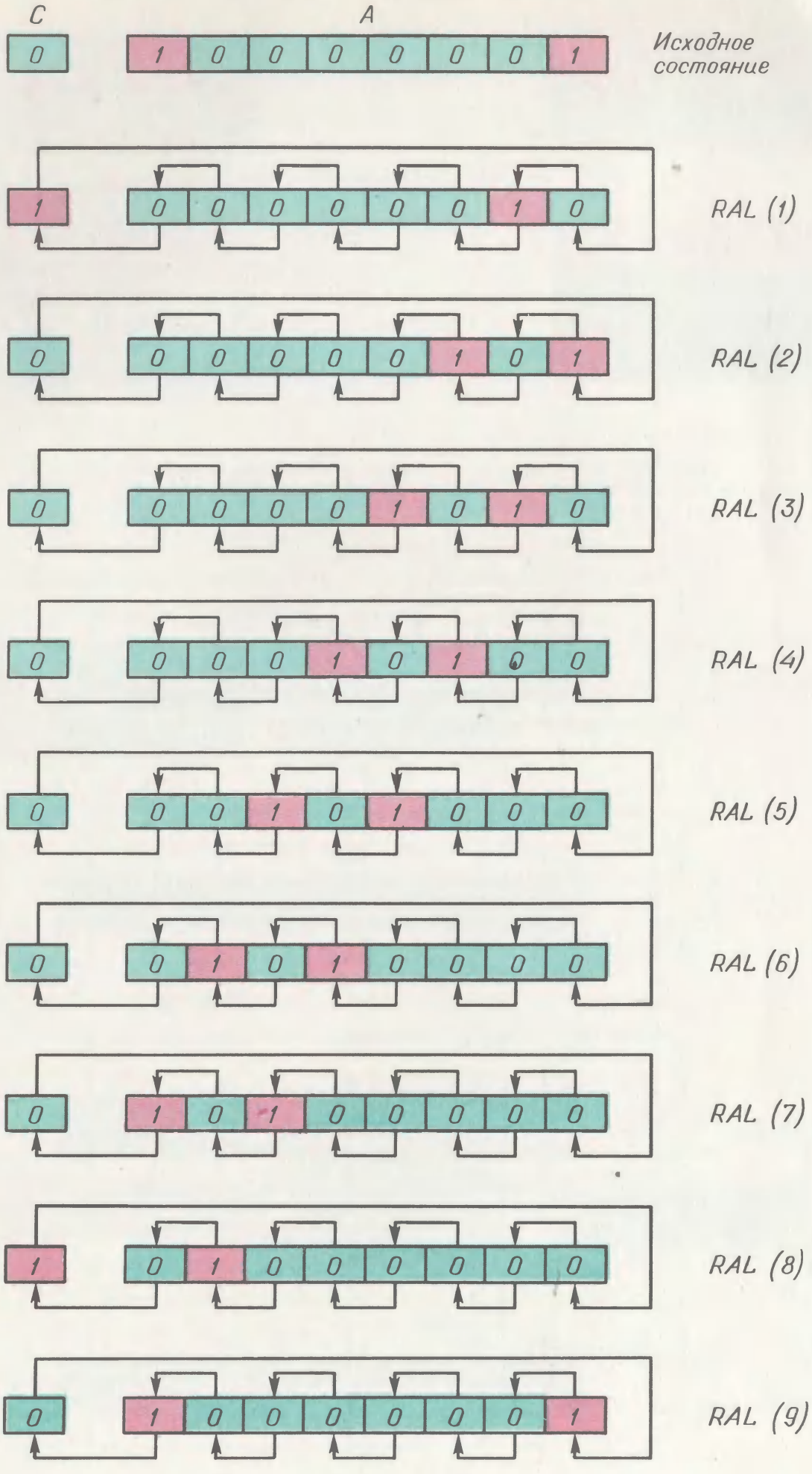

C
A
0
1 0 0 0 0 0 0 1
Исходное состояние
1
0 0 0 0 0 0 1 0
RAL (1)
0
0 0 0 0 0 1 0 1
RAL (2)
0
0 0 0 0 1 0 1 0
RAL (3)
0
0 0 0 1 0 1 0 0
RAL (4)
0
0 0 1 0 1 0 0 0
RAL (5)
0
0 1 0 1 0 0 0 0
RAL (6)
0
1 0 1 0 0 0 0 0
RAL (7)
1
0 1 0 0 0 0 0 0
RAL (8)
0
1 0 0 0 0 0 0 1
RAL (9)

Рис. 69

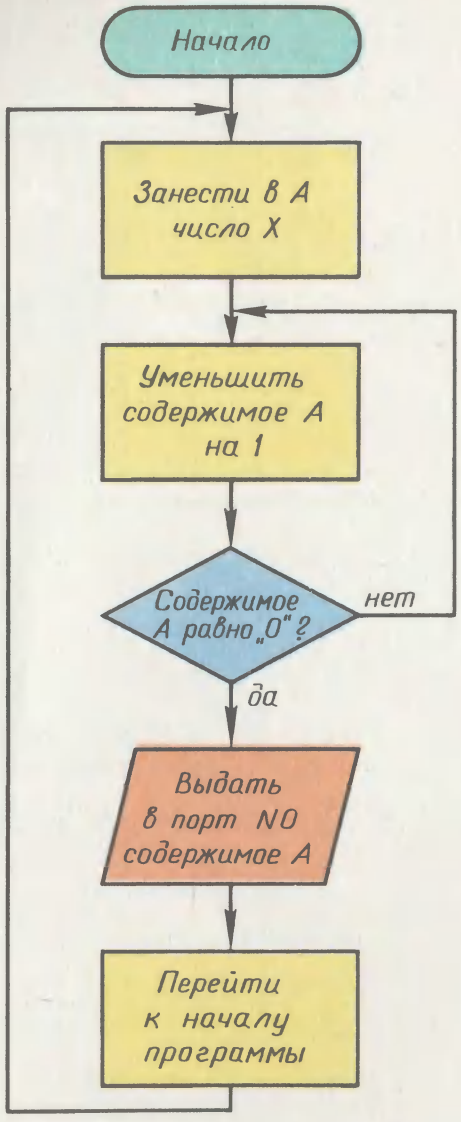


Рис. 70

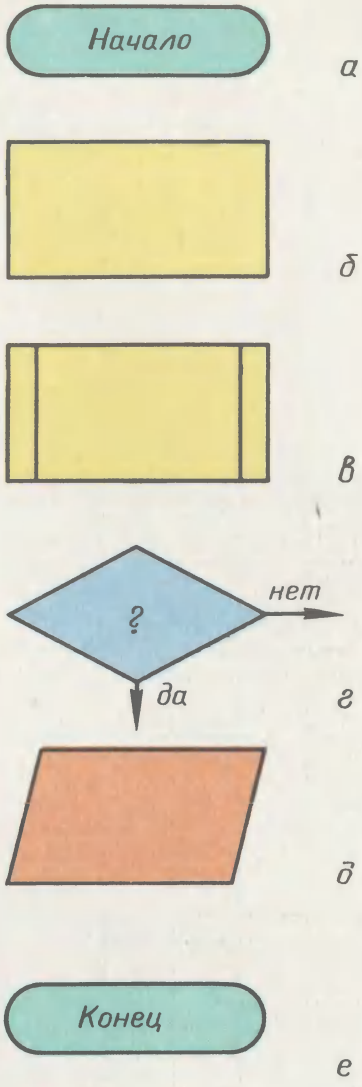


Рис. 71

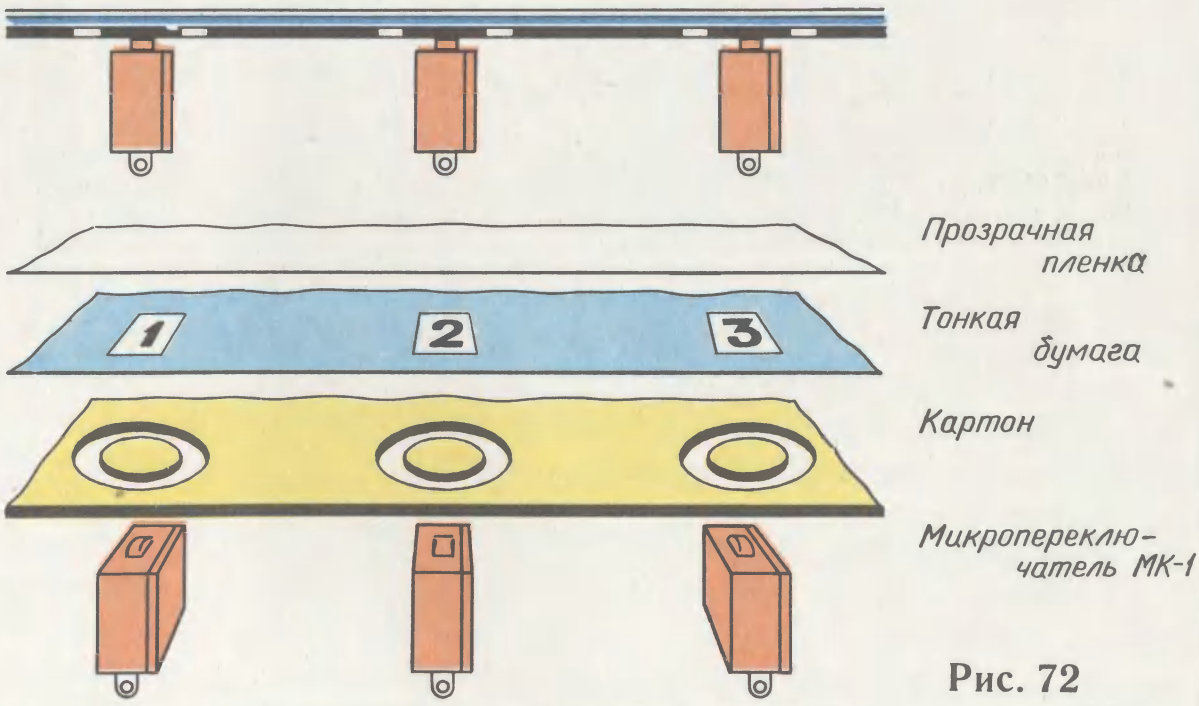


Рис. 72

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	LXI B,D16	STAX B	INX B	INR B	DCR B	MVI B,D8	RLC	–	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C,D8	RRC	0
1	–	LXI D,D16	STAX D	INX D	INR D	DCR D	MVI D,D8	RAL	–	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E,D8	RAR	1
2	–	LXI H,D16	SHLD ADR	INX H	INR H	DCR H	MVI H,D8	DAA	–	DAD H	LHLD ADR	DCX H	INR L	DCR L	MVI L,D8	CMA	2
3	–	LXI SP,D16	STA ADA	INX SP	INR M	DCR M	MVI M,D8	STC	–	DAD SP	LDA ADR	DCX SP	INR A	DCR A	MVI A,D8	CMC	3
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A	4
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A	5
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A	6
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A	7
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A	8
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A	9
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A	A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A	B
C	RNZ	POP B	JNZ ADR	JMP ADR	CNZ ADR	PUSH B	ADI D8	RST 0	RZ	RET	JZ ADR	–	CZ ADR	CALL ADR	ACI D8	RST 1	C
D	RNC	POP D	JNC ADR	OUT N	CNC ADR	PUSH D	SUI D8	RST 2	RC	–	JC ADR	IN N	CC ADR	–	SBI D8	RST 3	D
E	RPO	POP H	JPO ADR	XTHL	CPO ADR	PUSH H	ANI D8	RST 4	RPE	PCHL	JPE ADR	XCHG	CPE ADR	–	XRI D8	RST 5	E
F	RP	POP PSW	JP ADR	DI	CP ADR	PUSH PSW	ORI D8	RST 6	RM	SPHL	JM ADR	EI	CM ADR	–	CPI D8	RST 7	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

ОБОЗНАЧЕНИЯ

ADR – Двухбайтовый адрес
D8 – Однобайтовый операнд
D16 – Двухбайтовый операнд
N – Номер порта УВВ

ФОРМАТ КОМАНДЫ

Однобайтовая
Двухбайтовая
Трехбайтовая

ПРИМЕР:

Команда MOV B,C имеет код операции 41
Код операции D3 соответствует команде OUT N

Рис. 73

будет проходить по изображаемому проводнику и каково назначение того или иного вывода микросхемы. Вычертив схему, полезно составить спецификацию всех деталей, указанных на схеме, а по мере приобретения деталей производить предварительную проверку каждой детали. Проверку микросхем рекомендуется выполнять путем ее установки в специально смонтированную панельку, позволяющую видоизменять схему проверки и подключать измерительные приборы (пробник, вольтметр) к нужным выводам микросхемы. При проверке микросхем не следует забывать о влиянии электростатических зарядов, которые могут попасть на выводы микросхем.

Собирать модули микроЭВМ удобнее всего на унифицированных печатных платах, выполненных из двухстороннего фольгированного стеклотекстолита. На одной стороне платы располагаются площадки для установки микросхем, резисторов и конденсаторов, а с другой — контактные площадки, соединяемые отрезками монтажного провода. На обеих сторонах платы должны находиться и шины источника питания. Такие платы выпускаются [35] под торговым названием ПР (плата радиолюбителя). Часто в продаже бывают и другие типы аналогичных плат, предназначенных для сборки радиолюбительских конструкций, в том числе и с использованием интегральных микросхем. Планируется выпуск печатных плат для сборки радиолюбительского компьютера «Радио-86 РК». Такие платы будут иметь название ПР-8. Вместе с тем следует предостеречь читателей от разработки и изготовления печатных плат для модулей собираемой микроЭВМ, так как их не только трудно изготовить, но и сложно налаживать. Монтаж лучше выполнять навесным способом, при помощи тонких изолированных проводов. Разрабатывая конструкцию платы микропроцессорного модуля, необходимо помнить, что включать микропроцессор в схему модуля следует при помощи панельки. В целях сохранности микропроцессора предварительная проверка всех модулей должна выполняться без участия микропроцессора. Микропроцессор устанавливается в панельку только тогда, когда будут тщательно проверены рабочие напряжения и логические уровни сигналов на всех выводах его панельки, а также будет определено, что все остальные модули смонтированы правильно и работают нормально.

Панельку изготавливают, используя гнездовой многоконтактный разъем, имеющий шаг между гнездами, равный 2,5 мм. При этом нужно учесть, что панельку для микропроцессора, имеющего 48 выводов, изготовить очень трудно из-за малого шага, большой длины и недостаточной жесткости выводов. В этом случае можно поступить и так: на плате микропроцессорного модуля установить многоконтактный гнездовой разъем, имеющий не менее 40 гнезд (например, типа ГРПМ 1—45). Этот разъем будет служить своеобразной панелькой для будущего микропроцессора, который размещают на небольшой гетинаксовой пластинке рядом со штыревым разъемом, аналогичным по типу гнездовому разъему. Затем

осторожно, соблюдая все правила, предупреждающие перегрев выводов и влияние электростатических зарядов, подпаивают 40 рабочих выводов микропроцессора (см. табл. 7 Приложения) к 40 штыревым контактам разъема. Включение микропроцессора в схему модуля выполняется путем установки штыревого разъема в гнездовой.

Перед установкой деталей на плате полезно выполнить графическое моделирование. С этой целью вычерчивают сборочный чертеж платы и на чертеже карандашом изображают примерное расположение микросхем. Затем, согласно принципиальной схеме, цветными карандашами выполняют соединение всех выводов микросхем. Если при этом обнаружится, что какая-то группа соединительных проводников получается очень длинной, то расположение микросхем меняют, так как для устойчивой работы модулей и уменьшения воздействия помех соединения между выводами отдельных элементов схемы должны быть выполнены как можно короче. По этой же причине при проводном (навесном) способе монтажа не следует применять жгутование и параллельную укладку проводов.

На первый взгляд, предварительное вычерчивание вариантов расположения микросхем может показаться утомительным и ненужным. На самом же деле оказывается значительно проще и легче несколько раз перечертить схему, чем потом выпаивать неудачно установленную микросхему. Нужно помнить, что процесс выпаивания микросхемы очень трудоемкий и не всегда заканчивается удачно. Такие микросхемы после перепайки чаще всего приходят в негодность.

При монтаже важно выполнять соединения не только правильно, но и надежно. Пайка должна обеспечить хороший электрический контакт, прочность соединения, отсутствие последующей коррозии и невозможность короткого замыкания двух рядом расположенных паек. Для выполнения этих условий все места, подлежащие пайке, должны быть чистыми, а паяльник хорошо облужен и заточен.

Поскольку выводы микросхем расположены очень близко друг возле друга, то одна пайка от другой часто отстоит на 0,5—1 мм. Поэтому паять нужно миниатюрным паяльником, имеющим диаметр жала не более 2—3 мм. Прогревая паяльником место пайки, не следует забывать о возможности перегрева микросхем (обязательно пользуясь при пайке пинцетом).

Напомним, что номера выводов у всех микросхем отсчитываются от специальной метки, выполненной на корпусе. Этой меткой служит небольшое углубление (точка), сделанное на верхней лицевой поверхности корпуса, или же небольшая выемка с одного края корпуса (см. рис. 10, *а* вклейки).

Если микросхему расположить так, чтобы точка или выемка находились в верхней части корпуса, то в левом верхнем углу окажется первый вывод. Остальные выводы имеют нумерацию, идущую против часовой стрелки. Если смотреть на микросхему

с обратной стороны (со стороны выводов), то первый вывод находится в правом верхнем углу, а нумерация идет по часовой стрелке.

Как показывает опыт, эта особенность отсчета выводов является основной причиной ошибок при монтаже. Для правильной ориентации выводов микросхем и уменьшения вероятности ошибочного отсчета номера вывода рекомендуется на обратной стороне платы (на монтажной стороне) с помощью цветного лака или бумажного маркера обозначить положение первого вывода микросхемы и помнить, что отсчет остальных выводов при выполнении монтажа производится по часовой стрелке от этого первого вывода. Иногда радиолюбители отмечают красным и зеленым лаком шины питания +5 В и 0 В и соответствующие выводы питания микросхем.

Перед установкой микросхем на плату шины 0 В, +5 В, —5 В и +12 В между собой необходимо соединить перемычками и снять их после окончания монтажа платы. Монтаж микросхем, располагаемых на унифицированных платах, часто выполняют до их установки, а именно сперва соединяют проводниками контактные площадки с отверстиями под выводы микросхем, затем тщательно проверяют выполненный монтаж и после этого устанавливают навесные детали — микросхемы, резисторы, конденсаторы и др.

Один из возможных вариантов расположения основных деталей на монтажной плате микропроцессорного модуля приведен на рис. 54 вклейки. Цифры в белых кружках обозначают следующее:

1 — плата модуля;
2 — К155ЛА3 (DD14 на рис. 12);
3— К155ТМ2 (DD13);
4 — К155ЛЛ (DD4);
5 — кварц Z1;
6 — место для монтажа R и C;
7 — 9,12—16 — К589АП16 (DD5—DD12);
10 — К155ЛН1 (DD3);
11 — К580ГФ24 (DD1);
17 — микропроцессор КР580ИК80А (DD2);
18 — панелька для КР580ИК80А;
19— разъем ГРПМШ — 1—45 (XP1).

Вариант размещения деталей на плате модуля памяти изображен на рис. 55 вклейки. Здесь цифры обозначают следующее:

1 — плата модуля;
2 — место для дополнительных деталей;
3 — 8 — 14 — К565РУ2 (DD1—DD8 на рис. 14);
4 — К155ЛА3 (DD11);
5 — место для монтажа R и C;
6,7 — К589АП16 (DD9; DD10);
15 — разъем ГРПМШ — 1—45 (XP1).

В зависимости от типа печатных плат и комплекта деталей, конструктивное размещение элементов схемы может быть иным.

Важно только учесть, что при работе микросхемы ощутимо нагреваются (особенно микропроцессор и микросхемы памяти), поэтому платы и их размещение в корпусе нужно конструировать так, чтобы обеспечить достаточный приток воздуха для отвода тепла.

Налаживание модулей микроЭВМ

Схема сложного электронного устройства будет работать абсолютно правильно, если очень тщательно выполнены все требования монтажа. Но часто бывает так, что при первом включении схема не работает или работает не так, как нужно. Поэтому перед включением необходимо проверить, нет ли ошибок в монтаже модулей и общей шины микроЭВМ, ибо при выполнении нескольких сотен паек, как правило, забывают что-либо соединить или соединяют не те выводы, которые нужны. Пропущенное или неправильное соединение может полностью вывести из строя всю схему модуля или привести в негодность его основные детали. Особенно опасен выход из строя микропроцессора. Поэтому все предварительные проверки и налаживание нужно производить без микропроцессора. Его необходимо вынуть из панельки микропроцессорного модуля, завернуть в фольгу и хранить до тех пор, пока не будут выполнены все операции по предварительному испытанию модулей.

Несмотря на то, что монтаж модулей сопровождался отметкой соединений цветным карандашом на принципиальной схеме, следует еще раз проверить правильность и надежность соединений, чтобы убедиться в том, что каждый проводник схемы подсоединен и находится на своем месте. Особенно внимательно нужно осмотреть пайки выводов, чтобы определить нет ли коротких замыканий близко расположенных выводов микросхем.

Необходимо также проверить полярность включения электролитических конденсаторов, диодов, светодиодов, а также правильность выполненной маркировки и подключения первого вывода и выводов питания каждой микросхемы. Проверку монтажных соединений можно производить при помощи омметра, но лучше собрать простейший пробник, состоящий из последовательно соединенных лампочки на 2,5 В и одного элемента 373.

Начинают с проверки монтажа общей шины, подключив один конец пробника в гнездо А1 первого разъема (XS1 на рис. 11), а второй конец поочередно подключают к гнездам А1 всех других разъемов (кроме разъема XS5 блока питания) и по степени свечения лампочки судят о правильности и надежности выполненных соединений. Аналогично проверяют соединения гнезд Б1, А1, Б2 и т. д., пока не проверят соединения всех гнезд, задействованных на разъемах общей шины.

После этого приступают к проверке блока питания. Включив его в сеть, проверяют вольтметром все три выдаваемых напряжения (+5 В, —5 В, +12 В), затем нагружают выход каждого источника напряжения путем подсоединения низкоомного (5—

20 Ом) проволочного переменного резистора между выводом 0 В и выводами +5 В, —5 В и +12 В, наблюдая при этом, как работает стабилизатор, не выходит ли за допустимые пределы (±5 %), напряжение при изменении нагрузки. Величину сопротивления резистора устанавливают такой, чтобы ток, протекающий через него, был приблизительно равен току, указанному в табл. 3.2.

После проверки блок питания подключают к общей шине (в разъем XS5) и вольтметром проверяют наличие и величину питающих напряжений на гнездах питания (А1, Б1, А22, Б23 и др.) всех разъемов общей шины.

Если не отмечается отклонение от нормы, то в разъем XS3 включают модуль управления. При этом в цепь питания +5 В желательно включить амперметр и по его показаниям проверить, нет ли в схеме модуля причин для завышенного потребления тока. Если показания амперметра в 1,5 и более раз превышают данные табл. 3.2, то модуль управления немедленно выключают и выясняют причину завышения тока. Чаще всего это происходит из-за коротких замыканий в схеме, неправильной полярности включения микросхем или неисправных элементов (например, пробоя в конденсаторе или в микросхеме). Если потребление тока приближается к норме, то амперметр выключают и приступают к проверке работы модуля.

При включении модуля управления независимо от положения всех его переключателей и кнопок должны засветиться все светодиоды. Если какой-либо светодиод не засветится, то нужно проверить его исправность. Проверку выполняют, подключая параллельно светодиоду вольтметр. Если вольтметр показывает напряжение, превышающее 2 В, значит неисправен светодиод или он неправильно подключен (анод ошибочно принят за катод). Если на светодиоде напряжения не будет, нужно проверить логический элемент, к которому подключен светодиод, подавая на его вход уровни логического нуля и единицы и наблюдая, какой уровень имеется при этом на выходе. Проверку логического уровня можно производить вольтметром, однако для проверки логических элементов лучше смонтировать логический пробник и источник логических напряжений. Простейший логический пробник, изображенный на рис. 52 вклейки, состоит из транзистора VT1, светодиода VD и трех резисторов, R1, R2 и R3. Питается пробник от напряжения +5 В. Включают его в любой разъем общей шины. Если на вход X1 пробника подать уровень логического нуля, то транзистор VT1 будет заперт. Ток через светодиод VD не протекает и он не светится. При подключении к X1 уровня логической единицы транзистор открывается, что приводит к свечению светодиода.

В качестве источника низкого логического уровня (0) можно использовать провод, подключенный к выводу источника питания 0 В, а в качестве источника высокого уровня (1) — провод, подключенный через резистор R = 1 КОм к выводу источника питания +5 В. Еще лучше выполнить источник логических уровней по схеме рис. 53 вклейки. Такой источник, имея в своем составе

RS-триггер, не имеет дребезга контактов. В исходном состоянии кнопки SB1 (кнопка отжата) на выходе X1 имеется уровень логической единицы. При нажатии кнопки SB1 RS-триггер, собранный на элементах DD1.1 и DD1.2, опрокидывается и на X1 появляется уровень логического нуля.

Если при испытании модуля управления будут светиться все его светодиоды, то приступают к проверке работы шинных формирователей DD4—DD5 и DD8—DD11 (см. рис. 15). Для этого переключатель SA10 устанавливают в верхнее положение («Захват»), а на разъеме XS1 микропроцессорного модуля гнездо Б6 (ЗХ) соединяют с гнездом Б5 (ПЗХ), имитируя этим работу отсутствующего в схеме микропроцессора, который в ответ на сигнал ЗХ выдал бы сигнал ПЗХ. Переключатель SA10 переводят в нижнее положение, благодаря чему уровень логического нуля поступает в микропроцессорный модуль (на контакт Б6 разъема XS1) и возвращается (с контакта Б5) в модуль управления, создавая на выводах $\overline{ВК}$ всех шести шинных формирователей (DD4—DD5 и DD8—DD11) уровни логического нуля.

Под воздействием нулевого уровня на выводах $\overline{ВК}$ шинные формирователи выходят из высокоимпедансного состояния и переключатели SA11—SA26 оказываются подключенными к адресной шине. Проверяя работу этих переключателей, устанавливают их ручки в такое положение, при котором переключатели соединяются с верхними (на схеме рис. 15) контактами. При этом светодиоды VD9—VD24 должны погаснуть. При переводе переключателей в другое положение светодиоды должны загореться. Проверяют действие каждого переключателя, переводя его из одного положения в другое и наблюдая при этом за гашением или свечением светодиода, подключенного к тому проводу адресной шины, что и переключатель.

Во время проверки работы адресных переключателей SA11—SA26 и адресных светодиодов VD9—VD24 все светодиоды VD1—VD8, указывающие состояние шины данных должны светиться вне зависимости от положения переключателей SA1—SA8, так как на выводы ВР шинных формирователей DD4 и DD5 подается уровень логической единицы с вывода 3 микросхемы DD1.1, а это приводит к тому, что выводы B1—B4 соединены с выводами C1—C4, а не с выводами A1—A4, и переключатели SA1—SA8 по-прежнему остаются отключенными от шины данных. Если же нажать кнопку SB2, то в этом нажатом положении ее переключатель соединится с нижним контактом («Запись»), и на входе 2 микросхемы DD1.1 возникнет уровень логической единицы. Он инвертируется микросхемой DD1.1 и с вывода 3 логический нуль поступает на выводы ВР шинных формирователей DD4 DD5. При этом их выводы A1—A4 соединяются с выводами B1—B4 и это приводит к подключению переключателей SA1—SA9 к шине данных. Теперь светодиоды VD1—VD8 должны показывать положение этих переключателей. При нажатой кнопке SB2 проверяют работу переключателей SA1—SA8 в том же порядке, как это делалось при

проверке переключателей SA11—SA26. Одновременно с помощью вольтметра (или пробника, собранного по схеме рис. 52) следует проверить логические уровни на всех гнездах адресной шины и шины данных разъема XS1 модуля микропроцессора и разъема XS2 модуля памяти. Эти уровни должны соответствовать положению переключателей SA1—SA8 и SA11—SA26, а также свечению соответствующих светодиодов.

Затем проверяют работу триггера DD1.1—DD1.2 путем многократного нажатия кнопки SB2. Так, при нажатии кнопки SB2 светодиоды VD1—VD8 должны показывать положение переключателей SA1—SA8, а при отжатии — их свечение не должно зависеть от положения переключателей (они все должны светиться).

В это же время нужно проверить выдачу сигналов $\overline{\text{ЗПЗУ}}$ и $\overline{\text{ЧТЗУ}}$ путем подключения пробника к гнездам А8 и Б9 разъема XS2. При отжатой кнопке SB2 уровень логического нуля должен быть на выводе Б9 ($\overline{\text{ЧТЗУ}}$), а при нажатой — на выводе А8 ($\overline{\text{ЗПЗУ}}$).

Модуль памяти проверяют, включая его в разъем XS2 общей шины. Установка модуля в разъем должна выполняться при выключенном напряжении блока питания. При включении питания желательно с помощью амперметра проверить ток, потребляемый модулем памяти. Если он не выходит за допустимые пределы, то приступают к проверке работоспособности модуля. С помощью переключателей SA11—SA26 устанавливают на адресной шине начальный, нулевой адрес. При этом светодиоды VD9—VD24 должны погаснуть.

Переключателями SA1—SA8 набирают двоичное число 01010101 и нажимают, а затем отпускают кнопку SB2. Набранное число должно поступить в ячейку памяти, запомниться там и отобразиться с помощью светодиодов VD1—VD8. Затем адресными переключателями (SA11—SA26) набирают адрес соседней ячейки памяти (№ 1), а переключателями данных (SA1—SA8) набирают число 10101010, и снова кратковременно нажимают кнопку SB2. Теперь светодиоды VD1—VD8 должны отобразить это число. Изменяя адрес и данные, проверяют работу не менее 20—30 начальных ячеек памяти. При этом нужно помнить, что если в запоминающем устройстве будет неисправна хотя бы одна ячейка памяти (особенно в начальной области), то работать с такой памятью нельзя. Результат работы будет непредсказуемым. Неисправную микросхему памяти придется заменить. Для проверки всех 1024 ячеек памяти имеются специальные программы, а для начальной проверки необходимо вручную проверить несколько десятков ячеек памяти.

Проверив модуль памяти, переходят к самому ответственному этапу налаживания — проверке модуля микропроцессора. Для этого модуль вставляют в разъем XS1 общей шины, предварительно вынув микропроцессор из панельки, установленной на плате модуля. Модуль вставляют при выключенном напряжении источников питания.

Проверку и налаживание модуля микропроцессора производят в такой последовательности:

1) снимают провод, соединявший гнездо Б6 разъема XS1 с гнездом Б5;

2) при помощи амперметра измеряют расход тока модулем;

3) вольтметром проверяют напряжение на выводах панельки микропроцессора: 20 (+5 В), 28 (+12 В) и 11 (—5 В);

4) пробником проверяют логический уровень на выводе 13 (ЗХ) панельки при различных положениях переключателя SA10 («Захват»). Оставляют переключатель SA10 в таком положении, когда на выводе 13 имеется уровень логического нуля, и вывод 13 панельки соединяют с выводом 21 (ПЗХ);

5) пробником проверяют логический уровень на выводе 23 (ГТ) панельки путем подачи уровня логического нуля (с помощью проводника, соединенного с ОВ) на контакт А5 разъема XS3 модуля управления. При этом на выводе 23 должен появляться уровень логической единицы;

6) пробником проверяют логический уровень на выводе 12 (СБР) панельки. При нажатии кнопки SB3 («Сброс») на этом выводе должен возникать уровень логической единицы;

7) проверяют прохождение адресных сигналов, поступающих с панельки микропроцессора на общую шину. Для этого переключатель SA10 устанавливают в верхнее положение. При этом должны засветиться все светодиоды. Проводником, соединенным с выводом питания ОВ, поочередно касаются адресных выводов панельки. При каждом касании должен гаснуть светодиод соответствующего разряда адресной шины;

8) аналогично проверяют прохождение сигналов данных, поступающих с панельки микропроцессора на общую шину. Для этого на вывод 17 (ПМ) панельки подают низкий логический уровень, а затем выводов данных поочередно касаются проводником, соединенным с ОВ, наблюдая за гашением светодиода проверяемого разряда шины данных;

9) проверяют наличие тактовых сигналов на выводах 22 (Ф1) и 15 (Ф2) панельки. Для этого желательно воспользоваться осциллографом, который определит не только наличие, а и параметры (амплитуду, форму, длительность) тактовых сигналов. Если осциллографа нет, то тактовые сигналы можно проверить при помощи транзисторного приемника, подсоединяя выводы Ф1 и Ф2 через конденсатор 100—200 пФ к антенне приемника. При этом в приемнике будет прослушиваться характерный звук высокого тона. При выключении питания модуля этот звук должен исчезать. Просмотреть сигналы Ф1 и Ф2 удается также с помощью телевизора, если подать эти сигналы через конденсатор емкостью 10 мкФ на вход видеоусилителя. На экране появятся темные вертикальные полосы. По количеству этих полос, их ширине и равномерности засветки можно приблизительно судить о параметрах сигналов Ф1 и Ф2;

10) выключают напряжение источников питания;

11) снимают все временно установленные перемычки;

12) переключатель SA10 («Захват») устанавливают в верхнее положение (указанное на рис. 15);

13) микропроцессор устаналивают в панельку;

14) включают напряжение источников питания;

15) убеждаются в работе микропроцессора по миганию светодиодов. Лучше всего это мигание заметно на светодиодах старших разрядов адреса;

16) переключатель SA10 («Захват») переводят в нижнее положение и при помощи переключателей SA1—SA8 и SA11—SA26 в первые три ячейки памяти заносят такую программу:

NOP
NOP
HLT

Для ввода первой команды (NOP) этой программы все переключатели SA11—SA26 и SA1—SA8 устанавливают в верхнее положение (0), задавая этим начальный адрес (0000000000000000) и код (00000000) команды NOP. Затем нажимают кнопку SB2 («Запись»), производя запись кода команды NOP в начальную (нулевую) ячейку памяти.

Переключателями SA11—SA26 устанавливают адрес следующей (первой) ячейки памяти путем установки переключателя SA11 в нижнее положение (1), задавая этим адрес 0000000000000001. Снова нажимают кнопку SB2, производя запись кода NOP в первую ячейку памяти.

Переключателями SA11—SA26 набирают адрес второй ячейки памяти (0000000000000010), а переключателями SA1—SA8 — код команды HLT (01110110). Нажимают кнопку SB2 для записи команды HLT во вторую ячейку памяти.

Диагностическая программа оказывается записанной в первые три ячейки памяти.

17) переключатель SA9 устанавливают в нижнее положение («Пошаговый») и нажимают кнопку SB3 («Сброс»). При этом должны погаснуть все светодиоды. Затем в пошаговом режиме работы нажимают кнопку SB1 («Шаг»). После первого нажатия кнопки на адресной шине должно установиться двоичное число 0000000000000001, а на шине данных 00000000. Второе нажатие должно вызвать появление чисел: на адресной шине — 0000000000000010, а на шине данных — 01110110. Последующее нажатие должно вызвать свечение всех светодиодов VD1—VD8. Нажимают кнопку SB3 («Сброс») и еще раз проверяют выполнение микропроцессором этой короткой диагностической программы;

18) проверяют работу микропроцессора в автоматическом режиме. Для этого пишут и вводят в память программу сложения двух чисел и размещение результата сложения в какой-либо ячейке памяти (находящейся в начальной, проверенной области памяти). Устанавливают переключатель SA9 в верхнее положение («Автоматический») и отрабатывают программу. Перед отработкой про-

граммы нужно нажать кнопку SB3 («Сброс»). Затем переключатель SA10 устанавливают в нижнее положение и проверяют, какие данные поступили в выбранную ячейку памяти, в которую занесен результат сложения чисел. Для этого переключателями SA11—SA26 устанавливают адрес ячейки памяти, а по свечению светодиодов VD1—VD8 определяют результат сложения. Если он соответствует действительному, значит микропроцессор хорошо выполняет программу и в автоматическом режиме работы. Если микропроцессор не выполняет каких-либо из перечисленных действий, то тщательно проверяют монтаж, измеряют напряжение и логические уровни в цепи, где предполагается неисправность. Проверяют также логические уровни на управляющих выводах микропроцессора (ГТ, ЗХ, ПЗХ) и управляющих выводах общей шины.

Расширение возможностей собранной микроЭВМ

Простейшая микроЭВМ, собранная в виде трех модулей (микропроцессорного, памяти и управления), позволяет вводить написанную программу и исполнять ее в автоматическом и пошаговом режимах, что является достаточным для предварительного знакомства с принципом работы микроЭВМ и элементами программирования. Однако в дальнейшем, освоив начальную стадию работы и потренировавшись в написании и отработке программ, читатель пожелает расширить возможности микроЭВМ, используя ее для практических целей (например, для программного управления бытовыми приборами, елочной гирляндой, музыкальным синтезатором, таймером и другими устройствами автоматики). С этой целью необходимо собрать модуль ввода— вывода для подключения к общей шине микроЭВМ различных датчиков и исполнительных механизмов в виде переключателей, кнопок, индикаторов, реле, светодиодов и др. Такой модуль лучше всего выполнить на специальной микросхеме типа К580ИК55, допускающей подключение самых разнообразных устройств ввода—вывода. Однако, учитывая дефицитность и сравнительно высокую стоимость этой микросхемы, а также значительное усложнение программы (напомним, что К580ИК55 является программируемым устройством ввода—вывода), на первых порах можно ограничиться более простым модулем ввода—вывода.

Структурная схема такого упрощенного модуля приведена на рис. 56 вклейки. Модуль состоит из шинного формирователя ШФД, осуществляющего ввод или вывод данных, порта вывода ПВЫВ и логического узла ЛУ, вырабатывающего управляющие сигналы.

Модуль не имеет порта ввода. Этим портом служат входы шинного формирователя ШФД. Если же сигналы ввода появляются весьма кратковременно и их необходимо запоминать, прежде чем вводить в микроЭВМ, то следует установить дополнительно порт ввода ПВВ, изображенный на рис. 56 вклейки пунктиром. Модуль имеет один канал (порт) ввода и один канал вывода,

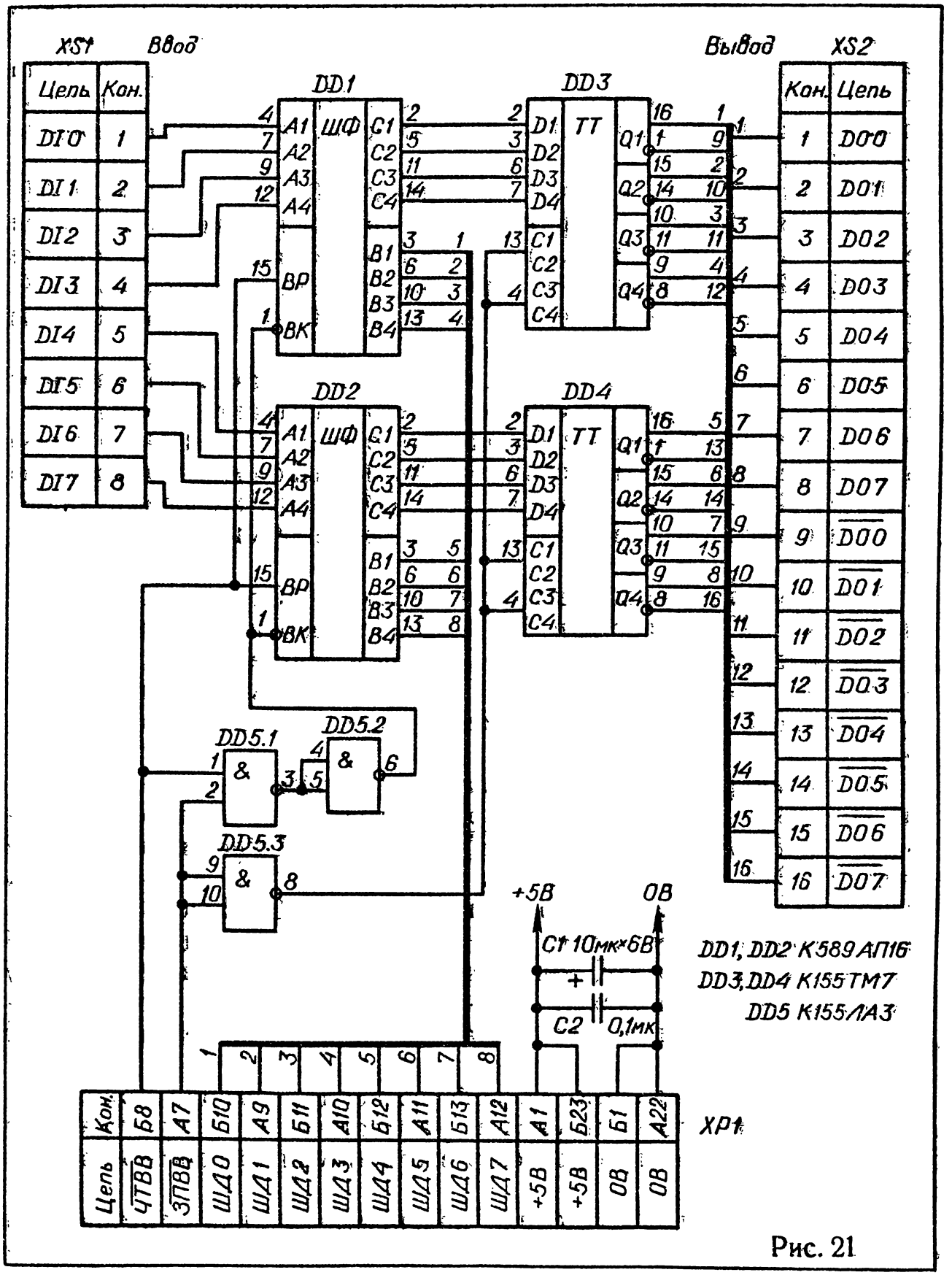


Рис. 21

поэтому нет дешифратора, определяющего номер порта. Если в дальнейшем потребуется развитие схемы, в виде подключения нескольких портов вывода или ввода, то следует добавить дешифратор ДША, который будет определять, какой из портов следует подключать к общей шине микроЭВМ.

Принципиальная схема модуля ввода—вывода приведена на рис. 21.

Шинные формирователи DD1—DD2 выполнены на микросхеме К589АП16. Порт вывода DD3—DD4 собран на триггерах типа К155ТМ7. Сигналы ввода поступают на разъем XS1, а сигналы вывода снимаются с разъема XS2. При этом на контактах разъема XS2 можно получить как прямое, так и инверсное значение сигналов вывода.

В режиме ввода данных на контакт Б8 $\overline{\text{ЧТВВ}}$ разъема XP1 общей шины приходит управляющий сигнал в виде уровня логического нуля. Этот сигнал поступает на выводы ВР микросхем DD1—DD2, что приводит к передаче данных с выводов А1—А4 на выводы В1—В4 и далее, на выводы ШД0—ШД7 общей шины, для ввода данных в микропроцессор.

В режиме вывода данных управляющий сигнал $\overline{\text{ЧТВВ}}$ изменяет свое значение с нуля на единицу и это вызывает изменение направления передаваемых данных с помощью DD1—DD2. Данные начинают поступать с выводов В1—В4 на выводы С1—С4 и далее на входы D1—D4 триггеров DD3—DD4. В это же время приходит управляющий сигнал $\overline{\text{ЗПВВ}}$ в виде уровня логического нуля.

Этот сигнал инвертируется микросхемой DD5.3 и поступает на выводы С1—С4 триггеров DD3—DD4, разрешая запоминания данных, подаваемых на выводы D1—D4 и осуществляя их выдачу на выводы Q1—Q4 и $\overline{\text{Q1}}$—$\overline{\text{Q4}}$. С этих выводов сигналы данных передаются на исполнительные механизмы, подключаемые к разъему XS2.

Когда микропроцессор осуществляет обмен данными с памятью, а не с устройствами ввода—вывода, то он посылает на контакты Б8 и А7 ($\overline{\text{ЧТВВ}}$ и $\overline{\text{ЗПВВ}}$) разъема XP1 уровни логической единицы. Эти уровни приходят на выводы 1 и 2 микросхемы DD5.1, вызывая нулевой уровень на выводе 3. Нулевой уровень инвертируется микросхемой DD5.2 и поступает в виде логической единицы на выводы $\overline{\text{ВК}}$ микросхем DD1—DD2, благодаря чему выводы В1—В4 микросхем DD1—DD2 переходят в высокоимпедансное состояние и модуль ввода—вывода отключается от общей шины.

Как было уже сказано, устройство ввода—вывода можно выполнить в виде ячеек памяти и разместить их в пространстве памяти так, чтобы микропроцессор воспринимал устройство ввода—вывода как память.

На практике при написании программ удобнее пользоваться специальными командами ввода (IN) и вывода (OUT), но при этом приходится формировать управляющие сигналы ЧТВВ и ЗПВВ, что усложняет аппаратную часть микроЭВМ.

На рис. 57 вклейки приведена функциональная схема модуля ввода—вывода, использующего порты ввода (ПВВ) и вывода (ПВЫВ) как условные ячейки памяти. Тот или иной порт вызывается микропроцессором путем установки на адресной шине ША нужного адреса порта. Этот адрес распознается дешифратором ДША, посылающим в устройство управления УУ сигнал, разре-

шающий подключение к шине данных ПВВ или ПВЫВ. По шине управления в УУ поступают также сигналы $\overline{\text{ЧТЗУ}}$ и $\overline{\text{ЗПЗУ}}$. Когда поступает сигнал $\overline{\text{ЧТЗУ}}$, то УУ включает ПВВ, а когда приходит сигнал $\overline{\text{ЗПЗУ}}$, то УУ разрешает включение ПВЫВ.

Принципиальная электрическая схема модуля ввода — вывода приведена на рис. 22. Разъем XS1 служит для подключения устройств ввода — переключателей, кнопок, датчиков и др. В качестве порта ввода использованы две микросхемы типа К155ЛП8, представляющих собой четыре управляемых усилителя с тремя логическими состояниями на выходе. Для работы К155ЛП8 в качестве усилителя на выводы 1, 4, 10 и 13 нужно подавать уровень логического нуля. Если же подавать уровень логической единицы, то выходы 3, 6, 8 и 11 переходят в высокоимпедансное состояние.

Разъем XS2 служит для подключения устройств вывода— светодиодов, индикаторов, реле, исполнительных механизмов и др.

Порт вывода выполнен на двух микросхемах типа К155ТМ7. Каждая микросхема К155ТМ7 содержит четыре триггера. Применение запоминающих триггеров вызвано тем, что микропроцессор выдает данные в течение очень короткого промежутка времени (микросекунды) и поэтому приходится запоминать данные перед тем, как их передавать устройству вывода.

Сигналом для запоминания служит уровень логической единицы, подаваемый на выводы 4 и 13 микросхем К155ТМ7.

Адрес порта ввода (условной ячейки памяти) выбран 7FFFH, а вывода — FFFFH, или (для наглядности) в двоичной системе соответственно — 0111111111111111 и 1111111111111111.

Такие адреса облегчают конструирование дешифратора при помощи широко распространенных микросхем типа К155ЛА2, представляющих собой логическую схему 8И—НЕ. У этой микросхемы на выходе (вывод 8) образуется уровень логического нуля только тогда, когда на все входы (выводы 1, 2, 3, 4, 5, 6, 11 и 12) подать уровень логической единицы.

При описании функциональной схемы модуля ввода—вывода для распознавания адресов порта ввода и порта вывода используется, как указывалось выше, дешифратор адреса ДША. В состав ДША входят микросхемы DD4, DD5, DD6 и DD2.1. Для дешифрации адреса 7FFFH (когда микропроцессор обращается к порту вывода) используются микросхемы DD5 и DD6, а для дешифрации адреса 7FFFH (обращение к порту ввода) — микросхемы DD5, DD4 и DD2.1. Микросхема DD2.1 необходима для инвертирования нуля в старшем разряде адреса ввода (0111111111111111). Сигналы с выхода дешифратора ДША (выводы 8 микросхемы DD4, DD5 и DD6) поступают на вход устройства управления УУ. На вход УУ из шины управления приходят также сигналы $\overline{\text{ЧТЗУ}}$ и $\overline{\text{ЗПЗУ}}$.

В состав УУ входят микросхемы DD1, DD2 (за исключением элемента DD2.1) и DD3.

Рассмотрим работу УУ. При обращении к порту вывода на его управляющие входы (выводы 4 и 13 микросхем DD9 и DD10) должен быть подан сигнал с уровнем логической единицы. Этот

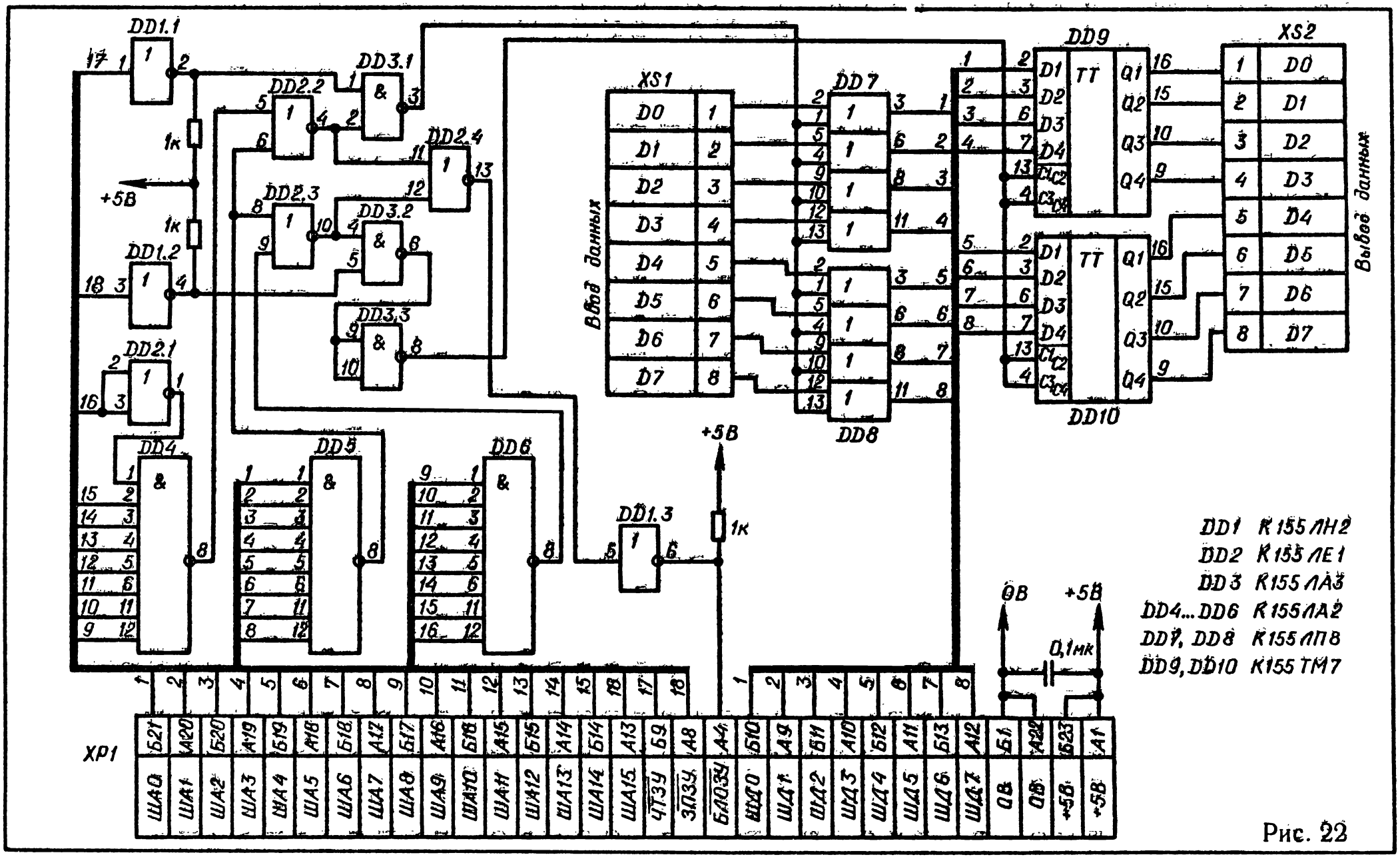

DD1.1
DD1.2
DD2.1
DD2.2
DD2.3
DD2.4
DD3.1
DD3.2
DD3.3
DD1.3
DD4
DD5
DD6
DD7
DD8
DD9
DD10
XS1
XS2
XP1
Ввод данных
Вывод данных
+5В
1к
0,1мк
0В
DD1 К155ЛН2
DD2 К155ЛЕ1
DD3 К155ЛА3
DD4...DD6 К155ЛА2
DD7, DD8 К155ЛП8
DD9, DD10 К155ТМ7

Рис. 22

сигнал появится на выходе УУ (вывод 8 микросхемы DD3.3) только в тот момент, когда микропроцессор выдаст на шину адреса ША адрес порта вывода FFFFH, а на шину управления — сигнал $\overline{\text{ЗПЗУ}}$. При этом с выхода дешифратора адреса (выводы 8 микросхем DD5 и DD6) сигналы с уровнем логического нуля поступают на вход УУ (выводы 8 и 9 микросхемы DD2.3). На выводе 10 микросхемы DD2.3 появится сигнал с уровнем логической единицы и поступит на вход схемы совпадения (вывод 4 микросхемы DD3.2). На второй вход (вывод 5 микросхемы DD3.2) поступит инвертированный микросхемой DD1.2 сигнал $\overline{\text{ЗПЗУ}}$ в виде уровня логической единицы. С выхода этой схемы совпадения (вывод 6 микросхемы DD3.2) сигнал с уровнем логического нуля поступит на вход инвертора DD3.3, а затем, уже инвертированный, имея уровень логической единицы, появится на выходе УУ.

Аналогично при обращении к порту ввода на выходе УУ (вывод 3 микросхемы DD3.1) сформируется сигнал с уровнем логического нуля только в тот момент, когда микропроцессор выдаст на шину адреса ША адрес порта ввода 7FFFH, а на шину управления ШУ — сигнал $\overline{\text{ЧТЗУ}}$. При этом есть некоторые отличия от обращения к порту вывода. Во-первых, не нужно инвертировать сигнал, поступающий с выхода схемы совпадения (вывод 3 микросхемы DD3.1), так как управляющий сигнал порта ввода должен иметь уровень логического нуля. Во-вторых, сигналы с уровнем логического нуля с выхода дешифратора адреса (выводы 8 микросхем DD4 и DD5) в этом случае поступают соответственно на вход УУ (выводы 5 и 6 микросхемы DD2.2). И, наконец, сигнал $\overline{\text{ЧТЗУ}}$ инвертируется микросхемой DD1.1 и поступает на вход схемы совпадения (вывод 1 микросхемы DD3.1) в виде логической единицы.

Как было указано в разделе 3.1., сигнал $\overline{\text{БЛ ОЗУ}}$ служит для отключения основного модуля памяти от общей шины при записи или считывании данных с других устройств, подключенных непосредственно к общей шине и использующих сигналы $\overline{\text{ЗПЗУ}}$ и $\overline{\text{ЧТЗУ}}$. Поэтому в УУ рассматриваемого модуля ввода—вывода предусмотрена схема формирования сигнала БЛ ОЗУ при обращении к порту ввода или порту вывода. Эта схема собрана на элементах DD2.4 и DD1.3.

Обратим внимание читателей еще на одну особенность модуля ввода—вывода, которая присуща и другим модулям микроЭВМ, использующим логические элементы с открытым коллектором на выходе. А именно, для улучшения помехозащищенности и устойчивой работы модуля ввода—вывода, выходы логических элементов с открытым коллектором DD1.1, DD1.2 и DD1.3 (соответственно выводы 2, 4 и 6) соединены через резистор 1 КОм с шиной +5 В источника питания.

Помимо рассмотренных способов включения устройств ввода—вывода существует еще один, применяемый в простейших микроЭВМ, содержащих ограниченный объем памяти и малое количество портов ввода—вывода. Для работы с памятью небольшого объема требуется ограниченное число адресных выводов. Поэтому

оставшиеся лишними старшие разряды адресных выводов используются для непосредственной передачи сигналов выбора того или иного порта ввода или вывода.

Например, микроЭВМ имеет память в 4 Кбайта. Для работы с такой памятью требуется 12 разрядов адресной шины. Оставшиеся 4 разряда можно использовать для включения четырех портов ввода или вывода, передавая по четырем старшим разрядам адресных шин сигналы включения (1 или 0) нужного порта. При этом для включения порта ввода дополнительно используется сигнал $\overline{\text{ЧТЗУ}}$, а для включения порта вывода — сигнал $\overline{\text{ЗПЗУ}}$.

Помимо добавления модуля ввода—вывода, представляется возможным приступить к выполнению второго этапа сборки микроЭВМ, позволяющего превратить нашу микроЭВМ в персональный компьютер.

С этой целью необходимо:

1) увеличить объем ОЗУ до 12—16 Кбайт;
2) добавить в память ПЗУ, имеющие объем 6—8 Кбайт;
3) собрать модуль сопряжения с магнитофоном, позволяющим записывать и считывать программы с помощью бытового магнитофона;
4) собрать модуль сопряжения с телевизором, позволяющим использовать телевизор в качесте дисплея;
5) изготовить клавишный пульт управления, имеющий модуль сопряжения;
6) увеличить мощность блока питания с учетом мощности, потребляемой дополнительными модулями;
7) изготовить простейший программатор, позволяющий вручную программировать ПЗУ.

Перечисленные работы выполняют в таком же порядке, как и работы первого этапа сборки, а именно:

1) выбирают и творчески дорабатывают принципиальные схемы дополнительных модулей в зависимости от поставленных целей модернизации, имеющегося опыта и наличия деталей;
2) решают конструктивные вопросы по размещению и монтажу деталей на платах модулей;
3) устанавливают и монтируют детали на платах модулей;
4) переделывают блок питания в соответствии с повышенной силой потребляемого тока;
5) испытывают и налаживают дополнительные модули;
6) испытывают и налаживают всю микроЭВМ в целом.

В журналах «Радио», «Моделист-конструктор» и «Микропроцессорные средства и системы» опубликованы принципиальные схемы и краткие описания четырех типов микроЭВМ, пригодных для радиолюбительской сборки и представляющих собой простейшие персональные компьютеры. Эти микроЭВМ получили такие названия: «Микро-80» [24], «Радио-86РК» [45], «Ириша» [44] и «Специалист» [8]. Все они собраны на микропроцессоре типа КР580ИК80А и поэтому хорошо согласовываются с нашей простейшей микроЭВМ, т. е. модули нашей микроЭВМ могут быть

использованы для сборки этих компьютеров. Выбор схем дополнительных модулей зависит от наличия деталей. Если ассортимент деталей широк, то целесообразно придерживаться схем модулей «Ириша» и «Радио-86РК», а при недостаточном ассортименте следует пользоваться схемами «Микро-80» или «Специалист».

Перед расширением модуля памяти необходимо точно и однозначно распределить все пространство (объем) памяти, начиная от нулевой ячейки (0000H) и кончая последней ячейкой (FFFFH), т. е. решить, какой вид памяти будет занимать определенное место во всем его объеме. Например, установить, что ОЗУ пользователя будет занимать место с 0000H по 35FFH, резервная область с 3600H по 3FFFH, неиспользуемая область с 4000H по DFFFH, ОЗУ курсора с E000H по E7FFH, ОЗУ экрана с E800H по EFFFH, ОЗУ монитора с F500H по F7FFH, ПЗУ монитора с F800H по FFFFH.

ПЗУ следует выполнять на микросхемах с ультрафиолетовым стиранием типа К573РФ1, К573РФ2, К573РФ4 или К573РФ5. Корпуса этих микросхем имеют малые размеры и поэтому свободно размещаются на плате прежнего модуля памяти (если, конечно, было предусмотрено для этого место). Корпуса дополнительного ОЗУ приходится размещать на отдельной плате. Модуль сопряжения с магнитофоном, позволяющий записывать и хранить программу на магнитофонной ленте, является наиболее простым из всех дополнительных модулей. Этот модуль состоит из операционного усилителя, нескольких микросхем, резисторов и конденсаторов. Все детали модуля занимают очень мало места и легко размещаются на плате любого модуля. Лучше всего монтировать их на плате одного из модулей памяти или клавишного пульта управления.

Модуль управления рассмотренной простейшей микроЭВМ используется только при наладке или при поиске неисправностей в работе модернизированной микроЭВМ. После наладки модуль управления отключается, а уже управление выполняется при помощи нового модуля — клавишного пульта управления. Этот пульт можно изготовить следующим образом. На листе плотной бумаги рисуют расположение клавиш в соответствии с принятым стандартом [24, 45]. Снизу, под каждой нарисованной клавишей (см. рис. 72 вклейки) приклеивают картонный кружок диаметром 10 мм и толщиной 2—3 мм, а под кружком устанавливают микропереключатель типа МП-1, являющийся составной частью широкораспространенного малогабаритного тумблера МТ-1 или малогабаритной кнопки КМ-1. При нажатии на рисунок клавиши картонный кружок передает давление кнопке микропереключателя, в результате чего происходит замыкание контактов клавиши. Для придания жесткости под бумагу с нарисованными клавишами помещают лист картона с отверстиями диаметром 20 мм. Центры этих отверстий должны совпадать с центрами картонных кружков, наклеенных на бумаге.

Наиболее сложным из всех дополнительных модулей является модуль сопряжения с телевизором, позволяющий использовать

телевизор в качестве дисплея. В целях упрощения схемы и конструкции этого модуля разработаны специальные БИС (КР580 ВГ75). Однако эти БИС очень дефицитные и дорогостоящие. При их отсутствии модуль сопряжения с телевизором следует выполнять по схеме «Микро-80» или же воспользоваться схемами журнала «Радио» [20], где рассказано, как заменить микросхему К580ВГ75 набором простых, широко используемых микросхем.

Простейшие программаторы, позволяющие вручную программировать ПЗУ (монитора, интерпретатора Бейсик и др.) рассмотрены при описании «Микро-80» и «Специалист». На рис. 58 и 60 вклейки приведены примеры (варианты) конструктивного расположения плат модулей микроЭВМ первого и второго этапов сборки. В зависимости от материально-технических условий, опыта и умения, конструкция может быть иной.

В первом случае (рис. 58 вклейки) общая шина с разъемами укрепляется на фанерной панели, имеющей размер 600 × 360 мм. Панель закрывается плоским коробом из оргстекла. Размер короба 600 × 360 × 60 мм.

В левые верхние разъемы общей шины устанавливают модули микропроцессора и памяти. В левый нижний включают разъем кабеля модуля управления, а в правый нижний — разъем кабеля блока питания. Верхние правые разъемы служат для включения размножителя общей шины и дополнительных модулей при совершенствовании структуры микроЭВМ.

Цифры в кружках на рисунке 58 вклейки обозначают такие элементы конструкции микроЭВМ:

1 — фанерная панель;
2 — общая шина;
3 — разъемы, соединяющие общую шину с модулем управления;
4 — разъем блока питания;
5 — микропроцессорный модуль;
6 — модуль памяти;
7 — место для включения дополнительных модулей;
8 — кабель, соединяющий общую шину с модулем управления;
9 — кабель, соединяющий общую шину с блоком питания;
10 — модуль управления.

Как показала практика работы с микроЭВМ, расположение модулей в одну линию и в одной плоскости оказывается очень удобным для монтажа, налаживания и выполнения учебной работы с микроЭВМ.

В целях дальнейшего усовершенствования микроЭВМ тумблерный модуль управления может быть заменен клавишным, допускающим ввод программы в шестнадцатиричном виде. Внешний вид клавишного пульта управления приведен на рис. 59 вклейки.

Конструктивно клавиши пульта выполнены так, как об этом сказано выше. 4 верхних индикатора пульта указывают адрес, установленный на шине адреса, а 2 нижних сообщают данные, находящиеся на шине данных. Клавишный пульт собран в соответствии с рекомендациями, приведенными в [31]. При этом состав-

лена и помещена в ПЗУ программа, позволяющая преобразовать функцию нажатой клавиши в команду, подаваемую в микропроцессор в виде машинного двоичного кода.

Помимо 16 клавиш (0—F), служащих для набора адреса или данных, пульт имеет 9 клавиш управления режимом работы микроЭВМ.

Модули микроЭВМ второго этапа сборки (рис. 60 вклейки) располагаются в каркасе, выполненном из алюминиевого уголка 15 × 15 мм. Размеры каркаса 380 × 250 × 80 мм. Каркас устанавливается в чемодан типа «Дипломат». На рис. 60 вклейки внутри «Дипломата» справа виден блок питания микроЭВМ, а слева — рабочие модули. Верхний левый модуль — микропроцессорный, а верхний правый — расширенной памяти (динамическое ОЗУ на 16 Кбайт, выполненное на микросхемах К565РУ3). Ниже, под микропроцессорным модулем, расположен модуль сопряжения с телевизором, а правее его, под модулем памяти, находится модуль ОЗУ и ПЗУ монитора. Если модули сопряжения с телевизором, расширенной памяти и монитора вынуть из «Дипломата» и установить в разъемы микроЭВМ первого этапа сборки, а также заменить тумблерный пульт управления клавишным, то микроЭВМ первого этапа будет выполнять те же функции, что и микроЭВМ второго этапа, собранная в «Дипломате».

В заключение укажем составные части микроЭВМ второго этапа сборки, обозначенные на рис. 60 вклейки цифрами в кружках:

1 — микроЭВМ, установленная в чемодане «Дипломат»;
2 — кабель, соединяющий микроЭВМ и клавиатуру;
3 — кабель, соединяющий микроЭВМ и телевизор;
4 — клавиатура;
5 — магнитофон;
6 — телевизор;
7 — кассета магнитофона.

При разработке схем и конструкций модулей микроЭВМ весьма полезно ознакомиться с такой литературой: [2, 3, 8, 19, 22, 24, 33, 39, 44, 45, 61].

1. Система команд МП—К580

Системой команд называют перечень (список, набор) команд, который может выполнить данный микропроцессор. Обычно система команд оформляется в виде таблицы, указывающей действие каждой команды, ее сокращенное название, двоичный код и другие данные.

Микропроцессоры разных типов имеют свои несколько отличающиеся друг от друга системы команд. В табл. 1 (с. 133) приведена система команд МП—К580. Поясним назначение колонок (граф) этой таблицы.

В колонке 1 перечислены порядковые номера команд.

В колонке 2 приведены описания команд, выполненные в сжатой, образной форме, рассчитанной на читателя, уже знакомого с данной командой. Пользоваться этими описаниями рекомендуется лишь для уточнения отдельных действий команды. Первоначальное же знакомство следует выполнять по более простому и общедоступному изложению, приведенному в тексте данного раздела.

В колонке 3 указаны условные обозначения той части команды, которая называется **операцией.**

В колонке 4 приведены условные обозначения другой части команды, называемой **операндом.**

В колонке 5 представлены форматы команд, позволяющие определять структуру и длину команды, а также узнавать двоичный код, который вводится в микропроцессор для выполнения команды.

В колонке 6 при помощи двух цифр, разделенных запятой, указано количество машинных циклов (первая цифра) и машинных тактов (вторая цифра), затрачиваемых микропроцессором на выполнение каждой команды.

В колонке 7 показаны изменения состояний триггеров (Z, S, P, C, AC) **регистра признаков** F при выполнении команды. Если состояние триггера не изменяется, то в колонке 7 ставится прочерк (—). При изменении состояния триггера в соответствии с результатом операции пишется плюс (+). Если триггер сбрасывается в нуль, то отмечается (0), а если устанавливается в единицу, то пишется (1).

В колонке 8 приведены комментарии (справочные данные, дополнения и разъяснения).

Пользуясь табл. 1 и правилами написания команд, можно составлять программы для МП—К580 в машинном двоичном коде (на машинном языке) и на языке ассемблера.

Прежде чем перечислять команды, входящие в табл. 1, коротко рассмотрим некоторые правила их написания.

Большинство команд состоит из двух частей.

Первая часть отвечает на вопрос: «Что делать?» и указывает на те действия (операции), которые должен выполнить микропроцессор. Например: «Сложить», «Вычесть», «Переслать» и др. Эта часть команды называется операцией (рис. 61, *а* вклейки) или кодом операции (см. рис. 61, *г* вклейки), если команда выражена в двоичном машинном коде.

Вторая часть команды отвечает на вопросы: «Какие данные обрабатываются?», «Где находятся эти данные?» или «Каков их адрес?» Эта часть команды получила название операнд (рис. 61, *а* вклейки).

Операнд может быть записан тремя способами: в виде числа (в двоичной, десятичной или шестнадцатеричной форме); в виде места, где хранится число (например, регистр B); в виде адреса ячейки памяти или порта, в которых находится число. Из трех перечисленных способов записи чаще всего пользуются адресом, что делает программу более универсальной и пригодной для многих аналогичных случаев. По этой причине операнд иногда именуют адресной частью команды или просто **адресом** (рис. 61, *б* вклейки), говоря, что команда состоит из операции и адреса.

Встречаются команды, состоящие только из первой части, то есть не имеющие операнда (рис. 61, *в* вклейки). К таким командам относятся некоторые служебные команды, например: «Запретить прерывание», «Остановить выполнение программы» и др.

Таблица 1

Система команд микропроцессора К580ИК80

№ п/п	Описание команд	Условное обозначение операции	Условное обозначение операнда	Двоичный код и формат команды	Количество машинных циклов M, тактов T (M, T)	Состояние регистра признаков F (Z, S, P, C, AC)	Комментарии
1	2	3	4	5	6	7	8
1	Переслать в регистр R1 содержимое регистра R2	MOV	R1, R2	01 код R1 код R2	1,5	— — — — —	Здесь M — ячейка памяти, адрес которой находится в регистровой паре HL R Код R Код A 111 E 011 B 000 H 100 C 001 L 101 D 010 M 110
2	Переслать в регистр R содержимое ячейки памяти M	MOV	R, M	01 код R 110	2,7	— — — — —	1) Предварительно поместить в регистровую пару HL адрес ячейки памяти M 2) Регистром R может быть только регистр A, B, C, D или E
3	Переслать в ячейку памяти M содержимое регистра R	MOV	M, R	01 110 код R	2,7	— — — — —	См. комментарии команды 2
4	Загрузить в регистр R число K, записанное во 2-м байте команды	MVI	R, K	00 код R 110 число K	2,7	— — — — —	Значение кода R см в комментариях команды 1

Продолжение табл. 1

1	2	3	4	5	6	7	8
5	Загрузить в ячейку памяти M число K, записанное во 2-м байте команды	MVI	M, K	0 0 1 1 0 1 1 0 число K	3,10	— — — — —	Предварительно поместить в регистровую пару HL адрес ячейки памяти M
6	Загрузить в регистровую пару P число K, записанное во 2-м и в 3-м байтах команды	LXI	P, K	00 код P 0001 мл. разряды K ст. разряды K	3,10	— — — — —	<table><tr><th>P</th><th>Регистровая пара</th><th>Код P</th></tr><tr><td>B</td><td>BC</td><td>00</td></tr><tr><td>D</td><td>DE</td><td>01</td></tr><tr><td>H</td><td>HL</td><td>10</td></tr><tr><td>S</td><td>SP</td><td>11</td></tr></table>
7	Переслать в аккумулятор содержимое ячейки памяти M, адрес которой записан во 2-м и 3-м байтах команды	LDA	ADR	0 0 1 1 1 0 1 0 мл. полуадрес ст. полуадрес	4,13	— — — — —	Адрес ячейки памяти M записать во 2-м и 3-м байтах команды
8	Переслать в ячейку памяти M, адрес которой записан во 2-м и 3-м байтах команды, содержимое аккумулятора	STA	ADR	0 0 1 1 0 0 1 0 мл. полуадрес ст. полуадрес	4,13	— — — — —	См. комментарии команды 7
9	Переслать в аккумулятор содержимое ячейки памяти M, адрес которой находится в регистровой паре P	LDAX	P	00 код P 1010	2,7	— — — — —	1) Значение кода регистровой пары P см в комментариях команды 6. 2) Могут использоваться только регистровые пары BC и DE

10	Переслать в ячейку памяти M, адрес которой находится в регистровой паре P, содержимое аккумулятора	STAX	P	00 код P 0010	2,7	— — — — —	См. комментарии команды 9
11	Переслать в аккумулятор данные из порта N, адрес которого указан во 2-м байте команды	IN	N	1 1 0 1 1 0 1 1 адр. порта ввода	3,10	— — — — —	
12	Переслать в порт N, адрес которого указан во 2-м байте команды, данные из аккумулятора	OUT	N	1 1 0 1 0 0 1 1 адр. порта вывода	3,10	— — — — —	
13	Переслать в стек содержимое регистровой пары P	PUSH	P	11 код P 0101	3,11	— — — — —	Регистровая пара SP не может быть использована в данной команде
14	Вернуть из стека содержимое в регистровую пару P	POP	P	11 код P 0001	3,10	— — — — —	См. комментарии команды 13
15	Переслать в стек содержимое регистров A и F	PUSH	PSW	1 1 1 1 0 1 0 1	3,11	— — — — —	
16	Вернуть из стека содержимое в регистры A и F	POP	PSW	1 1 1 1 0 0 0 1	3,10	+ + + + +	
17	Обменять содержимое регистровых пар DE и HL	XCHG	—	1 1 1 0 1 0 1 1	1,4	— — — — —	
18	Обменять содержимое двух верхних ячеек стека с регистровой парой HL	XTHL	—	1 1 1 0 0 0 1 1	5,18	— — — — —	

Продолжение табл. 1

1	2	3	4	5	6	7	8
19	Переслать содержимое регистровой пары HL в указатель стека SP	SPHL	—	1 1 1 1 1 0 0 1	1,5	— — — — —	
20	Загрузить регистровую пару HL содержимым двух соседних ячеек памяти	LHLD	—	0 0 1 0 1 0 1 0 мл. полуадр. 1 яч. ст. полуадр. 1 яч.	5,16	— — — — —	В регистр L передается содержимое первой ячейки памяти, адрес которой записан во 2-й и 3-й байты команды. В регистр H передается содержимое второй ячейки памяти, адрес которой на единицу больше адреса первой ячейки
21	Загрузить две соседние ячейки памяти содержимым регистров H и L	SHLD	—	0 0 1 0 0 0 1 0 мл. полуадр. 1 яч. ст. полуадр. 1 яч.	5,16	— — — — —	В первую ячейку памяти, адрес которой записан во 2-м и 3-м байтах команды, передается содержимое регистра L Во вторую ячейку памяти, адрес которой на единицу больше адреса первой ячейки, передается содержимое регистра H
22	Сложить содержимое аккумулятора и регистра R	ADD	R	10000 код R	1,4	+ + + + +	1) Значение кода R (см. в комментариях команды 1) 2) Результат вычислений размещается в аккумуляторе

23	Сложить содержимое аккумулятора и ячейки памяти M, адрес которой находится в регистровой паре HL	ADD	M	1 0 0 0 0 1 1 0	2,7	+ + + + +	1) Предварительно поместить в регистровую пару HL адрес ячейки памяти M 2) Результат вычислений размещается в аккумуляторе
24	Сложить содержимое аккумулятора регистра R и признака переноса	ADC	R	10001 код R	1,4	+ + + + +	См. комментарии команды 22
25	Сложить содержимое аккумулятора, признака переноса и ячейки памяти M, адрес которой находится в регистровой паре HL	ADC	M	1 0 0 0 1 1 1 0	2,7	+ + + + +	См. комментарии команды 23
26	Сложить содержимое аккумулятора с числом K, записанным во 2-м байте команды	ADI	K	1 1 0 0 0 1 1 0 число K	2,7	+ + + + +	
27	Сложить содержимое аккумулятора с признаком переноса и с числом K, записанном во 2-м байте команды	ACI	K	1 1 0 0 1 1 1 0 число K	2,7	+ + + + +	
28	Вычесть из содержимого аккумулятора содержимое регистра R	SUB	R	10010 код R	1,4	+ + + + +	См. комментарии команды 22
29	Вычесть из содержимого аккумулятора содержимое ячейки памяти M, адрес которой находится в регистровой паре HL	SUB	M	1 0 0 1 0 1 1 0	2,7	+ + + + +	См. комментарии команды 23

Продолжение табл. 1

1	2	3	4	5	6	7	8
30	Вычесть из содержимого аккумулятора содержимое регистра R с учетом признака переноса	SBB	R	10011 код R	1,4	+ + + + +	См. комментарии команды 22
31	Вычесть из содержимого аккумулятора содержимое ячейки памяти M, адрес которой находится в регистровой паре HL, с учетом признака переноса	SBB	M	1 0 0 1 1 1 1 0	2,7	+ + + + +	См. комментарии команды 23
32	Вычесть из содержимого аккумулятора число K, записанное во 2-м байте команды	SUI	K	1 1 0 1 0 1 1 0 число K	2,7	+ + + + +	Результаты вычислений помещаются в аккумулятор
33	Вычесть из содержимого аккумулятора число K, записанное во 2-м байте команды, и признак переноса	SBI	K	1 1 0 1 1 1 1 0 число K	2,7	+ + + + +	См. комментарии команды 32
34	Увеличить на 1 содержимое регистра R	INR	R	00 код R 100	1,5	+ + + − +	1) Значение кода R (см. в комментарии команды 1)
35	Увеличить на 1 содержимое ячейки памяти M, адрес которой находится в регистровой паре HL	INR	M	0 0 1 1 0 1 0 0	3,10	+ + + − +	1) Предварительно поместить в регистровую пару HL адрес ячейки памяти M

36	Увеличить на 1 содержимое регистровой пары P	INX	P	00 код P 0011	1,5	— — — — —	Значение кода P (см. комментарии команды 6)
37	Уменьшить на 1 содержимое регистра R	DCR	R	00 код R 101	1,5	+ + + — +	Значение кода R (см. комментарии команды 1)
38	Уменьшить на 1 содержимое ячейки памяти M, адрес которой находится в регистровой паре HL	DCR	M	0 0 1 1 0 1 0 1	3,10	+ + + — +	Предварительно поместить в регистровую пару HL адрес памяти
39	Уменьшить на 1 содержимое регистровой пары P	DCX	P	00 код P 1011	1,5	— — — — —	Значение кода P (см. комментарии команды 6)
40	Сложить содержимое регистровой пары HL с содержимым регистровой пары P	DAD	P	00 код P 1001	3,10	— — — + —	1) Значение кода P (см. комментарии команды 6) 2) Результат вычисления размещается в регистровой паре HL
41	Коррекция десятичная	DAA	—	0 0 1 0 0 1 1 1	1,4	+ + + + +	Применяется для выполнения двоично-десятичного сложения
42	Выполнить логическое умножение (И) содержимого аккумулятора и регистра R	DNA	R	10100 код R	1,4	+ + + 0 +	См. комментарии команды 22
43	Выполнить логическое умножение (И) содержимого аккумулятора и ячейки памяти M, адрес которой находится в регистровой паре HL	ANA	M	1 0 1 0 0 1 1 0	2,7	+ + + 0 +	См. комментарии команды 23

Продолжение табл. 1

1	2	3	4	5	6	7	8
44	Выполнить логическое умножение (И) содержимого аккумулятора и числа K, записанного во 2-м байте команды	ANI	K	1 1 1 0 0 1 1 0 число K	2,7	+ + 0 0 +	
45	Выполнить логическое сложение (ИЛИ) содержимого аккумулятора и регистра	ORA	R	10110 код R	1,4	+ + 0 0 +	Значение кода R (см. комментарии команды 1)
46	Выполнить логическое сложение (ИЛИ) содержимого аккумулятора и ячейки памяти M, адрес которой находится в регистровой паре HL	ORA	M	1 0 1 1 0 1 1 0	2,7	+ + 0 0 +	См. комментарии команды 23
47	Выполнить логическое сложение (ИЛИ) содержимого аккумулятора и числа K, записанного во 2-м байте команды	ORI	K	1 1 1 1 0 1 1 0 число K	2,7	+ + 0 0 +	См. комментарии команды 32
48	Выполнить логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимыми аккумулятора и регистра R	XRA	R	10101 код R	1,4	+ + 0 0 +	См. комментарии команды 22
49	Выполнить логическую операцию ИСКЛЮЧАЮ-	XRA	M	1 0 1 0 1 1 1 0	2,7	+ + 0 0 +	См. комментарии команды 23

	ЩЕЕ ИЛИ над содержимыми аккумулятора и ячейки памяти M, адрес которой находится в регистровой паре HL						
50	Выполнить операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и числом K, записанным во 2-м байте команды	XRI	K	1 1 1 0 1 1 1 0 число K	2,7	+ + 0 0 +	
51	Инвертировать содержимое аккумулятора	CMA	—	0 0 1 0 1 1 1 1	1,4	— — — — —	
52	Сравнить содержимое аккумулятора и регистра R	CMR	R	10111 код R	1,4	+ + + + +	Значение кода R (см. комментарии команды 1)
53	Сравнить содержимое аккумулятора и ячейки памяти M, адрес которой записан в регистровой паре HL	CMP	M	1 0 1 1 1 1 1 0	2,7	+ + + + +	Предварительно поместить в регистровую пару HL адрес ячейки памяти M
54	Сравнить содержимое аккумулятора и числа K, записанного во 2-м байте команды	CPI	K	1 1 1 1 1 1 1 0 число K	2,7	+ + + + +	
55	Сдвинуть влево на один разряд циклически содержимое аккумулятора с учетом признака переноса	RAL	—	0 0 0 1 0 1 1 1	1,4	— — — + —	
56	Сдвинуть влево на один разряд содержимое аккумулятора	RLC	—	0 0 0 0 0 1 1 1	1,4	— — — + —	

Продолжение табл. 1

1	2	3	4	5	6	7	8
57	Сдвинуть вправо на один разряд циклически содержимое аккумулятора с учетом признака переноса	RAR	—	0 0 0 1 1 1 1 1	1,4	— — — + —	
58	Сдвинуть вправо на один разряд содержимое аккумулятора	RRC	—	0 0 0 0 1 1 1 1	1,4	— — — + —	
59	Установить в 1 признак переноса C	STC	—	0 0 1 1 0 1 1 1	1,4	— — — 1 —	
60	Безусловный переход по адресу, записанному во 2-м и 3-м байтах команды	JMP	ADR	1 1 0 0 0 0 1 1 мл. полуадр. перех. ст. полуадр. перех.	3,10	— — — — —	
61	Безусловный переход по адресу, находящемуся в регистровой паре H	PCHL	—	1 1 1 0 1 0 0 1	1,5	— — — — —	Предварительно поместить в регистровую пару HL адрес перехода
62	Безусловный переход к выполнению подпрограммы, адрес которой (начальный) указан во 2-м и 3-м байтах команды	CALL	ADR	1 1 0 0 1 1 0 1 мл. полуадр. перех. ст. полуадр. перех.	5,7	— — — — —	
63	Условный переход по адресу, записанному во 2-м и 3-м байтах в зависимости от состояния T триггеров регистра признаков F	J(F)	ADR	11 код T 010 мл. полуадр. перех. ст. полуадр. перех.	3,10	— — — — —	

Состояние	Код T	Состояние	Код T
Z=0	000	P=0	100
Z=1	001	P=1	101
C=0	010	S=0	110
C=1	011	S=1	111

64	Условный переход к выполнению подпрограммы, начальный адрес которой записан во 2-м и 3-м байтах команды в зависимости от состояния T триггеров регистра признаков F	C(F)	ADR	11 код T 100 мл. полуадр. перех. ст. полуадр. перех.	3,5 11,17	— — — — —	См. комментарии команды 63
65	Безусловный возврат из подпрограммы	RET	—	1 1 0 0 1 0 0 1	3,10	— — — — —	
66	Условный возврат из подпрограммы в зависимости от состояния T триггеров регистра признаков F	R(F)	ADR	11 код T 000	1,3 5,11	— — — — —	Значение кода T (см. комментарии команды 63)
67	Переход к специальной подпрограмме режима прерывания	RST	X	11 код X 111	3,11		X — трехразрядное двоичное число, определяющее одну из восьми возможных подпрограмм прерываний микропроцессора
68	Разрешить прерывание	EI	—	1 1 1 1 1 0 1 1	1,4	— — — — —	
69	Запретить прерывание	DI	—	1 1 1 1 0 0 1 1	1,4	— — — — —	
70	Пустая операция	NOP	—	0 0 0 0 0 0 0 0	1,4	— — — — —	
71	Остановить выполнение программы	HLT	—	0 1 1 1 0 1 1 0	1,7	— — — — —	

Чтобы упростить написание и облегчить запоминание команды, ее первую часть — операцию — записывают в виде сокращенных английских слов, указывающих выполняемые действия. При этом употребляют сокращение до двух, трех или четырех букв (очень редко — до пяти). Например, операцию ввода данных из внешнего устройства обозначают двумя буквами IN, операцию пересылки данных — тремя буквами MOV, а вызов подпрограммы — четырьмя буквами CALL.

При написании второй части команды—операнда — придерживаются таких правил. Если операнд обозначается в виде числа, то обязательно указывают его систему счисления. После двоичного числа пишут символ B, после десятичного — символ D и после шестнадцатиричного — символ H.

Для определения места, где находятся данные, применяют условные обозначения. Расположение данных в регистрах общего назначения (РОН) микропроцессора обозначают латинской буквой R или буквой (A, B, C, D, E, H, L), присвоенной каждому регистру. Если данные находятся в ячейке памяти, то пишут букву M.

Когда операнд записывают в виде адреса, то эту запись выполняют при помощи символа ADR или какого-либо условного символического слова, состоящего из четырех-пяти букв (например, MULTI).

Команды бывают простые и сложные, короткие и длинные. В системе команд МП — К580 употребляются команды длиною в один, два и три байта.

В однобайтовых командах старшие разряды предназначаются для размещения кода операции, а младшие — для указания кода регистров (рис. 62, *в* вклейки). Количество разрядов, выделяемое под каждую из составляющих частей команды, бывает различным, зависящим от типа команды.

В двухбайтовых командах первый байт полностью отводится под код операции (рис. 62, *б* вклейки), а второй — служит для записи данных (операнда) в виде числа, кода регистров или адреса.

Восемь бит второго байта позволяют записать наибольшее число, равное 255 ($11111111_2 = 255_{10}$). При необходимости ввода чисел, превышающих 255 (до 65 535), используют трехбайтовую команду (рис. 62, *в* вклейки), у которой первый байт предназначается для размещения кода операции, а второй и третий — для размещения шестнадцатиразрядных двоичных чисел или шестнадцатиразрядного адреса ячейки памяти, где хранятся данные.

При написании команд очень важно правильно выбрать способ адресации данных, который требовал бы наименьший объем памяти для размещения данных и наименьшее время исполнения команды. При этом выборе следует учесть, что МП—К580 позволяет применять такие способы адресаций: прямой, регистровый, косвенный, непосредственный и неявный.

При прямой адресации адрес места, где хранятся данные, помещают сразу же за кодом операции. Этим адресом может быть номер порта устройства ввода—вывода или номер ячейки памяти. В первом случае команда будет двухбайтовой (рис. 63, *а* вклейки), а во втором — трехбайтовой (рис. 63, *б* вклейки). Прямая адресация наиболее простая, но в то же время и наиболее неэкономная из-за достаточно длинных командных слов, приводящих к увеличению занимаемого объема памяти и удлинению времени выполнения программы.

При регистровой адресации вслед за кодом операции указывают код регистра, где хранятся данные (рис. 64 вклейки). Этот код занимает три разряда для каждого регистра, поэтому команды получаются однобайтными и выполняются достаточно быстро.

При косвенной адресации указывают код регистров, в которых находится адрес ячейки памяти, содержащей данные (рис. 65 вклейки). Косвенный способ адресации позволяет компактно адресовать пространство памяти при помощи коротких однобайтных команд. Поэтому применение такого способа встречается довольно часто.

При непосредственной адресации данные в виде восьмиразрядного двоичного числа непосредственно размещают во втором байте команды (рис. 66, *а* вклейки). Если число имеет длину более восьми разрядов, то его записывают во втором и в третьем байтах (рис. 66, *б* вклейки). При этом младшие разряды числа заносят во второй байт, а старшие — в третий. Команды с непосредственной адресацией обозначают буквой I, написанной в конце мнемонического слова операции, а само

слово уменьшают на одну букву. Например, вместо MOV пишут MVI, вместо ADD – ADI и вместо ORA – ORI.

При неявной (подразумеваемой) адресации один из операндов должен находиться во внутреннем регистре микропроцессора, чаще всего в регистре А, называемом аккумулятором. Поэтому нет необходимости дополнительно сообщать его адрес. Команды с неявной адресацией получаются более короткими, что сокращает их написание и выполнение. Неявный способ адресации часто встречается в командах МП—К580, который устроен так, что все логические и арифметические операции выполняются только через аккумулятор, когда один из операндов должен находиться в аккумуляторе еще до выполнения команды.

Существует также смешанный способ адресации. Например, команда CALL использует косвенную адресацию и прямую.

Приведенные описания способов адресации могут показаться вначале не совсем понятными. Однако позже, по мере дальнейшего изучения и выполнения практических задач по составлению программ, все неясное прояснится и к сказанному можно будет вернуться для уточнения при выборе наилучшего способа адресации.

Система команд МП—К580 насчитывает более 240 команд. Многие из них повторяют свое мнемоническое название, отличаясь друг от друга только операндом. Например, команда MOV повторяется более 60 раз, указывая при помощи новых операндов разные места пересылки данных. Поэтому основных (ключевых) команд с непохожими мнемоническими наименованиями имеется около 70. Из этого числа более половины команд встречается при программировании очень редко. Таким образом, для начала достаточно запомнить 20—30 мнемонических названий операций, чтобы приступить к написанию простейших команд на машинном языке или языке ассемблера.

Все команды МП—К580 можно условно разделить на такие группы:

1) команды пересылок;
2) команды арифметических операций;
3) команды логических операций;
4) команды перехода;
5) специальные команды.

Перед описанием команд этих групп сделаем замечание относительно терминологии. В литературе по микропроцессорной технике наряду с выражением «Переслать данные в регистр из ячейки памяти» иногда употребляют более образное высказывание «Загрузить регистр содержимым ячейки памяти», понимая под словом «загрузить» — поместить, разместить пересылаемые данные, а под словом **«содержимое»** — то, что находится (содержится) в ячейке памяти, то есть ее данные. Такой терминологический разнобой часто затрудняет начинающему читателю понять основной смысл высказывания. Поэтому для облегчения чтения, на первых порах, будем приводить в скобках синонимы слов и выражений, обозначающих одно и то же действие.

Команды пересылок

Рассмотрим особенности команд группы пересылок, осуществляющих обмен данными между регистрами микропроцессора, ячейками памяти и устройствами ввода — вывода. Команды этого вида чаще других встречаются при составлении программ и поэтому на них следует обратить особое внимание.

Каждая пересылка данных всегда связана с тем местом, куда они пересылаются, а именно с приемником данных и тем местом, откуда эти данные поступают — источником данных. При написании операнда условились сначала указывать приемник (место, куда помещаются данные), а затем источник (место, откуда поступают данные).

Наименование приемника и источника разделяют запятой. Например, команда MOV B,C означает: переслать в регистр B (приемник), данные из регистра C (источник). После выполнения команды пересылки данные, находившиеся в приемнике, стираются и заменяются новыми, а данные, бывшие в источнике, остаются прежними, то есть в приемник пересылается как бы копия данных источника, а их подлинник остается в источнике неизменным.

Как отмечалось ранее, описание команд в колонке 2 таблицы 1 отличается чрезмерной краткостью и по содержанию не всегда доступно для начинающего

программиста. Поэтому далее приводится более подробное описание команд, выполненное в популярной форме.

Для удобства пользования описание каждой команды сопровождается условным обозначением, которое состоит из порядкового номера команды и мнемонического названия ее операции. Обозначение обрамлено прямоугольником. Например, 1. MOV означает: «Далее описана команда 1, согласно перечня колонки 1 табл. 1, имеющая название MOV (в соответствии с названием, указанным в колонке 3)». В конце описания каждой команды приводятся примеры, обозначенные двумя цифрами, разделенными точкой. Первая цифра указывает номер команды, а вторая сообщает порядковый номер примера. Таким образом, запись «Пример 3.2» означает: второй пример для команды 3 MOV.

Описание команд группы пересылок начнем с наиболее распространенной в этой группе команды MOV.

1. MOV

При обмене данными между внутренними регистрами A, B, C, D, E, H и L команда MOV повторяется 49 раз, указывая в своем операнде различные сочетания регистров (A,B; A,C; A,D; A,E; A,H; A,L; B,A; B,C; B,D и т. д.).

Чтобы не занимать в табл. 1 49 строк для размещения этих команд, поступают следующим образом: в табл. 1 приводят описание одной обобщенной команды межрегистровых пересылок MOV R1, R2, где R1 — регистр-приемник, а R2 — регистр-источник. В качестве R1 и R2 могут быть использованы любые из семи регистров (РОН) микропроцессора: A, B, C, D, E, H и L. Таким образом, все 49 команд MOV можно написать, подставляя вместо R1 и R2 буквенные обозначения регистров, получая команды: MOV A,A; MOV A,B; MOV A,C; MOV A,D; MOV A,E; MOV A,H; MOV A,L; MOV B,A; MOV B,B; MOV B,C и т. д.— всего 49 сочетаний.

Чтобы представить какую-либо из этих команд в двоичном коде, следует воспользоваться данными, приведенными в колонках 5 и 8 табл. 1. Выписав формат команды из колонки 5 и поместив в нужных местах байта команды код регистра-приемника R1 и код регистра-источника R2, взятые из таблицы колонки 8, получим запись команды в двоичном коде.

Выполним несколько примеров таких записей.

Пример 1.1.

Задание. Имеется команда MOV B,C («Переслать в регистр B содержимое регистра C»). Требуется найти двоичный и шестнадцатеричный коды команды.

Решение. Согласно сведениям колонки 5 табл. 1, имеем:

01	R1	R2

Подставляя вместо обозначений регистров R1 и R2 их действительные имена (буквы, присвоенные этим регистрам), напишем:

01	B	C

Из таблицы, приведенной в колонке 8, узнаем коды регистров B и C:

$$B = 000 \qquad C = 001$$

Вместо B и C записываем значение этих кодов:

01	000	001

Это и есть выражение команды MOV B,C в двоичном коде.

Условимся записывать такое выражение в виде набора, состоящего из восьми двоичных цифр (0 или 1) без обозначения системы счисления, а в скобках справа будем приводить мнемоническое название операций, условное обозначение операнда и краткие комментарии, отделенные друг от друга черточкой.

01000001 (MOV—B,C — пересылка в регистр B содержимого регистра C).

Такая запись по своей структуре напоминает форму записи программы на языке ассемблера, который изучается далее. Поэтому в конце выполнения каждого

примера будем приводить запись подобного вида, привыкая, таким образом, к языку ассемблера.

Зная двоичный код команды, легко получить и ее шестнадцатеричные выражения, которые часто применяются для более короткой записи программ, а также при вводе команд в память с помощью шестнадцатеричной клавиатуры, заменяющей переключатели.

Для перевода двоичного кода в шестнадцатеричный нужно разделить двоичную запись на тетрады:

0100 0001

и заменить каждую тетраду шестнадцатеричным эквивалентом. Так как $0100_2 = 4_{16}$, а $0001_2 = 1_{16}$, то выражение команды MOV B,C в шестнадцатеричном коде будет таким:

41 (MOV—B,C — пересылка в регистр B содержимого регистра C).

Здесь, как и в примере с двоичным кодом, обозначение шестнадцатеричной системы счисления (41_{16}) приводить не будем.

Пример 1.2.

Задание. Найти двоичный и шестнадцатеричный коды команды

MOV C, D

Решение. Формат этой команды (из колонки 5)

01	C	D

Подставив значение кода регистров C и D (из колонки 8), находим:

01	001	010

Таким образом, запись команды с указанием ее двоичного кода будет выглядеть так:

01001010 (MOV—C,D — пересылка в регистр C содержимого регистра D).

Эту же команду можно записать, указав шестнадцатеричный код. Так как $0100_2 = 4_{16}$ и $1010_2 = A_{16}$, то получаем:

4A (MOV—C,D — пересылка в регистр C содержимого регистра D).

Заметим, что выполнение команд типа MOV A,A; MOV B,B; MOV C,C и подобных им сводится к пересылке содержимого какого-либо регистра в этот же самый регистр. Собственно говоря, здесь никакой пересылки не происходит. Получается что-то вроде «переливания из пустого в порожнее», при котором никаких полезных действий не происходит. При этом выполняется своеобразная пустая операция, занимающая только некоторое время, и не приводящая ни к каким изменениям содержимого регистров. По своему действию эти команды (с одинаковыми операндами) соответствуют нулевой команде NOP, которая будет рассмотрена нами позже, в группе специальных команд:

2, 3 MOV

Рассмотрим команды обмена данными между регистрами микропроцессора и ячейкой памяти M.

Эти команды можно было бы написать по правилам, рассмотренным ранее для команд межрегистровых пересылок (MOV R1, R2), считая ячейку памяти некоторым регистром M, имеющим код M = 110 (из колонки 8).

В этом случае команду «Переслать в регистр R содержимое ячейки памяти M» можно было бы записать в виде: MOV R,M, а команду «Переслать в ячейку памяти M содержимое регистра R» как MOV M,R. Поступить так можно, но при этом следует учесть, что выполнение при помощи МП—К580 команд пересылок, у которых приемником или источником данных является ячейка памяти M, возможно только в том случае, если адрес ячейки памяти M находится в определенном, оговоренном месте (чаще всего в регистрах H и L). Если же адрес ячейки памяти еще не находится там, то нельзя ни писать, ни выполнять команду пересылки. Сперва обяза-

тельно следует разместить адрес ячейки памяти M в нужном месте. Поэтому следует запомнить, что перед тем, как писать или выполнять команды MOV R,M и MOV M,R, нужно поместить в регистровую пару HL адрес ячейки памяти M.

Примеры написания и выполнения команд MOV R,M и MOV M,R, а также способ пересылки адреса ячейки памяти M в регистровую пару HL будут приведены далее, при описании следующей команды.

4. MVI

Команда MVI R,K предназначена для размещения в каком-либо регистре РОН микропроцессора произвольного числа K. Эта команда отличается от команды межрегистровых пересылок (MOV R1, R2) тем, что данные поступают в регистр-приемник R1 не из регистра-источника R2, а из второго байта самой команды, куда предварительно заносят восьмиразрядное двоичное число K.

Мнемоническое обозначение команды MVI R,K говорит о том, что в ней участвует число K (на это указывает буква I в конце мнемонического слова).

Первый операнд R сообщает, что приемником данных может быть любой РОН микропроцессора (A, B, C, D, E, H или L), а второй операнд K называет число, которое мы хотим поместить в регистр R.

Из формата команды (колонка 5) видно, что она двухбайтовая. В первый байт команды (в его третий, четвертый и пятый разряды) помещают код нужного регистра R, взятый из колонки 8, а во второй байт записывают восьмиразрядное двоичное число. Если число K было указано в команде не в двоичном виде, то перед занесением во второй байт оно должно быть преобразовано в двоичную форму. Кроме того, если число, записанное в двоичной системе счисления, имеет меньше восьми разрядов, то нужно слева дописать недостающее количество нулей. Приведем примеры написания команд.

Пример 4.1.

Задание. Поместить в регистр C двоичное число 0101.

Р е ш е н и е. Воспользуемся командой MVI C,0101B.

Первый байт команды (из колонки 5) имеет вид

00	C	110

или, подставив код регистра C (из колонки 8),

00 001 110 1-й байт

Чтобы разместить число 0101_2 во втором байте, необходимо вначале превратить его из четырехразрядного в восьмиразрядное, дописав слева четыре нуля, а затем уже записать во второй байт:

0000 0101 2-й байт

Таким образом, команда MVI C, 0101B в двоичном коде будет выглядеть так:
00001110
00000101 (MVI—C,0101B — загрузка в регистр C числа 101_2).

Как видим, команда MVI двухбайтная. Чтобы ее выполнить, нужно ввести в память микроЭВМ два восьмиразрядных числа (байта), написанных слева.

Пример 4.2.

Задание. Записать в двоичном и шестнадцатеричном виде команду MVI B,2FH («Поместить в регистр B число 2F, представленное в шестнадцатеричной системе счисления»).

Р е ш е н и е. Первый байт команды имеет вид:

00	B	110

или, подставив код регистра B, получим:

00 000 110

1-й байт

Переводим шестнадцатеричное число 2F в двоичную форму:

$$2F_{16} = 0010\ 1111_2$$

Записываем переведенное двоичное число во второй байт команды:

0010 1111

2-й байт

Таким образом, команда MVI B, 2FH в двоичном коде будет выглядеть так:

00000110
00101111

и соответственно в шестнадцатеричном коде:

06
2F

Пример 4.3.

Задание. Поместить в регистр E десятичное число 15.

Решение. Используем команду MVI E,15D. Первый байт этой команды:

00	E	110

или, подставив код регистра E,

00	011	110

Переводим десятичное число 15 в двоичное:

$$15_{10} = 1111_2$$

Добавляя слева четыре нуля, преобразуем число в восьмиразрядный вид

0	0	0	0	1	1	1	1

Это и будет то число, которое нужно записать во второй байт команды. Таким образом, команда MVI E,15D может быть представлена в двоичном коде так:

00011110
00001111 (MVI—E,15D — загрузка в регистр E числа 15_{10})

и в шестнадцатеричном коде:

1E
0F (MVI—E,15D — загрузка в регистр E числа 15_{10})

При описании команд MOV R,M и MOV M,R упоминалось о необходимости предварительной пересылки в регистровую пару HL адреса ячейки памяти M. Такую пересылку можно выполнить при помощи различных команд, указанных в табл. 1, в том числе и с помощью команды MVI R,K.

Так как адрес ячейки памяти M является шестнадцатиразрядным двоичным числом, а команда MVI R,K позволяет пересылать в регистр R только восьмиразрядное число, то приходится применять команду дважды, пересылая каждый раз половину адреса в виде восьмиразрядного числа. При этом в регистр H засылают старший полуадрес (старшие разряды адреса), а в регистр L — младший полуадрес (младшие разряды адреса). Рассмотрим, как это делается.

Пример 4.4.

Задание. Разместить адрес $00AF_{16}$ ячейки памяти M в регистровой паре HL с помощью команды MVI R,K.

Решение. Адрес $00AF_{16}$ имеет 16 разрядов двоичного числа, а команда MVI R,K допускает пересылку в регистр R только восьмиразрядное двоичное число. Поэтому используем команду дважды, пересылая каждый раз половину (8 разрядов) шестнадцатиразрядного числа. Вначале пересылаем в регистр H старшие разряды адреса $00AF_{16}$:

MVI H,00H

Потом пересылаем в регистр L младшие разряды адреса $00AF_{16}$:

MVI L, AFH

Выполняя преобразование этих команд в порядке, указанном в примере 4.2, находим двоичный код первой команды (MVI H,00H)

00	H	110

или, подставив код регистра H,

00	100	110

Переводим шестнадцатиразрядное число старшего полуадреса 00_{16} в двоичную восьмиразрядную форму

$$00_{16} = 00000000_2$$

Это число необходимо записать во второй байт первой команды. Аналогично выполняем преобразование для второй команды (MVI L, AFH). Двоичный код первого байта второй команды:

00	101	110

Двоичный код второго байта второй команды:

$$AF_{16} = 10101111_2$$

Таким образом, двоичный код обеих команд будет выглядеть так:

00100110 (MVI—H,00H — загрузка старшего полуадреса ячейки памяти M
00000000 в регистр H).

00101110 (MVI—L,AFH — загрузка младшего полуадреса ячейки памяти M
10101111 в регистр L).

Если в память ввести четыре байта двоичных восьмиразрядных чисел, написанных слева, то МП—К580 выполнит обе команды, записанные в скобках справа, в результате чего адрес $00AF_{16}$ ячейки памяти M окажется загруженным в регистровую пару HL. Зная порядок пересылки адреса ячейки памяти M в регистровую пару HL, можно решить несколько примеров пересылки данных с помощью команды MOV R,M и команды MOV M,R.

Пример 2.1.

Задание. В ячейке памяти M, имеющей адрес $F0A1_{16}$, хранится десятичное число 132. Нужно переслать это число в регистр B.

Решение. Так как в пересылке участвует ячейка памяти, то прежде всего размещаем ее адрес в регистровой паре HL. Для этого используем два раза команду MVI R, K. Пересылаем в регистр H старшие разряды адреса, а в регистр L младшие:

MVI H, F0H
MVI L, AIH

Затем, пользуясь командой MOV R, M, пересылаем в регистр B содержимое ячейки памяти M:

MOV B, M

Таким образом, для пересылки числа 132_{10}, содержащегося в ячейке памяти M, нужно выполнить такие три команды:

MVI H, F0H — загрузка старшего полуадреса ячейки памяти M в регистр H;
MVI L, AIH — загрузка младшего полуадреса ячейки памяти M в регистр L;
MOV B, M — пересылка в регистр B содержимого ячейки памяти M.

Как видим, число 132_{10} не значится в операндах команд. Оно пересылается автоматически. Команды выполняют пересылку копии того, что содержится в ячейке памяти M. Так как там находится число 132_{10}, то оно и передается в регистр B.

Для выполнения задачи пересылки указанные выше команды нужно представить в двоичном коде и ввести в память микроЭВМ.

Пример 3.1.

Задание. Переслать в ячейку памяти М с адресом $0A0B_{16}$ содержимое регистра С.

Р е ш е н и е. Вначале заносим в регистровую пару HL адрес ячейки памяти М. Старший полуадрес ($0A_{16}$) помещаем в регистр Н —

MVI H, 0AH

Младший полуадрес ($0B_{16}$) помещаем в регистр L —

MVI L, 0BH

Пользуясь командой MOV M, R, пересылаем данные из регистра С в ячейку памяти М

MOV M, C

Таким образом, программа пересылки будет состоять из трех команд

```
MVI H, 0AH
MVI L, 0BH
MOV M, C
```

или, представив двоичный код каждой из них,

00100110
00001010 (MVI—H,0AH — загрузка старшего полуадреса $0A_{16}$ ячейки памяти М в регистр Н);

00101110 (MVI—L, 0BH — загрузка младшего полуадреса $0B_{16}$ ячейки памяти М в регистр L);
00001011

01110001 (MOV—M, C — пересылка в ячейку памяти М содержимого регистра С).

Пример 3.2.

Задание. Написать перечень команд для загрузки (размещения) числа 1010_2 в регистр В и его последующей пересылки из регистра В в ячейку памяти М с адресом 0102_{16}.

Р е ш е н и е. Используя команду MVI R, K, засылаем число 1010_2 в регистр В:

MVI B, 1010B

С помощью этой же команды размещаем адрес ячейки памяти М в регистрах Н и L:

```
MVI H, 01H
MVI L, 02H
```

Используя команду MOV M, R, пересылаем в ячейку памяти М содержимое регистра В:

MOV M, R

Таким образом, перечень команд, выполняющих поставленную задачу, запишется так:

```
MVI B,1010 B
MVI H,01 H
MVI L,02 H
MOV M, B
```

Чтобы выполнить эти команды, их необходимо представить в двоичном коде и ввести в память микроЭВМ.

5. MVI

Команда MVI M, K является разновидностью команд MVI и позволяет загружать ячейку памяти М двоичным восьмиразрядным числом К. Таким образом,

в отличие от команды MVI R, K, загружающей число K в регистр R, здесь команда по тем же правилам загружает число K в ячейку памяти M. Следует только помнить, что перед выполнением этой команды адрес ячейки памяти M должен находиться в регистровой паре HL.

Пример 5.1.

Задание. Разместить в ячейке памяти M с адресом $OFFO_{16}$ десятичное число 17.

Решение. Сперва засылаем в регистровую пару HL адрес $OFFO_{16}$ ячейки памяти M (см. пример 3.1):

MVI H, 0FH
MVI L, F0H

Затем размещаем число 17_{10} в ячейке памяти M:

MVI M,17 D

Пользуясь известными правилами (см. пример 4.4), представляем эти три команды в двоичном коде:

00100110
00001111 (MVI—H, OFH — загрузка старшего полуадреса $0F_{16}$ ячейки памяти M в регистр H);

00101110
11110000 (MVI—L, FOH — загрузка младшего полуадреса $F0_{16}$ ячейки памяти M в регистр L);

00110110
00010001 (MVI—M, 17 D — размещение в ячейке памяти M числа 17_{10}).

6. LXI

Команда LXI P, K, в отличие от команды MVI R, K, позволяет загружать шестнадцатиразрядное, а не восьмиразрядное число K, причем в регистровую пару P, а не в регистр R. Так как в мнемоническом обозначении команды LXI содержится символ I, то это указывает на то, что команда принадлежит к группе пересылок с участием числа K.

Операнд P называет регистровую пару (BC, DE или HL), в которую загружается число K. Для сокращения записи условились регистровую пару P обозначать здесь при помощи ее первого символа, т. е. пару BC именуют как B, пару DE — как D и пару HL — как H.

Команда LXI P, K — трехбайтовая.

В первый байт команды (в его четвертый и пятый разряды) помещают код регистровой пары P, взятый из колонки 8, а во второй и третий байты записывают по восемь разрядов шестнадцатиразрядного числа. При этом во второй байт записывают младшие разряды числа, а в третий — старшие.

Приведем пример написания команды LXI.

Пример 6.1.

Задание. Используя команду LXI P, K, переслать в регистровую пару HL адрес $FFOO_{16}$ ячейки памяти M.

Решение. Согласно условию задания команда LXI запишется так:

LXI H, FF 00H

Представим ее в двоичном коде. Первый байт (согласно колонке 5)

00	H	0001

или, подставив значение кода регистровой пары H (из колонки 8),

00	10	0001

1-й байт

Преобразовываем адрес $FF00_{16}$ ячейки памяти M в двоичную форму:

$$FF00_{16} = 1111111100000000_2$$

Записываем младшие разряды во второй байт:

00000000	2-й байт

а старшие разряды — в третий байт:

11111111	3-й байт

Окончательно получаем:

00100001
00000000 (LXI—H, FF00H — загрузка в регистровую пару HL адреса $FF00_{16}$
11111111 ячейки памяти M).

Как видим, здесь пересылка адреса ячейки памяти M в регистровую пару HL выполняется за три байта вместо четырех, нужных для решения этой же задачи при помощи команд MVI (см. пример 4.4).

7. LDA

С помощью команды LDA ADR можно загружать аккумулятор (регистр A) содержимым ячейки памяти M, т. е. пересылать в аккумулятор копию данных, находящихся в ячейке памяти M. При этом адрес ячейки памяти M нужно записать во второй и в третий байт команды. Колонка 5 табл. 1 указывает на то, что команда LDA ADR трехбайтовая. Первый байт предназначен для кода операции. Второй байт служит для записи восьми младших разрядов адреса ячейки памяти M, а третий байт — для записи восьми старших разрядов этого же адреса.

Пример 7.1.

Задание. Содержимое ячейки памяти M с адресом $2FFF_{16}$ переслать в аккумулятор.

Р е ш е н и е. Пересылку выполняем при помощи команды LDA ADR. Согласно заданию, эта команда запишется так:

LDA 2FFFH

Первый байт команды (из колонки 5):

00111010

Второй байт команды (младший полуадрес ячейки памяти M):

$$FF_{16} = 11111111_2$$

Третий байт команды (старший полуадрес ячейки памяти M):

$$2F_{16} = 00101111_2$$

Таким образом, команда в двоичном коде будет выглядеть так:

00111010
11111111 (LDA—2FFFH — пересылка в аккумулятор содержимого ячейки
00101111 памяти M с адресом $2FFF_{16}$).

8. STA

Команда STA ADR, в отличие от команды LDA ADR, служит для пересылки содержимого аккумулятора в ячейку памяти M. При этом адрес ячейки памяти M так же, как и в команде LDA, нужно занести во второй и в третий байты команды.

Команда STA ADR трехбайтовая. В первый байт помещают код операции. Во второй байт записывают младшие разряды адреса ячейки памяти M, а в третий — старшие разряды.

Пример 8.1.

Задание. Переслать в ячейку памяти M, имеющую адрес 1100_{16}, содержимое аккумулятора.

Р е ш е н и е. Для пересылки воспользуемся командой

STA ADR

или, согласно заданию

STA 1100H

Первый байт команды (из колонки 5)

00110010

Во второй байт записываем младший полуадрес 00_{16} ячейки памяти M

$$00_{16} = 00000000_2$$

а в третий — старший полуадрес 11_{16}

$$11_{16} = 00010001_2$$

Таким образом, команда запишется в двоичном коде так:

00110010
00000000
00010001

(STA—1100H — пересылка в ячейку памяти M с адресом 1100_{16} содержимого аккумулятора).

9. LDAX	10. STAX

Команды LDAX P и STAX P выполняют те же операции, что и команды LDA ADR и STA ADR, то есть пересылают содержимое ячейки памяти M в аккумулятор (LDAX P) и обратно — содержимое аккумулятора в ячейку памяти M (STAX P). Отличие этой пары команд в том, что при выполнении команд LDA ADR и STA ADR адрес ячейки памяти размещается во 2-м и 3-м байтах самой команды, а при выполнении команд LDAX P и STAX P адрес ячейки памяти M необходимо предварительно разместить в регистровой паре P. Причем регистровой парой для размещения адреса может быть только пара BC или DE. В регистровой паре HL размещать такой адрес нельзя.

Команды LDAX P и STAX P однобайтовые, но если учесть, что перед их выполнением необходимо производить многобайтовые операции по размещению адреса ячейки памяти M в регистровой паре P, то в итоге пересылка занимает значительное количество байтов. Команды LDAX P и STAX P обычно используют в тех случаях, когда по условию выполнения программы адрес ячейки памяти M по тем или иным причинам уже находится в регистровой паре BC или DE.

Пример 9.1.

Задание. Переслать в аккумулятор содержимое ячейки памяти M, имеющей адрес F0 FA_{16}, используя команду LDAX P.

Р е ш е н и е. Если адрес ячейки памяти M уже находится в регистровой паре BC, то пересылка содержимого ячейки памяти в аккумулятор может быть выполнена с помощью простой однобайтовой команды:

LDAX B

двоичный код которой (из колонки 5) равен:

00001010

В то же время, если в регистровой паре BC адрес ячейки памяти M еще не содержится, то его следует туда переслать при помощи команды LXI B.
Первый байт этой команды (из колонки 5):

00000001

Второй байт (младший полуадрес ячейки памяти M):

$$FA_{16} = 11111010_2$$

Третий байт (старший полуадрес ячейки памяти M):

$$F0_{16} = 11110000_2$$

Таким образом, пересылка в аккумулятор содержимого ячейки памяти M может быть осуществлена с помощью ввода в память микроЭВМ следующих четырех байтов команд:

00000001
11111010 (LXI—B — пересылка адреса ячейки памяти M в регистровую пару BC).

11110000
00001010 (LDAX—B — пересылка в аккумулятор содержимого ячейки памяти M).

Эту же задачу можно решить и с помощью других команд пересылки, используя команду MOV A, M и предварительно поместив адрес ячейки памяти M в регистровую пару HL.

Рассмотрим еще один пример использования различных команд из группы пересылок.

Пример 9.2.

Задание. Переслать содержимое ячейки памяти M с адресом $00A0_{16}$ в рядом расположенную (старшую по адресу) ячейку.

Решение. Эту задачу можно решить различными путями.

Вариант 1

Так как в операции пересылки участвует ячейка памяти M, то размещаем ее адрес в регистровой паре HL с помощью команды MVI:

```
MVI H, 00H
MVI L, A0H
```

Пересылаем содержимое ячейки памяти M в аккумулятор:

```
MOV A, M
```

Размещаем в регистровой паре HL новый адрес $00A1_{16}$ соседней ячейки памяти:

```
MVI H, 00H
MVI L, AIH
```

Затем пересылаем в эту соседнюю ячейку памяти содержимое аккумулятора:

```
MOV M,A
```

Таким образом, задача пересылки решается при помощи следующих команд:

```
MVI H, 00H
MVI L, A0H
MOV A, M
MVI H, 00H
MVI L, AIH
MOV M, A
```

Если учесть, что команда MVI двухбайтовая, а MOV однобайтовая, то пересылка данных занимает 10 байт.

Вариант 2.

Используем команду LXI P, K.

Засылаем адрес начальной ячейки памяти M в регистровую пару HL

```
LXI H, 00 A0H
```

Пересылаем содержимое ячейки памяти M в аккумулятор:

```
MOV A, M
```

Размещаем в регистровой паре HL новый адрес $00A1_{16}$ соседней ячейки памяти:

```
LXI H, 00 A1H
```

Посылаем содержимое аккумулятора в соседнюю ячейку памяти:

```
MOV M, A
```

Таким образом, пересылка осуществляется при помощи следующих команд:

```
LXI H, 00 A0H
MOV A, M
LXI H, 00 AIH
MOV M, A
```

Если учесть, что команда LXI трехбайтовая, а MOV однобайтовая, то пересылка занимает 8 байт (при варианте 1 — 10 байт).

Вариант 3.

Для решения используем пару команд LDA ADR и STA ADR. При помощи команды LDA ADR в аккумулятор пересылаем содержимое ячейки памяти M:

```
LDA 00A0H
```

Затем по команде

```
STA 00A1H
```

пересылаем содержимое аккумулятора в соседнюю ячейку памяти, имеющую адрес на единицу больше.

Как видим, та же задача пересылки решается здесь при помощи двух команд:

```
LDA 00A0H
STA 00AIH
```

Если учесть, что каждая из них занимает три байта, то для пересылки содержимого из ячейки памяти M в соседнюю ячейку нужно выполнить 6 байт (при варианте 1—10 байт, а при варианте 2—8 байт).

Таким образом, вариант 3 наиболее рациональный, так как он занимает наименьший объем памяти и быстрее выполняет задачу пересылки.

Прежде чем перейти к рассмотрению команд обмена данными с устройствами ввода—вывода, напомним, что к шине данных их подсоединяют через специальные схемы, включающие эти устройства по сигналам микропроцессора во время обмена данными. Такие включающие схемы получили название **порт.** Поэтому в литературе часто пишут «порт ввода—вывода», вместо «устройство ввода—вывода», имея в виду, что порт является главной составной частью схемы при обмене данными. И еще одно замечание.

МП—К580 устроен так, что обмен данными между устройствами ввода— вывода и микропроцессором осуществляется только через аккумулятор, причем данные поступают из порта ввода сначала в аккумулятор, а затем уже с помощью соответствующих команд могут быть пересланы в нужный регистр микропроцессора или ячейку памяти. Аналогично, при вводе данных из регистра микропроцессора или ячейки памяти они предварительно должны быть помещены в аккумулятор, а затем уже при помощи команды вывода могут быть переданы из аккумулятора в нужный порт вывода.

11. IN

Команда IN N служит для ввода данных в аккумулятор из устройства ввода, имеющего номер N. Часто номер, по аналогии с номером ячейки памяти M, именуют адресом N. В этой команде приемником данных всегда является аккумулятор, поэтому он и не указывается в операнде.

Команда IN N двухбайтовая. Первый байт указывает код операции, а второй — номер (адрес) устройства ввода или его порта.

Пример 11.1.

Задание. Переслать данные из порта ввода с адресом 04_{16} в регистр B.

Решение. Пользуясь командой IN N, пересылаем данные из порта ввода в аккумулятор:

```
IN 04H
```

а затем содержимое аккумулятора пересылаем в регистр B:

MOV B, A

Таким образом, пересылка выполняется при помощи команд:

11011011
00000100 (IN—04H — пересылка в аккумулятор данных из порта ввода);

01000111 (MOV—B, A — пересылка в регистр B содержимого аккумулятора)

Пример 11.2.

Задание. Переслать данные из порта ввода с адресом $F1_{16}$ в ячейку памяти M, имеющую адрес $F010_{16}$

Р е ш е н и е. Пересылаем данные из порта ввода в аккумулятор:

IN F1H

Так как в пересылке данных участвует ячейка памяти M, то размещаем ее адрес в регистровой паре HL

LXI H, F010H

Затем пересылаем содержимое аккумулятора в ячейку памяти M

MOV M, A

Таким образом, задача пересылки сводится к выполнению команд:

IN F1H
LXI H, F010H
MOV M, A

В двоичном коде эти команды будут выглядеть так:

11011011
11110001 (IN—F1H — пересылка данных в аккумулятор из порта ввода);

00100001
00010000
11110000 (LXI—H, F010H — пересылка в регистровую пару HL адреса ячейки памяти M);

01110111 (MOV M, A — пересылка в ячейку памяти M содержимого аккумулятора).

12. OUT

Команда OUT N служит для вывода данных из аккумулятора в порт вывода с адресом N. В этой команде источником данных всегда является аккумулятор, поэтому он и не указывается в операнде.

Команда OUT N двухбайтовая. Первый байт отводится для кода операции, а второй — для записи адреса порта вывода.

Ниже приводятся примеры записи команды OUT N.

Пример 12.1.

Задание. Переслать содержимое регистра C в порт вывода с адресом $0B_{16}$.

Р е ш е н и е. Содержимое регистра C пересылаем в аккумулятор:

MOV A, C

а затем содержимое аккумулятора пересылаем в порт вывода:

OUT 0BH

Эти же команды в двоичном коде имеют вид:

01111001 (MOV—A, C — пересылка в аккумулятор содержимого регистра C)

11010011
00001011 (OUT— 0BH — пересылка в порт вывода $0B_{16}$ содержимого аккумулятора).

Пример 12.2.

Задание. Переслать содержимое ячейки памяти М, имеющей адрес $ABBA_{16}$, в порт вывода с адресом $1A_{16}$.

Р е ш е н и е. Адрес ABBAH ячейки памяти М пересылаем в регистровую пару HL:

LXI H, ABBAH

Содержимое ячейки памяти М пересылаем в аккумулятор:

MOV A,M

Содержимое аккумулятора пересылаем в порт вывода с адресом $1A_{16}$:

OUT 1AH

Представляя эти команды в двоичном коде, получаем:

00100001
10111010 (LXI—H, ABBAH — пересылка в регистровую пару HL адреса ячейки памяти М).

10101011
01111110 (MOV—A, M — пересылка в аккумулятор содержимого ячейки памяти М).

11010011
00011010 (OUT—1AH — пересылка в порт вывода $1A_{16}$ содержимого аккумулятора).

13.15 PUSH	17. XCHG	19. SPHL	
14.16 POP	18. XTHL	20. LHLD	21. SHLD

Команды 13—21 используются для пересылки данных при выполнении стековых операций. При выполнении программы регистры РОН всегда бывают загружены данными промежуточных вычислений и преобразований. Когда возникает необходимость перехода к выполнению другой программы (подпрограммы), то прежде всего приходится очищать регистры РОН для этой новой программы, а также решать вопрос о том, как сохранить данные регистров прежней программы, чтобы, вернувшись к ее выполнению, можно было восстановить первоначальное состояние регистров.

Перед началом выполнения новой программы по команде PUSH содержимое регистров РОН пересылается в ячейки стековой памяти, а после завершения программы по команде POP — происходит возврат содержимого из ячеек стековой памяти в те самые регистры, из которых содержимое было переслано в стек. Процесс пересылки и возвращения данных довольно громоздкий и сложный. Для его упрощения применяется ряд дополнительных команд XCHG, XTHL, SPHL, LHLD, SHLD, ускоряющих и облегчающих операции пересылки. Кроме того, напомним, что МП—К580 имеет специальный шестнадцатиразрядный регистр УС, называемый **указателем стека** и предназначенный для указания адреса ячеек стековой памяти. Этот регистр иногда обозначается символами SP.

Для полноты изложения в табл. 1 приведены все команды стековых операций, однако в тексте нашего изложения мы не будем рассматривать их все подробно. Наша задача — ознакомить с языком ассемблера и его простейшими командами. С помощью дополнительной литературы [1, 14, 38, 58, 62] можно будет освоить все операции стековых пересылок и включить их в лексикон языка ассемблера.

Команды арифметических операций

Как было указано выше, все команды арифметических операций выполняются только через аккумулятор. При этом один из операндов должен находиться в аккумуляторе еще до выполнения команды. Таким образом, складывая два числа, необходимо одно из них поместить в аккумулятор, а второе можно загрузить в регистр R, в ячейку памяти М или записать во второй байт команды. Только после этого следует выполнять команду сложения.

После выполнения команды арифметической операции результат вычисления автоматически размещается в аккумуляторе, а число, которое там было, стирается.

МП—К580 выполняет лишь команды сложения и вычитания, а операции умножения и деления осуществляются по особым подпрограммам с помощью набора команд сложения, сдвига и других преобразований в соответствии с теми правилами, которые были рассмотрены нами в разделе арифметики двоичного счисления.

К группе команд арифметических операций относятся также команды, уменьшающие или увеличивающие на единицу содержимое регистров РОН или ячеек памяти.

Помимо простейших операций по сложению и вычитанию двоичных чисел, результат которых не превышает восьми разрядов (десятичное число 255), МП—К580 может выполнять ряд арифметических вычислений с числами произвольной длины, с отрицательными и дробными числами. Подобные действия осуществляются при помощи команд, учитывающих признаки переноса и займа, образующиеся в специальном регистре признаков F. Эти операции довольно громоздкие, доступные лишь специалисту и поэтому, как и сложные операции стековых пересылок, рассматриваться нами не будут. Далее описаны только простейшие арифметические команды.

22 , 23. ADD

Команда ADD R служит для сложения числа, содержащегося в аккумуляторе, с числом, размещенным в регистре R. Сумма, которая образуется после выполнения этой команды, помещается в аккумуляторе, а число, находившееся там, стирается.

Команда ADD R однобайтовая. Пять старших разрядов этого байта (см. колонку 5 табл. 1) заняты кодом операции, а три младших — кодом регистра R. Как и в командах пересылок, регистром может быть любой регистр РОН. Значение кода R приведено в колонке 8 табл. 1 для команды 1.

Пример 22.1.

Задание. Написать команду сложения двух чисел, находящихся в регистре B и в аккумуляторе. Определить двоичный и шестнадцатеричный код такой команды.

Р е ш е н и е. Используем команду ADD R, которая запишется для нашего случая так:

ADD B

Согласно сведениям колонки 5, формат этой команды имеет вид:

10000	R

или, подставив значения кода регистра B:

10000000

Это и есть двоичная форма записи команды. Ее шестнадцатеричное представление будет таким:

$$10000000_2 = 80_{16}$$

Поэтому окончательно:

80 (ADD—B — сложение содержимого аккумулятора и регистра B).

Пример 22.2.

Задание. Используя регистры РОН, сложить два десятичных числа 13 и 7. Результат сложения поместить в ячейке памяти M с адресом $F1F0_{16}$.

Р е ш е н и е. Применяя команду MVI R, K, размещаем числа 13 и 7 в регистрах A и B:

MVI A, 13D
MVI B, 7D

По команде ADD R складываем содержимое регистров A и B:

ADD B

После выполнения этой команды сумма окажется размещенной в аккумуляторе.

Чтобы переслать содержимое аккумулятора в ячейку памяти М, воспользуемся командой STA ADR:

STA F1F0H

Таким образом, программа решения будет состоять из таких команд:
MVI A, 13D — размещение в аккумуляторе числа 13.
MVI B, 7D — размещение в регистре B числа 7.
ADD B — сложение содержимого аккумулятора и регистра B.
STA F1F0H — пересылка в ячейку памяти с адресом $F1F0_{16}$ содержимого аккумулятора.

Если представить эти команды в двоичном коде и ввести их в память микроЭВМ, то после выполнения команд в ячейке памяти М окажется число, равное двадцати $(13 + 7 = 20)$.

Команда ADD M является разновидностью команды сложения, отличающейся от команды ADD R тем, что число, содержащееся в аккумуляторе, складывается с числом, находящимся в ячейке памяти М. При выполнении этой команды необходимо вместо кода R подставить код ячейки памяти M = 110, а также не забыть разместить адрес ячейки памяти М в регистровой паре HL.

Пример 23.1.

Задание. Написать команду сложения содержимого аккумулятора с числом, находящимся в ячейке памяти М, адрес которой $00BA_{16}$. Определить двоичный код команды.

Р е ш е н и е. Так как в операции сложения участвует ячейка памяти М, то раньше всего размещаем ее адрес в регистровой паре HL.

Это размещение можно выполнить при помощи различных команд, однако, лучше всего использовать команду LXI P, K. Поэтому запишем:

LXI H, 00BAH

Теперь складываем содержимое аккумулятора с числом, находящимся в ячейке памяти М, при помощи команды ADD M:

ADD M

Находим двоичный код команды LXI H, 00BAH — первый байт команды (см. пример 6.1):

00100001_2

второй байт (младший полуадрес BA_{16} ячейки памяти М):

10111010_2

третий байт (старший полуадрес 00_{16} ячейки памяти М):

00000000_2

Двоичный код команды ADD M (из колонки 5):

10000110

Таким образом, окончательно получим:

00100001
10111010
00000000 (LXI— 00BAH — загрузка адреса $00BA_{16}$ ячейки памяти М в регистровую пару HL);

10000110 (ADD—M — сложение содержимого аккумулятора и ячейки памяти М).

24, 25. ADC	30, 31. SBB
27. ACI	33. SBI

Команды 24, 25, 27, 30, 31, 33 выполняют операции арифметического сложения и вычитания с учетом признаков переноса и займа, которые образуются после выполнения команды в виде изменения состояния триггеров, входящих в состав регистра F.

Признаки переноса и займа используются при сложных вычислениях дробных и отрицательных чисел, а также для выполнения действий повышенной точности с числами произвольной длины. Чтобы разобраться с такими операциями, нужна достаточная подготовка, поэтому отсылаем читателей к литературе [1, 14, 58, 62].

26. ADI

Команда ADI K позволяет сложить число K с числом, находящимся в аккумуляторе. Для этого во второй байт команды нужно записать двоичное выражение числа K. После выполнения команды сумма вычислений остается в аккумуляторе.

Пример 26.1.

Задание. Прибавить к содержимому аккумулятора десятичное число 16.

Р е ш е н и е. Пользуясь командой ADI K, записываем:

ADI 16D

первый байт команды (из колонки 5):

11000110

Так как $16_{10} = 10000_2$, то второй байт команды:

00010000

и вся команда запишется так:

11000110
00010000 (ADI—16D — сложение содержимого аккумулятора и числа 16_{10}).

28, 29. SUB

32. SUI

Команды 28, 29 и 32 выполняют операции вычитания. По своей структуре и особенностям действий они аналогичны командам сложения 22, 23 и 26.

Все, что было сказано о командах сложения, действительно и для команд вычитания. Таким образом, по команде SUB R выполняется вычитание из содержимого аккумулятора числа, расположенного в регистре R, по команде SUBM выполняется вычитание числа, расположенного в ячейке памяти M, а по команде SUI K — вычитание числа K, записанного во втором байте команды.

Единственное отличие команд вычитания от команд сложения состоит в том, что в аккумуляторе должно находиться уменьшаемое число. В то же время в регистре R, в ячейке памяти M или во втором байте команды должно размещаться вычитаемое.

Кроме того, нужно помнить, что все операции вычитания выполняются микропроцессором, как операции сложения уменьшаемого с вычитаемым, представленным в дополнительном коде.

Пример 28.1.

Задание. Написать команду, позволяющую вычесть из содержимого аккумулятора число, находящееся в регистре C. Представить команду в двоичной форме.

Р е ш е н и е. Пользуясь командой SUB R, записываем ее согласно заданию:

SUB C

формат такой команды:

10010	R

Подставляя значение кода R для регистра C, получаем:

10010	001

Это позволяет записать команду в двоичной форме:

10010001 (SUB—C — вычитание из содержимого аккумулятора содержимого регистра C).

Пример 29.1.

Задание. Вычесть из числа, находящегося в аккумуляторе, число, расположенное в ячейке памяти M с адресом 1001_{16}.

Р е ш е н и е. Зная о том, что команда с участием ячейки памяти M требует предварительного размещения адреса ячейки памяти M в регистровой паре HL, применяем команду LXI P, K для такого размещения:

LXI H, 1001H

Теперь можно вычесть содержимое ячейки памяти M из содержимого аккумулятора, если воспользоваться командой:

SUB M

Таким образом, согласно принятой форме записи будем иметь:

00100001
00000001 (LXI—H, 1001H — загрузка адреса 1001_{16} ячейки памяти M в регистры H и L).

00010000
10010110 (SUB—M — вычитание из содержимого аккумулятора содержимого ячейки памяти M).

Пример 30.1.

Задание. Сложить содержимое регистров B и C, а затем вычесть из полученной суммы десятичное число 5. Результат вычислений переслать в порт вывода с адресом $0A_{16}$

Р е ш е н и е. Учитывая, что операция сложения выполняется только через аккумулятор, пересылаем содержимое одного из регистров (регистра B) в аккумулятор:

MOV A,B

Затем складываем содержимое аккумулятора и регистра C:

ADD C

Результат сложения оказывается размещенным в аккумуляторе. Чтобы вычесть из содержимого аккумулятора число 5_{10}, применяем команду:

SUI 5D

После выполнения этой команды содержимое аккумулятора уменьшится на 5_{10}. Для пересылки нового значения содержимого аккумулятора в порт вывода с номером (адресом) $A0_{16}$ нужно воспользоваться командой:

OUT 0AH

Таким образом, программа решения поставленной задачи запишется так (без обозначения двоичного кода):

MOV A, B — пересылка в аккумулятор содержимого регистра B.

ADD C — сложение содержимого аккумулятора и регистра C.

SUI 5D — вычитание из содержимого аккумулятора числа 5_{10}.

OUT OAH — пересылка в порт вывода $A0_{16}$ содержимого аккумулятора.

34, 35. INR	37, 38. DCR

Команда INR R позволяет увеличить на единицу содержимое регистра R, а команда INR M — увеличить на единицу содержимое ячейки памяти M. Аналогично, команда DCR R позволяет уменьшить на единицу содержимое регистра R, а команда DCR M — уменьшить содержимое ячейки памяти M.

Команды INR R и DCR R однобайтовые.

Третий, четвертый и пятый разряды байта команды заняты под код регистра R или ячейки памяти M, а остальные разряды отведены для кода операции.

Перед выполнением команд INR M и DCR M, в которых участвует ячейка памяти M, адрес ячейки памяти M нужно разместить в регистровой паре HL.

В литературе операцию увеличения числа на единицу называют инкрементированием, а операцию уменьшения — декрементированием.

При составлении программ команды INR и DCR чаще всего используют в сочетании с командами логических операций и переходов. Примеры написания и применения команд INR и DCR приведены далее.

36. INX

39. DCX

Команды INX P и DCX P являются разновидностью команд инкрементирования и декрементирования применительно к регистровым парам P.

Команда INX P позволяет увеличить на единицу содержимое регистровой пары P, а команда DCX P — уменьшить на единицу содержимое регистровой пары P.

Команды INX P и DCX P однобайтовые. В четвертом и пятом разрядах байта команды помещен код регистровой пары P, а в остальных разрядах — код операции.

40. DAD

Команда DAD P позволяет складывать два шестнадцатиразрядные числа. Одно из этих чисел помещают в регистровую пару HL, а другое — в регистровую пару P (BC, DE или SP). Результат сложения размещается в регистровой паре HL.

Команда DAD P однобайтовая.

Пример 40.1.

Задание. Используя команду DAD P, сложить два десятичных числа: 1263 и 284.

Р е ш е н и е. С помощью команды LXI P первое слагаемое (1263_{10}) размещаем в регистровой паре HL:

LXI H, 1263D

а второе (284_{10}) — в регистровой паре BC:

LXI B, 284D

После этого по команде DAD P складываем содержимое регистровых пар HL и BC:

DAD B

Выполнив необходимые преобразования, получаем следующую запись команд в принятой нами форме:

00100001
11101111 (LXI—H, 1263D — размещение числа 1263_{10} в регистровой паре HL).
00000100

00000001
00011100 (LXI—B, 284D — размещение числа 284_{10} в регистровой паре BC).
00000001

00001001 (DAD—B — сложение содержимого регистровых пар HL и BC).

41. DAA

Команда DAA служит для десятичной коррекции при выполнении особого способа двоично-десятичного сложения. Этот необычный вид сложения применяется довольно редко и поэтому его не рассматриваем.

Команды логических операций

Команды логических операций во многом похожи на команды арифметических операций. Они также оперируют с двумя восьмиразрядными двоичными числами, одно из которых должно находиться в аккумуляторе, а второе — в регистре R, в ячейке памяти M или во втором байте команды. Результат операции размещается в аккумуляторе.

МП—К580 выполняет следующие логические операции: И, ИЛИ, НЕ и ИСКЛЮЧАЮЩЕЕ ИЛИ.

К группе логических команд условились относить также операции сравнения двух чисел и сдвига числа, находящегося в аккумуляторе.

Логические действия выполняются над числами поразрядно, то есть нулевой разряд одного числа логически сравнивается с нулевым разрядом другого числа, и результат сравнения располагается в нулевом разряде аккумулятора. Затем такое же сравнение производится над первыми разрядами чисел, потом над вторыми и т. д. до последних седьмых разрядов.

42, 43. ANA	44. ANI

Команды ANA R и ANI K выполняют логическое умножение (операцию И) над двумя восьмиразрядными двоичными числами, одно из которых находится в аккумуляторе, а второе — в регистре R (команда ANA R) или во втором байте команды (команда ANI K).

Напомним, что операция И выполняется согласно такому правилу: логическое И образует единицу тогда и только тогда, когда оба числа равны единице. Если одно из чисел или оба равны нулю, то и результат будет нулевым.

Приведем пример выполнения этого правила:

1-е число (находится в аккумуляторе):	10010011
2-е число (находится в регистре R, в ячейке памяти M или во 2-м байте команды):	11001110
Результат операции И (размещается в аккумуляторе)	10000010

Если результат этой записи покажется не достаточно ясным, следует еще раз просмотреть раздел алгебры логики, уделяя особое внимание таблицам истинности логических операций И, ИЛИ и НЕ.

Команды ANA R и ANA M однобайтовые. Они состоят из кода регистра R, размещенного в младших разрядах байта команды и кода операции, расположенного в старших разрядах. Перед выполнением команды ANA M в регистровую пару HL нужно загрузить адрес ячейки памяти M.

Команда ANI K двухбайтовая. В первый байт занесен код операции, а во второй — двоичное число K, которое подвергают операции И (логическому умножению) совместно с числом, находящимся в аккумуляторе.

Результат операции И во всех трех командах (42, 43 и 44) размещается в аккумуляторе, а бывшее там число стирается.

При выполнении каждой логической операции триггер C признака переноса, входящий в состав регистра F, устанавливается в нулевое значение. Этим часто пользуются для решения практических задач, возникающих в процессе написания программы. Более подробно о действии триггеров регистра признаков F рассказано при описании команд переходов.

Пример 42.1.

Задание. Выполнить операцию И над двоичными числами 0110 и 1001. Результат операции переслать в порт вывода с адресом FF_{16}.

Р е ш е н и е. Используя команду MVI R, K, помещаем первое число (0110_2) в регистр A, а второе (1001_2) в регистр B:

```
MVI A, 0110B
MVI B, 1001B
```

По команде ANA R выполняем операцию И над содержимым регистров A и B.

ANA B

Применяя команду OUT N, результат операции И пересылаем из аккумулятора в порт вывода с адресом

OUT FFH

Таким образом, программа выполнения задачи запишется так (значение двоичного кода команд опускаем):

MVI A, 0110B — размещение числа 0110_2 в аккумуляторе A.
MVI B, 1001 B — размещение числа 1001_2 в регистре B.
ANA B — операция И над содержимым аккумулятора и регистра B.
OUT FFH — пересылка содержимого аккумулятора в порт вывода FF_{16}.

Пример 44.1.

Задание. Произвести логическое умножение (операцию И) числа 111_2 и числа, расположенного в ячейке памяти M, имеющий адрес $01AB_{16}$.

Р е ш е н и е. Учитывая участие в операции ячейки памяти M, размещаем ее адрес в регистровой паре HL:

LXI H, 01ABH

Используем команду MOV A, M для пересылки содержимого ячейки памяти M в аккумулятор A:

MOV A, M

по команде ANI K выполняем логическое умножение числа 111_2 и содержимого аккумулятора A:

ANI 111B

Таким образом, для выполнения задания нужно реализовать такую программу из трех команд:

LXI—H, 01ABH — размещение адреса ячейки памяти M в регистровой паре HL.
MOV—A,M — пересылка в аккумулятор содержимого ячейки памяти M.
ANI—111B — операция И над содержимым аккумулятора и числом 111_2.

45, 46. ORA

47. ORI

Эти три команды выполняют логическое сложение (операцию ИЛИ). По своей структуре и порядку выполнения они полностью соответствуют командам логического умножения (операции И). А именно: команда ORA R выполняет операцию ИЛИ над числами, расположенными в аккумуляторе и регистре R, команда ORA M выполняет такую же операцию над числами, расположенными в аккумуляторе и ячейке памяти M, а команда ORI K выполняет операцию ИЛИ над содержимым аккумулятора и числом K, записанным во втором байте команды.

Как уже было сказано, операция ИЛИ выполняется по такому правилу: логическое ИЛИ образует нуль тогда и только тогда, когда оба числа равны нулю. Если одно из чисел или оба равны единице, то и результат будет равен единице.

Пример выполнения операции ИЛИ:

1-е число (находится в аккумуляторе)	10010011
2-е число (находится в регистре R, ячейке памяти M или во 2-м байте команды)	11001110
Результат операции ИЛИ над 1-м и 2-м числами (размещается в аккумуляторе)	11011111

Пример 46.1.

Задание. Произвести операцию логического ИЛИ над числами, расположенными в портах ввода с адресами $0F_{16}$ и $F0_{16}$. Результат операции выдать в порт вывода с адресом 11_{16}.

Р е ш е н и е. Пользуясь командой IN N, пересылаем первое число из порта OF_{16} в аккумулятор А:

IN 0FH

Освобождаем аккумулятор для второго числа, пересылая содержимое аккумулятора в регистр В:

MOV B,A

Пересылаем в аккумулятор из порта ввода $F0_{16}$ второе число:

IN F0H

Таким образом, первое число находится в аккумуляторе, а второе — в регистре В, и мы можем, пользуясь командой ORA R выполнить между ними операцию ИЛИ:

ORA B

Затем результат операции ИЛИ пересылаем в порт с адресом 11_{16}:

OUT 11H

Следовательно, программа решения задания будет состоять из таких пяти команд:

IN OFH — пересылка в аккумулятор числа из порта $0F_{16}$.
MOV B,A — пересылка в регистр В содержимого аккумулятора А.
IN FON — пересылка в аккумулятор числа из порта $F0_{16}$.
ORA B — операция ИЛИ над содержимым аккумулятора и регистра В.
OUT 11H — пересылка в порт 11_{16} содержимого аккумулятора.

48, 49. XRA	50. XRI

Эти три команды выполняют логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ. Они полностью соответствуют структуре и правилам выполнения уже рассмотренных команд И и ИЛИ. А именно: команда XRA R выполняет операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над числом, расположенным в аккумуляторе, и числом, находящимся в регистре. Команда XRA M выполняет ту же операцию над числами, расположенными в аккумуляторе и ячейке памяти М, а команда XRI K выполняет ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и числом К, записанным во второй байт команды.

Действие ИСКЛЮЧАЮЩЕЕ ИЛИ сводится к тому, что единица образуется тогда и только тогда, когда складываются числа, имеющие различные значения истинности (у одного числа — нуль, а у другого единицы или, наоборот, у первого — единица, а у второго — нуль) и образуют нуль, когда числа имеют одинаковые значения истинности (оба равны нулю или оба равны единице).

Пример выполнения операции ИСКЛЮЧАЮЩЕЕ ИЛИ:

1-е число (находится в аккумуляторе)	10010011
2-е число (находится в регистре R, в ячейке памяти М или во 2-м байте команды)	11001110
Результат операции ИСКЛЮЧАЮЩЕЕ ИЛИ над 1-м и 2-м числами (размещается в аккумуляторе)	01011101

Как видим из этого примера, любой разряд числа, находящегося в аккумуляторе, изменяется на противоположное значение (инвертируется) с помощью единицы, расположенной в таком же разряде второго числа и не изменяется, если в разряде второго числа записан нуль. Проверим это, начиная с правых (нулевых) разрядов. В нулевом разряде второго числа записан нуль. Следовательно, этот нуль не должен инвертировать единицу первого числа. А вот в первом разряде записана единица, которая инвертирует единицу первого числа и в результате операции (в ее первом разряде) пишется нуль.

Этим обстоятельством часто пользуются для установки разрядов числа аккумулятора в нужное значение (в нуль или единицу).

Пример 50.1.

Задание. В аккумуляторе содержится число 10101010_2. Требуется установить все разряды этого числа в единицу.

Р е ш е н и е. Для выполнения задания необходимо на нулевые разряды числа воздействовать единицами во втором числе при помощи операции ИСКЛЮЧАЮЩЕЕ ИЛИ. Это можно выполнить, использовав команду XRI K, записав во второй байт команды двоичное число 01010101:

XRI 01010101 B

Пример 50.2.

Задание. Установить все разряды числа 10101010_2, содержащегося в аккумуляторе предыдущего примера, в нуль.

Р е ш е н и е. Эту задачу можно выполнить, записав во второй байт команды XRI K число K = 10101010_2, чтобы воздействовать на все единицы разрядов числа аккумулятора единицами второго числа. Однако проще обнулить (словом «обнулить» часто называют операцию получения нулей во всех разрядах числа) содержимое аккумулятора при помощи команды XRA R, т, е. совершить операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над двумя одинаковыми значениями содержимого аккумулятора.

Таким образом, задание может быть выполнено с помощью одной из двух команд:

XRI 10101010 B

или

XRA A

Если обратиться к колонке 6 таблицы 1, то можно увидеть, что выполнение команды XRI K занимает семь машинных тактов, а выполнение команды XRA R — только четыре. Следовательно, второе решение более предпочтительно.

51. CMA

Команда CMA выполняет логическую операцию НЕ. В результате этой операции все единицы во всех разрядах числа заменяются на нули, а все нули — на единицы, т. е. число инвертируется.

Такое инвертирование часто применяют для получения обратного кода числа.

Пример 51.1.

Задание. Получить обратный код числа, находящегося в ячейке памяти M, имеющей адрес 0001_{16}.

Р е ш е н и е. Размещаем адрес ячейки памяти M в регистровой паре HL:

LXI H, 0001 H

Пересылаем содержимое ячейки памяти M в аккумулятор A:

MOV A, M

Инвертируем содержимое аккумулятора:

CM A

Пересылаем инвертированное значение числа в ту же ячейку памяти M:

MOV M, A

Если нужно получить дополнительный код числа, то следует применить команду INR M, увеличивающую число на единицу.

52, 53. CMP

54. CPI

С помощью этих трех команд выполняют сравнение двух чисел, одно из которых должно находиться в аккумуляторе. По команде CMP R происходит сравнение чисел, содержащихся в аккумуляторе и в регистре R, по команде CMP M — чисел,

находящихся в аккумуляторе и ячейке памяти M, а по команде CPI K числа K, записанного во втором байте команды, и числа, расположенного в аккумуляторе.

Сравнение выполняется путем внутреннего вычитания содержимого регистра R, ячейки памяти M или числа K из содержимого аккумулятора. Результат вычитания определяется по состоянию триггеров Z и AC, входящих в состав регистра признаков F. Если сравниваемые числа равны друг другу, то триггер Z устанавливается в единицу, а если не равны, то сбрасывается в нуль. Когда сравниваемое число больше того, которое находится в аккумуляторе, то срабатывает триггер AC, устанавливаясь в единицу, а если меньше, то устанавливается в нуль.

Команда CMP по своему действию соответствует команде вычитания SUB с той лишь разницей, что результат вычислений после выполнения команды CMP не размещается в аккумуляторе, и число, находящееся в аккумуляторе, не изменяет своего значения.

Примеры написания команд CMP и CPI даны в разделе описания команд группы переходов.

55. RAL	56. RLC	57. RAR	58. RRC

Эти четыре команды выполняют операцию сдвига числа, находящегося в аккумуляторе. Напомним, что операцией сдвига называют перемещение каждого разряда числа на одну позицию влево (команды RAL и RLC) или вправо (команды RAR и RRC). Если единица находилась, например, в третьем разряде числа (рис. 67, *а* вклейки), то после сдвига она окажется в четвертом (рис. 67, *б* вклейки) или во втором (рис. 67, *в* вклейки) разряде.

В операциях сдвига, выполняемых МП—К580, принимает участие триггер С регистра признаков F, расположенный левее старших разрядов аккумулятора и представляющий собой как бы продолжение аккумулятора (его «восьмой» разряд). На рис. 69 вклейки схеметически показано, как происходит сдвиг числа, находящегося в аккумуляторе, под воздействием команды RAL и наличии триггера С. Для наглядности действия в аккумулятор загружено число, содержащее единицу в самом старшем (седьмом) и самом младшем (нулевом) разрядах, а именно: число 10000001_2.

Команда RAL производит сдвиг числа влево на один разряд, в результате чего единица перемещается из седьмого разряда в триггер С, а на место единицы поступает нуль из шестого разряда. Аналогично, нуль из триггера С помещается в нулевой разряд аккумулятора, а единица, находившаяся там, «выталкивается» в первый разряд. Следующая команда RAL снова сдвигает число влево. При этом единица из триггера С переходит в нулевой разряд аккумулятора, а единица из первого разряда — во второй разряд. Многократно выполняя команду RAL, можно достичь первоначального расположения единиц в разрядах числа. Такое состояние образуется после девятой по счету команды RAL. При этом говорят, что произошел полный цикл сдвига. Продолжая выполнять команду RAL, можно получить несколько циклов сдвига.

На рис. 68, *а* и *б* вклейки указаны схемы сдвигов числа, называемые циклом сдвига через триггер переноса С. Помимо такого порядка сдвига, применяются еще и другие (рис. 68, *в* и *г* вклейки). Здесь путь движения какой-либо единицы в разряде числа более короткий, так как она движется, минуя триггер С, и поэтому из седьмого разряда единица поступает в свой нулевой разряд. Однако при таком переходе в триггер С посылается копия содержимого седьмого разряда, следовательно триггер С все время дублирует, повторяет значение старшего разряда числа, находящегося в аккумуляторе.

На рис. 68, *б* и *г* вклейки показаны схемы сдвигов числа вправо под действием команд RRC и RAR.

Команды сдвига применяются для выполнения операций умножения, деления и некоторых преобразований данных, поступающих в микропроцессор.

Пример 56.1.

Задание. Требуется получить в триггере переноса С значение, соответствующее 5-му разряду числа, находящегося в регистре В.

Р е ш е н и е. Пересылаем содержимое регистра B в аккумулятор:

MOV A,B

Чтобы пятый разряд регистра B был сдвинут в триггер C («восьмой» разряд), необходимо выполнить три команды RLC (8 — 5 = 3):

RLC — сдвиг влево на один разряд. Содержимое пятого разряда передается в шестой разряд.

RLC — сдвиг влево на один разряд. Содержимое шестого разряда передается в седьмой разряд.

RLC — сдвиг влево на один разряд. Содержимое седьмого разряда передается в триггер C.

59. STC

Перед выполнением некоторых логических и арифметических операций требуется предварительная установка в единицу триггера переноса C, входящего в состав регистра признаков.

В каком бы положении не находился триггер C после команды STC, он устанавливается в единицу.

Команды перехода

Сначала поясним, что понимают под операцией перехода, и кратко расскажем о действии триггеров регистра признаков F, так как переходы часто выполняются в зависимости от состояния этих триггеров.

Как известно, программа работы микропроцессора представляет собой последовательность команд, расположенных одна за другой в ячейках памяти. Обычно выполнение программы начинается с начальной (нулевой) ячейки и продолжается в соответствии с возрастающей последовательностью номеров (адреса) ячеек до тех пор, пока не будет выполнена поставленная задача. При этом программа может насчитывать сотни тысяч и даже миллионы команд. Замечено, что в этом множестве команд довольно часто встречаются небольшие группы одинаковых команд, выполняющих определенные, всегда одни и те же действия, например: умножение двух чисел или их деление, задержку выполнения команды на какое-то время и др.

Чтобы не повторять в программе перечень таких команд, поступают следующим образом. Их записывают в виде короткого списка команд, называемого **подпрограммой**, и помещают на хранение в определенное, известное программисту место в памяти. Когда пишут программу и встречаются с необходимостью выполнить подпрограмму, то вместо того, чтобы перечислять в программе все команды подпрограммы, пишут всего одну команду: «Перейти к выполнению подпрограммы». Вот такой переход от программы к подпрограмме и составляет сущность операции перехода. Этот вид перехода получил название безусловного в отличие от условного, который происходит не тогда, когда этого пожелает программист, а в зависимости от состояния сцециальных триггеров условий, имеющихся в схеме микропроцессора. Разные типы микропроцессоров имеют разное количество и разные функциональные свойства таких триггеров. МП—К580 насчитывает пять триггеров условий, обозначаемых буквами Z, S, P, C и AC. Для удобства эти триггеры сведены в один регистр, называемый **регистром признаков** (регистр F). Поясним назначение и действия триггеров регистра F.

Триггер Z именуется признаком нуля. Он устанавливается в единицу (Z = 1), если в аккумуляторе после выполнения какой-либо команды образуется нулевой результат, то есть появляется число 00000000. Во всех других случаях триггер устанавливается в нуль (Z = 0).

Триггер S называется признаком знака. Он устанавливается в единицу (S = 1), когда результат выполнения операции является отрицательным числом.

Триггер P называется признаком четности; он устанавливается в единицу (P = 1), когда двоичный код результата операции содержит четное число единиц восьмиразрядного числа.

Триггер C называется триггером переноса. Он устанавливается в единицу (C = 1), когда в результате сложения двух чисел в аккумуляторе образуется переполнение, возникает число, длиннее восьми разрядов.

Триггер AC обозначает признак вспомогательного переноса. В отличие от триггера C он устанавливается в единицу (AC = 1) когда происходит перенос из третьего разряда числа, находящегося в аккумуляторе, в его четвертый разряд. Этот признак используется при сложении чисел, записанных в особом двоично-десятичном коде (его называют также кодом BCD).

Изменение состояния триггеров Z, S, P, C и AC после выполнения каждой команды, указано в колонке 7 табл. 1.

60. JMP

По команде JMP ADR выполняется безусловный переход к выполнению другой программы, адрес которой указывается во втором и третьем байтах команды. Из колонки 5 видно, что двоичный код этой команды равен 11000011 или в шестнадцатеричной форме C3.

Для возвращения к прежней программе в конце новой программы необходимо снова записать команду JMP ADR, указав в ней адрес того места программы, с которого начался переход.

Примеры написания команд переходов даны при изучении правил составления программ.

61. PCHL

Команда PCHL осуществляет безусловный переход к выполнению программы, адрес которой находится в регистровой паре HL. Команда PCHL однобайтовая.

Перед выполнением команды необходимо в регистровую пару HL поместить начальный адрес новой программы, в регистр H — старший полуадрес, а в регистр L — младший.

62. CALL

65. RET

Команда CALL ADR осуществляет безусловный переход к выполнению подпрограммы, а команда RET — безусловное возвращение из подпрограммы к выполнению основной программы.

В отличие от команды JMP ADR, команда CALL ADR осуществляет автоматическое возвращение к основной программе после выполнения подпрограммы. С этой целью адрес возврата размещается в ячейках памяти стека, а в конце подпрограммы пишут команду возврата RET.

63. J(F)

По команде J(F) ADR осуществляется условный переход к выполнению программы, адрес которой записан во втором и в третьем байтах команды. Если же условие не удовлетворяется, и триггер признака не устанавливается в единицу, то переход не происходит, а продолжается выполнение команд основной программы. В первом байте команды указывается двоичный код триггеров Z, S, R и C, взятый из колонки 8 табл. 1.

Иногда в литературе вместо обобщенного вида команды условного перехода J(F) ADR указывают восемь отдельных команд, учитывающих признаки конкретного триггера, входящего в состав регистра F. Для наглядности перечислим эти частные команды с указанием признака условия перехода для каждой из них:

JNZ ADR — ненулевой результат (Z = 0);
JZ ADR — нулевой результат (Z = 1);
JP ADR — число положительное (S = 0);
JM ADR — число отрицательное (S = 1);
JPO ADR — нечетный результат (P = 0);
JPE ADR — четный результат (P = 1);
JNC ADR — нет переноса (C = 0);
JS ADR — есть перенос (C = 1).

64. C(F)	66. R(F)

Команда C(F) ADR осуществляет условный переход к выполнению подпрограммы, адрес которой записан во втором и в третьем байтах команды, а команда R(F) выполняет условный возврат из подпрограммы в основную программу.

Команды C(F) ADR и R(F) аналогичны командам CALL ADR и RET и отличаются от них только тем, что переход к подпрограмме и возвращение из нее осуществляется в зависимости от состояния триггеров регистра признаков. Так как состояний триггеров насчитывается восемь (два состояния, умноженные на четыре триггера), то и число частных команд для C(F) ADR и R(F) может быть записано по восемь команд. Ниже приводится список этих команд:

CNZ ADR и RNZ, если Z = 0;
CZ ADR и RZ, если Z = 1;
CP ADR и RP, если S = 0;
CM ADR и RM, если S = 1;
CPO ADR и RPO, если P = 0;
CPE ADR и RPE, если P = 1;
CNC ADR и RNC, если C = 0;
CC ADR и RC, если C = 1.

В этих командах так же, как и в частных командах предыдущего случая (см. команду J(F) ADR), вместо обобщенного обозначения регистра признаков F пишут частное значение какого-либо из триггеров, входящих в его состав, то есть вместо C(F) пишут CNZ, CZ, CP, CM, CPO, CPE, CNC и CC, где

NZ — ненулевой результат (Z = 0);
Z — нулевой результат (Z = 1);
P — положительное число (S = 0);
M — отрицательное число (S = 1);
PO — нечетный результат (P = 0);
PE — четный результат (P = 1);
NC — нет переноса (C = 0);
C — есть перенос (C = 1).

67. RST

Команда RST X осуществляет переход к одной из восьми специальных подпрограмм режима прерывания, расположенных по фиксированному адресу X.

Команда RST X является однобайтовым вариантом команды перехода с возвратом. Адрес очередной команды определяется тремя разрядами X, содержащимися в коде операции. Код команды поступает в микропроцессор непосредственно из внешнего устройства.

Команда RST X позволяет выполнять срочный переход (прерывание) на другую программу в аварийных ситуациях. Действия команды прерывания подобно стековым операциям довольно сложные и мы их подробно рассматривать не будем.

Специальные команды

Эта группа команд не производит каких-либо действий над данными. Команды не имеют операнда, а только код операции. Большинство из них является служебными командами, изменяющими режим работы микропроцессора.

68. EI	69. DI

В системе команд МП—К580 имеется две специальные команды, управляющие режимом прерывания. Команда EI — разрешает прерывание, а команда DI — запрещает.

В разд. 2.3 указывалось, что режим прерывания используется в крайних ситуациях для прерывания основной программы и перехода на выполнение спе-

циальной подпрограммы обработки аварийных сигналов. Если прерывания не запрещены, (в программе была своевременно записана команда EI, разрещающая прерывание), то микропроцессор приостанавливает выполнение своей программы и переходит на обработку внеочередной подпрограммы. Если же программист забыл поставить в нужном месте команду EI, то микропроцессор будет игнорировать внешние запросы прерывания, что может привести к нежелательным последствиям. Риск возникновения подобной ситуации можно уменьшить с помощью команды RST X, рассмотренной выше. Подавая специальным образом эту команду, можно заставить микропроцессор перейти к выполнению аварийного прерывания в случае отсутствия в программе команды EI.

В ряде случаев может оказаться, что прерывание программы нецелесообразно, тогда с помощью команды DI можно запретить прерывание.

70. NOP

Команда NOP является пустой операцией.

Микропроцессор не выполняет здесь каких-либо полезных действий по обработке данных, а просто затрачивает некоторое время для различных служебных переключений во время четырех машинных тактов. Зная длительность машинного такта (она равна длительности периода тактовых сигналов Ф1, Ф2, поступающих из тактового генератора), можно с помощью некоторого количества команд NOP получить нужную задержку по времени выполнения программы.

Команду NOP часто используют для пространственного заполнения программы. Если появляется необходимость ввести в программу дополнительную команду, то ее размещают взамен команд NOP, записанных в нужном месте программы.

71. HLT

По команде HLT микропроцессор останавливает выполнение очередной команды и переходит в режим останова. Вывести его из этого состояния можно, подав сигнал сброса или прерывания.

2. Как написать программу для микроЭВМ

Общие сведения.

В предыдущих главах рассказано об устройстве **аппаратной** (технической) части микроЭВМ и даны советы по ее сборке. Но так как аппаратная часть неразрывно связана с **программой**, то, собирая микроЭВМ, следует уделить достаточное внимание изучению **программирования.** С этой целью расскажем об общем принципе программирования, о практических навыках составления простейших учебных **программ**, позволяющих понять работу микроЭВМ, а также о том, как использовать для собранной микроЭВМ готовые программы.

Программа микроЭВМ определяется решаемой задачей. Если ставится простая задача, то программа получается короткой и размещается в небольшом количестве ячеек памяти. Например, составленная нами (см. табл. 2.14) программа сложения двух чисел, помещалась в восьми ячейках памяти. Более сложная программа умножения двух чисел требует для размещения несколько десятков ячеек, а программа управления движением транспорта на перекрестке — несколько сотен. В настоящее время многие программы микроЭВМ занимают в памяти сотни тысяч и даже миллионы ячеек.

Сложность решаемых задач и объем памяти, необходимый для размещения программы, существенно влияют на смысл, который вкладывается в понятие программирование. При написании простых, коротких программ под программированием понимают составление перечня команд, способных выполнить с помощью микроЭВМ поставленную задачу. В то же время при создании сложных программ, насчитывающих сотни тысяч команд, понятие программирования преобразуется в обширную область теоретических и практических действий, включающих в себя постановку задачи, разработку **алгоритма** ее решения, составление **схемы алгоритма,** выбор **языка программирования,** написание программы на выбранном языке,

перевод написанной программы на машинный язык, **отладку программы** и некоторые другие этапы.

Чтобы представить себе в общем виде всю сложность процесса программирования современных микроЭВМ, кратко рассмотрим каждый из перечисленных этапов.

При постановке задачи, помимо ее общего наименования, предполагается запись условия задачи с помощью математических выражений, обозначений, а также четкое формулирование исходных данных, возможных ограничений и желаемых результатов.

После постановки задачи определяется правило ее решения. Такое правило в виде точного и однозначного предписания о выполнении в определенной последовательности действий, направленных на решение поставленной задачи, называют алгоритмом, а каждое отдельное действие алгоритма именуют **командой**. Примером алгоритма могут служить известные правила выполнения арифметических задач над многозначными числами, которые можно записать в виде таблицы (списка) указаний, поясняющих, как подписывать числа друг под другом, какие действия совершать с цифрами и знаками чисел.

Кроме табличного способа записи алгоритма (см. табл. 5), существуют и другие, например графическое изображение алгоритма в виде схемы, состоящей из геометрических фигур, соединенных друг с другом стрелками (см. рис. 70 вклейки). Придавая фигурам различную форму, можно наглядно получить зрительное представление о том, какое действие выполняет каждая фигура. При этом стрелки указывают, в каком направлении и в какой последовательности выполняются действия.

После составления алгоритма приступают к написанию программы в виде упорядоченной последовательности команд, служащих приказом для микроЭВМ выполнить конкретное действие. В отличие от алгоритма программу пишут с помощью условных знаков, букв и цифр, называемых языком программирования. Запись команд в двоичном виде, которой мы пользовались при составлении табл. 2.14, именуют **машинным языком.** Помимо машинного языка, существует большое количество других языков программирования. Тот или иной язык применяется в зависимости от типа решаемой задачи и технических возможностей микроЭВМ. На каком бы языке не была написана программа, ее перед вводом в память приходится переводить на машинный язык, так как микроЭВМ воспринимает только команды, состоящие из комбинации нулей и единиц (двоичное представление чисел). Существуют специальные программы для перевода с выбранного языка программирования на машинный язык. Такие программы называют **транслятором.**

После перевода (трансляции) программы приступают к ее отладке, заключающейся в проверке правильности составления программы, выявления ошибок и их устранения. Для исправления недостатков иногда приходится перепрограммировать заново какую-то часть программы или устранять отдельные ошибки в написании символов. Все эти операции чаще всего совершают не вручную, а при помощи специальных отладочных программ и особых технических средств. По завершению отладки программа может поступить в эксплуатацию для выполнения поставленной задачи при помощи микроЭВМ.

Из описанных этапов видно, на сколько трудоемким является процесс программирования, требующий обширных знаний, большого опыта и значительных усилий. В виде примера укажем, что опытный программист за месяц работы может составить и отладить программу объемом всего около 200 команд.

Чтобы облегчить и ускорить процесс программирования, применяют различные средства и действия, направленные на автоматизацию этого процесса. Основой автоматизации является использование языков программирования, близких к естественному человеческому восприятию **(языков высокого уровня)**, а также применение большого количества вспомогательных (сервисных) программ, объединенных в **операционную систему.**

Включения нужной программы операционной системы осуществляется простым нажатием клавиши на пульте управления и позволяет автоматически выполнять различные действия по составлению, редактированию, переводу, пересылке, загрузке в память и отладке программ, а также по автоматическому управлению режимами работы микроЭВМ.

Как показал опыт работы радиотехнического кружка при Киевской городской станции юного техника, где под руководством одного из авторов кружковцы собирали собственную микроЭВМ, изучение программирования лучше всего начинать с машинного языка, который органически связан с изготавливаемой микроЭВМ. Освоив машинный язык, следует переходить к языку ассемблера, являющегося своеобразным отображением машинного языка. И лишь на втором этапе сборки микроЭВМ (ее модернизации) целесообразно ознакомиться с алгоритмическим языком (для тех, кто не изучал его в школе) и языком **бейсик**. В соответствии со сказанным и построим наше описание программирования.

Каждый язык программирования имеет свои достоинства и недостатки. Машинный язык хорош тем, что отпадает необходимость в переводе с этого языка. Программа сразу вводится в память. Однако программировать на машинном языке неудобно и утомительно. Нужно знать устройство микропроцессора и его систему команд. А главное при написании нулей и единиц легко допустить ошибку, которую потом сложно отыскать. Кроме того, очень трудно расширять или сокращать уже написанные программы. Поэтому машинный язык применяется только для составления коротких, чаще всего учебных программ, а также для написания отдельных команд при отладке программы. Машинный язык называют **языком программирования низкого уровня**. В отличие от этого языка существует большое количество алгоритмических **языков высокого уровня** (бейсик, паскаль, ПЛ/М и др.), которые свободны от многих недостатков машинного языка. Программировать на этих языках просто и удобно, так как они приближаются к привычному математическому и даже естественному человеческому языку.

Вместе с тем для каждой программы, составленной на языке высокого уровня, требуется выполнить перевод на машинный язык. Программа перевода получается очень сложной, требующей больших объемов памяти, значительных затрат машинного времени и больших усилий для их составления. Поэтому программы на языке высокого уровня целесообразно писать только для очень длинных программ, насчитывающих многие тысячи байтов. В то же время большинство повседневных практических задач, решаемых на микроЭВМ, требуют программ длиной в сотни байт. Писать такие программы, занимающие промежуточное положение между очень короткими программами (десятки байт) и очень длинными (тысячи байт), целесообразнее всего на **языке ассемблера,** который объединяет в себе некоторые достоинства самого нижнего и самого верхнего уровней языков программирования. Язык ассемблера по своей структуре очень близок к машинному языку и отличается от него только тем, что команды, адреса и данные обозначаются с помощью алфавитно-цифровых символов, имеющих мнемонический смысл, облегчающий их запоминание и применение. Написанная на языке ассемблера программа транслируется на машинный язык с помощью специальной программы, называемой **ассемблером**. Программа ассемблера очень простая и занимает в памяти сравнительно мало места. Представляется возможным выполнять перевод с языка ассемблера вручную, при помощи тумблерного пульта управления или шестнадцатеричной клавиатуры.

Язык ассемблера

Язык программирования ассемблера, как и всякий язык, имеет азбуку, словарь и грамматику. Азбука ассемблера состоит из букв, цифр и специальных символов. Словарем является система команд. А грамматика представляет собой правила написания программы. Познакомимся с этими правилами.

В отличие от формы записи программы сложения двух чисел, выполненной нами ранее на машинном языке (см. табл. 2.14, с. 58), запись программы на языке ассемблера исполняется в виде таблицы, состоящей из четырех колонок (полей), которые называются: метка, операция, операнд и комментарий (табл. 2). Записи в этих полях отделяются друг от друга не вертикальными линиями, как это делается в традиционных таблицах, а с помощью пробела между полями. Кроме того, метка отделяется от операции при помощи двоеточия, а операнд от комментария — точкой с запятой.

Из этих четырех полей обязательным является наличие записи в поле операции, а в остальных полях (или некоторых из них) записи может и не быть.

Таблица 2

Код адреса	Код данных	Метка	Операция	Операнд	Комментарий
0000	DB	Начало:	IN	0H;	;Переслать данные в ак-
0001	00				;кумулятор из порта, ;имеющего адрес ОН
0002	47		MOV	B, A;	;Переслать данные в ре- ;гистр В из аккумулятора
0003	DB		IN	1H;	;Переслать данные в ак- ;кумулятор из порта, ;имеющего адрес 1Н
0004	01				;Сложить данные реги-
0005	80		ADD	B;	;стра В и аккумулятора
0006	D3		OUT	0H;	;Выдать данные из ак-
0007	00				;кумулятора в порт, ;имеющий адрес 0Н

Поле метки предназначено для записи условного адреса команды в виде символического слова (имени) на русском или английском языках, означающем действие команды. Метка требуется тогда, когда в программе встречается ссылка на эту команду. И нужно перейти к выполнению данной команды. В этом случае в программе (в колонке операнда) вместо адреса перехода пишут название метки.

Поле операции содержит сокращенную мнемоническую буквенную запись команды, выбрачной из системы команд данного микропроцессора.

Поле операндов отведено для одного или нескольких операндов, представляющих собой данные, над которыми выполняется операция. В поле операнда может быть помещен не сам операнд, а его адрес, в том числе условный адрес в виде названия метки.

В поле комментария помещаются различные сведения для пояснения команды или фрагмента программы. Эта запись выполняется чаще всего в повелительной, командной форме: «Переслать», «Сложить» и др. Однако допускается и произвольная запись в любой другой форме. Например, «Пересылка», «Сложение» и др., как это сделано в табл. 3.

Если текст комментария не помещается в одной строке, то новую строку обозначают слева точкой с запятой.

Комментарий не воспринимается микропроцессором. Он лишь повышает удобство чтения программы для того, кто желает разобраться в ней.

Программа, написанная на языке ассемблера в виде четырех полей, называется **исходной.** Она удобна и понятна для программиста, но микроЭВМ воспроизвести ее не может, потому что единственный язык, который понимает микроЭВМ, это машинный язык, состоящий из комбинаций нулей и единиц (двоичный код). Поэтому исходную программу перед исполнением нужно перевести в так называемую **объектную** программу, представляющую собой запись программы на машинном языке (в двоичном коде).

Если перевод предполагается выполнять вручную, то слева от четырех полей табл. 2 необходимо предусмотреть место для размещения еще двух полей. Первое поле «Код адреса» служит для записи адресов ячеек памяти, в которой размещаются байты команд, а второе — «Код данных» предназначено для записи кода байтов команд. Записи в этих полях производятся в том же порядке, в каком выполнялись записи в колонках 2 и 4 табл. 2.14. А именно: пользуясь колонкой 5 табл. 1, определяем структуру команды и узнаем, сколько и какие байты входят в ее

Таблица 3

Код адреса	Код	Метка	Операция	Операнд	Комментарий
0000	DB 00	Начало:	IN	0H	;Пересылка данных в ;аккумулятор из порта, ;имеющего адрес 0H
0002	47		MOV	B, A	;Пересылка данных в ;регистр B из аккумуля- ;тора
0003	DB 01		IN	1H	;Пересылка данных в ;аккумулятор из порта, ;имеющего адрес 1H
0005	80		ADD	B	;Сложение данных ре- ;гистра B и аккумулятора
0006	D3 00		OUT	0H	;Выдача данных из ;аккумулятора в порт ;с адресом 0H

состав. Затем записываем кодовое значение этих байтов в поле «Код данных», а потом для каждого байта в поле «Код адреса» записываем адрес ячейки памяти, в которой он будет размещен.

Если ввод программы в память предполагается выполнять при помощи переключателей, тумблерного пульта управления, то адреса и коды левых полей табл. 2 записывают в двоичном виде, а при вводе с помощью шестнадцатеричной клавиатуры — в шестнадцатеричном виде.

В табл. 2 поля «Код адреса» и «Код данных» для экономии места записаны в шестнадцатеричном виде. Численное значение этих записей соответствует аналогичным записям в колонках 2 и 4 табл. 2.15, представленных в двоичном коде.

Учитывая, что записи кода адреса и данных в левых полях занимают для двухбайтных команд две строки, а для трехбайтных — три, приходится, выполняя построчную запись в правых полях («Операция» и «Операнд»), предусматривать пробел между строками, что не всегда удобно. Поэтому начальный адрес и коды всех байтов, относящихся к данной команде, часто записывают в одной строке, так как это сделано в табл. 3. В поле кода адреса записывают адрес ячейки памяти, где размещается первый байт команды, а в поле кодов данных последовательно друг за другом коды всех байтов (одного, двух или трех), выполняемых данной командой. Следующий адрес, записываемый в поле кода адреса, будет порядковым только в том случае, если предыдущая команда была однобайтной, а для двухбайтной он будет на два больше прежнего, а для трехбайтных — на три.

В целях уплотнения записи длинных объектных программ, насчитывающих сотни байт, их иногда записывают в компактной форме, так как это выполнено в табл. 4. Каждая строка этой таблицы начинается с записи четырехразрядного шестнадцатеричного числа, представляющего собой адрес размещения в памяти первого из шестнадцати байтов, расположенных в строке правее адреса и записанных в виде двухразрядных шестнадцатеричных чисел. Такая форма записи занимает мало места и очень удобна для ввода программы в память при помощи шестнадцатеричной клавиатуры. Однако запись лишается смысловой наглядности, так как очень трудно понять, какие действия выполняют коды байтов написанных команд.

Если работу с микроЭВМ выполняют при помощи шестнадцатеричной клавиатуры или стандартной клавиатуры дисплея, то удобно пользоваться табл. 11 При-

Таблица 4

0000	FA	00	47	FA	01	80	C3	00	C3	82	03	EB	00	E6	0B	FE
0010	19	C3	01	06	58	66	7E	03	C1	C3	16	08	07	00	01	5C
0020	AB	00	E6	06	CE	0A	02	00	21	17	01	11	00	00	72	7E
0030	BC	A3	00	00	00	00	00	00	00	00	00	00	00	00	00	00

ложения. Для расшифровки готовых программ, записанных по форме табл. 4, целесообразно применять рис. 73 вклейки.

Рассмотрим несколько примеров составления программ на языке ассемблера.

Для начала рассмотрим простой пример написания на языке ассемблера известной нам программы сложения двух чисел, которая была составлена выше на машинном языке (см. табл. 2.15). Запись на языке ассемблера начинаем с записи метки (см. табл. 2). Возможно эта программа понадобится нам в дальнейшем и мы сможем обратиться к ней, если вставим сейчас в программу название метки. Поэтому присваиваем метке имя «Начало», записываем ее в поле меток и отделяем от других полей двоеточием. Затем переходим к записям в полях «Операция» и «Операнд». Согласно порядку, предусмотренному алгоритмом решения задачи, нам предстоит переслать в регистр А первое слагаемое из порта ввода № 0. Пользуясь табл. 1 системы команд МП—К580, выбираем для этой операции команду IN N. Записываем мнемоническое обозначение IN в поле «Операция», а в поле «Операнд» пишем адрес порта ввода в шестнадцатеричной форме. Эта запись будет выглядеть так: 0H. Напоминаем, что символ H означает шестнадцатеричную форму числа, а 0 является нулевым адресом порта ввода. В поле комментария сообщаем о требуемых действиях: «Переслать данные в аккумулятор из порта, имеющего адрес 0H». Далее, согласно очередному действию алгоритма, следует освободить регистр А, переслав его данные в регистр В. Эту операцию выполняем при помощи команды MOV R1, R2, записывая на полях «Операция» и «Операнд» соответственно MOV и B, A, а в Комментарии указываем: «Переслать данные в регистр В из аккумулятора». После чего переходим к размещению в освободившийся аккумулятор второго слагаемого, находящегося в порту с адресом 1H. Для этого пользуемся той же командой IN N. В поле «Операция» записываем IN, в поле «Операнд» — 1H и в поле «Комментарий» — «Переслать данные в аккумулятор из порта, имеющего адрес 1H».

Приступаем к сложению чисел. Сложение выполняем при помощи команды ADD R. Для этого в поле «Операция» записываем ADD, а в поле «Операнд» — обозначение регистра В. В Комментарий заносим: «;Сложить данные регистра В и аккумулятора». В заключение необходимо выдать результат сложения в порт вывода с адресом 0H.

Используя команду OUT N, пишем в строке нужных полей: OUT, пробел, 0H, пробел, точка с запятой и комментарий: «Выдать данные из аккумулятора в порт, имеющий адрес 0H».

На этом составление программы сложения двух чисел на языке ассемблера заканчиваем. Чтобы ввести программу в память, нужно заполнить левые поля («Код адреса» и «Код данных») табл. 2, и, пользуясь программатором, ввести значения кода этих полей в ячейки памяти.

Рассмотрим составление более сложной программы на языке ассемблера для программной реализации устройства, называемого генератором импульсов. Это устройство широко применяется в электронике при изготовлении электромузыкальных инструментов, генерируя импульсы звуковой частоты (20—10 000 Гц), для изготовления электронных часов, создавая секундные импульсы (1 Гц), а также в телевизионных играх для формирования синхронизирующих импульсов строк (15 625 Гц) и кадров (50 Гц).

Обычно генераторы импульсов собирают по разным схемам на транзисторах или логических интегральных микросхемах. Мы же создадим генератор импульсов программно, путем составления программы для микроЭВМ.

Как генератор импульсов можно собрать с помощью различных электрических схем, так и программу генератора можно составить, используя несколько вариантов алгоритма. Наиболее часто используют такой алгоритм: в один из регистров РОН микропроцессора вводят число, а затем с помощью соответствующей команды

вычитают из этого числа единицу. После вычитания проверяют, не стало ли число равным нулю. Если нет, то снова вычитают, и так поступают до тех пор, пока число не уменьшится до нуля. Как только в регистре образуется нуль, в порт вывода автоматически выдается импульс и процесс, называемый циклом, повторяется, а именно: в регистр снова засылается то же число и опять постепенно уменьшается до нуля. Чтобы совершить цикл вычитания единицы из числа, затрачивается время, поэтому импульсы в порт выдаются с некоторым временным интервалом. Чем большее число было записано в регистр, тем большим будет промежуток времени между выдаваемыми импульсами, имитирующими собой выход генератора импульсов.

В табл. 5 записан табличный алгоритм рассмотренных нами действий. В колонке 1 перечислены шаги алгоритма, а в колонке 2 указаны выполняемые действия. В регистр А здесь заносится в общем случае число X. В зависимости от требуемой частоты выдаваемых импульсов это число должно иметь конкретное значение. Как определить значение X, будет сказано далее.

Табличная запись алгоритма иногда дополняется схемой алгоритма, изображенной на рис. 70 вклейки. Из многочисленных видов фигур, служащих для изображения схемы алгоритма, чаще всего пользуются четырьмя фигурами, представленными на рис. 71 вклейки. Прямоугольник с полукруглыми сторонами (рис. 71, *а, е* вклейки) показывает начало или конец выполняемого алгоритма, обычный прямоугольник (рис. 71, *б, в* вклейки) отображает действия, не требующие проверки выполнения условия, ромб (рис. 71, *г* вклейки) означает проверку выполнения условий и выбор одного из двух возможных действий в зависимости от ее результата и, наконец, параллелограмм изображает ввод или вывод данных из внешнего устройства (рис. 71, *д* вклейки).

В последнее время в связи с широким применением языков программирования высокого уровня схемой алгоритма пользуются редко. Однако в литературе по вычислительной технике она еще встречается довольно часто. Поэтому иметь общее представление о структуре схемы весьма полезно. Кроме того, схема алгоритма благодаря своей наглядности почти всегда применяется при составлении учебных программ, удобно иллюстрируя выбираемый способ решения задачи.

В соответствии с алгоритмом табл. 5 и схемой алгоритма, приведенной на рис. 70 вклейки в табл. 6 представлена программа работы генератора импульсов, написанная на языке ассемблера. В поле комментария дано объяснение порядка ее написания. Последней командой в программе является команда перехода JMP ADR, под воздействием которой осуществляется переход к адресу «Начало», то есть к началу выполнения цикла программы. Такое многократное повторение цикла программы называют **зацикливанием**. Как видим, режим зацикливания обеспечивает в нашем случае периодичность выдачи импульсов, что приводит к выполнению режима работы генератора импульсов.

Таблица 5

Шаг алгоритма	Выполняемые действия
1	Загрузить в регистр А число X
2	Уменьшить содержимое регистра А на 1
3	Если содержимое А не равно нулю, возвратиться к шагу 2, в противном случае перейти к шагу 4
4	Выдать в порт с адресом 0H содержимое А
5	Перейти к шагу 1, повторив этим выполнение цикла действий

Таблица 6

Метка	Операция	Операнд	Комментарий
Начало:	MVI	A, X	;Загрузить в регистр А число X
Время:	DCR	A	;Уменьшить содержимое А на 1
	JNZ	Время	;Если содержимое А не равно нулю, по- ;вторить уменьшение на 1, если равно ;нулю — продолжить выполнение ;программы
	OUT	0H	;Выдать содержимое А в порт 0H
	JMP	Начало	;Возвратиться к началу нового цикла

Программа табл. 6 написана в общем виде, и пока не будет задано значение X, она не работоспособна. В качестве учебного примера зададим X наибольшее возможное значение и вычислим, какова будет частота следования импульсов, выдаваемых в порт вывода, а затем зададим X наименьшее значение и снова подсчитаем частоту следования импульсов. Таким образом, мы определим диапазон возможного изменения частоты импульсов нашего программного генератора в зависимости от числа X, загружаемого в регистр А.

Из табл. 6 видно, какие команды используются в этой программе. Определив из колонки 6 табл. 1 количество тактов, нужных для выполнения каждой команды, а также, зная, что такт работы нашей микроЭВМ выполняется за 1 мкс (период времени между тактирующими импульсами Ф1 и Ф2), можно определить время выполнения каждой команды и, следовательно, подсчитать время, требуемое для выполнения цикла всей программы. Это время будет соответствовать периоду следования импульсов, выдаваемых программным генератором.

Составим следующую таблицу времени выполнения каждой команды (согласно колонке 6 табл. 1):

MVI = 7 тактов = 7 мкс
DCR = 5 —«— = 5 —«—
JNZ = 10 —«— = 10 —«—
OUT = 10 —«— = 10 —«—
JMP = 10 —«— = 10 —«—

Команды MVI, OUT и JMP используются в цикле один раз, а команды DCR и JNZ — многократно в зависимости от значения X. Занесем в регистр А наибольшее двоичное число 11111111, соответствующее десятичному 255. Подсчитаем, сколько времени уходит на команды, выполняемые один раз:

$$\underset{7}{\text{MVI}} + \underset{10}{\text{OUT}} + \underset{10}{\text{JMP}} = 27\ (\text{мкс}).$$

Аналогично определим время, затрачиваемое на выполнение многократно повторяемых команд:

$$\underset{5}{\text{DCR}} + \underset{10}{\text{JNZ}} = 15\ (\text{мкс}).$$

Эти команды будут повторяться 255 раз, поэтому их длительность выполнения составит:

$$15 \times 255 = 3825\ (\text{мкс})$$

Добавим время выполнения единичных команд:

$$3825 + 27 = 3852 \text{ (мкс)}.$$

Таким образом, через каждые 3852 мкс в порт вывода будет поступать импульс программного генератора. Переходя от периода времени T импульсов к их частоте F по формуле

$$F = \frac{1}{T},$$

получим:

$$F = \frac{1}{3852} = 259{,}8 \approx 260 \text{ (Гц)}.$$

Если же в регистр A занести число X = 1, то после первого же уменьшения содержимого A на 1 с помощью команды DCR число станет равным нулю и далее произойдет выполнение очередной команды OUT — в порт будет выдан импульс. Процесса многократного уменьшения содержимого A на единицу не будет, все команды программы станут использоваться в цикле по одному разу. Следовательно, общая продолжительность выполнения цикла программы будет соответствовать времени выполнения всех команд, а именно:

$$27 + 15 = 42 \text{ (мкс)}.$$

что обеспечит частоту F выдаваемых импульсов:

$$F = \frac{1}{T} = \frac{1}{42} \approx 38 \text{ (кГц)}.$$

Итак, помещая в регистр A число X в пределах от 1 до 255, можно получить на выходе порта импульсы с частотой от 38 кГц до 260 Гц, расположенные в диапазоне звуковых и ультразвуковых частот.

Если задаться целью получить программным путем радиолюбительский звуковой генератор, то нужно учесть, что импульсы на выходе порта имеют чрезвычайно малую длительность (миллионную долю секунды), поэтому, подключив к порту телефоны, невозможно будет услышать звуковой сигнал, так как мембрана телефона не успеет за столь короткий промежуток действия импульса совершить колебание.

Для получения звукового генератора обычного типа импульсы нужно расширить так, чтобы длительность импульса была равна паузе между ними (форма таких импульсов называется МЕАНДР). Это можно сделать как аппаратными, так и программными средствами. К аппаратным средствам относятся: а) подключение к выходу порта одновибратора (типа К155АГ1), который в ответ на короткий импульс, поступающий на его вход, выдает импульс нужной длительности; б) подключение к порту двоичного счетчика. В этом случае частота импульсов на выходе счетчика будет в два раза меньшей, чем частота импульсов, подаваемых с порта на его вход, а по длительности импульс будет равен паузе.

При программном выполнении звукового генератора нужно написать программу, обеспечивающую выдачу в порт (за каждую половину периода следования импульсов) логическую единицу или логический нуль, а в порту установить элемент (триггер), запоминающий смену этих логических уровней.

Если требуется получить импульсы с частотой следования меньшей чем 260 Гц, то эту задачу можно решить различными путями. Во-первых, для размещения числа X можно использовать не регистр, а пару регистров (BC, DE или HL), допускающую загрузку числа X = 65 355 вместо X = 255 и уменьшать это число до нуля при помощи команды DCX P, как это делалось в предыдущем случае; во-вторых, можно поместить число X в регистр, но каждый раз перед уменьшением числа на единицу вводить дополнительно в программу задержку по времени путем исполнения программы, написанной в табл. 6, помещая эту программу в нашу программу (такая операция называется **вложением**) и этим увеличивая общее время выполнения цикла программы. Если время окажется недостаточным, то можно произвести двухкратное вложение: вложить выполнение обеих команд в третью.

Возможны и другие способы программного получения низких частот генерируемых импульсов. Различные варианты творческих путей нахождения наилучшего

алгоритма решения поставленной задачи является самой характерной чертой программирования.

Учтя эту особенность, программирование часто считают искусством, сравнимым с творческим трудом поэта или композитора.

При однократном использовании программы генератора импульсов (см. табл. 6), когда в конце программы нет команды перехода JMP ADR, возвращающей исполнение программы к началу цикла, можно получить устройство для задержки времени, аналогичное электронному устройству, называемому реле времени или **таймер.**

Напишем программу для таймера, выполняющего задержку по времени на 1 секунду. Алгоритм таймера будем составлять по первому из указанных способов, то есть путем загрузки в регистровую пару BC числа X.

Загрузку выполним при помощи команды

LXI B,X

а уменьшение числа X на единицу произведем, используя команду

DCX B

Проверку того, стало ли число X в регистровой паре BC равным нулю, целесообразно выполнить при помощи команды ORA R, переслав предварительно в аккумулятор данные одного из регистров (например, регистра B):

MOV A, B
ORA C

Если результат окажется нулевой, то с помощью команды JMP ADR осуществится переход к выполнению команды OUT N, выдающей в порт импульс временной задержки, а если результат получится не нулевой, то произойдет возвращение к исполнению команды DCX P, уменьшающей число X на единицу.

Таким образом, программа таймера будет выглядеть так:

```
Начало: LXI  B,X    ;Загрузить в регистровую пару BC число X
Время:  DCX  B      ;Уменьшить содержимое BC на единицу
        MOV  A,B    ;Переслать содержимое регистра B в аккумулятор
        ORA  C      ;Сравнить содержимое регистра C и аккумулятора
        INZ  Время  ;При нулевом результате продолжать программу, в противном
                    ;случае вернуться к выполнению метки «Время».
        OUT  N      ;Выдать импульс временной задержки в порт N
```

Используя данные колонки 6 табл. 1, определим время выполнения каждой команды этой программы:

LXI = 10 тактов = 10 мкс
DCX = 5 « = 5 «
MOV = 5 « = 5 «
ORA = 4 « = 4 «
JNZ = 10 « = 10 «
OUT = 10 « = 10 «

Команды LXI и OUT исполняются в программе один раз и занимают время:

$$10 + 10 = 20 \text{ (мкс)}$$

Команды DCX, MOV, ORA и JNZ исполняются многократно и занимают в одном цикле время:

$$5 + 5 + 4 + 10 = 24 \text{ (мкс)}$$

Подсчитаем, сколько раз нужно повторить цикл исполнения многократных команд, чтобы получить время, равное 1 с.

Так как 1 с = 1 000 000 мкс, то

$$X = \frac{1\,000\,000}{24} = 41\,666(6)\ \ 41\,667 \text{ (раз)}.$$

Определяем точное время исполнения 41 667 циклов многократных команд

$$41\,667 \times 24 = 1\,000\,008 \text{ (мкс)}$$

или, прибавив время исполнения однократных команд,

$$1\,000\,008 + 20 = 1\,000\,028 \text{ мкс.}$$

Как видим, время выполнения цикла получилось на 28 мкс больше, чем нам нужно. В связи с этим уменьшим на 2 число X, заставив микропроцессор, выполняя программу, повторять цикл не 41 667 раз, а на 2 раза меньше

$$41\,667 - 2 = 41\,665 \text{ (раз)}$$

Определим время выполнения этих циклов

$$41\,665 \times 2 = 999\,960 \text{ (мкс)}$$

или, прибавив время выполнения однократных команд, получим:

$$999\,960 + 20 = 999\,980 \text{ (мкс)}$$

До 1 с нам недостает времени в 20 мкс

$$1\,000\,000 - 999\,980 = 20 \text{ (мкс)}$$

Это время можно получить, если ввести в программу пятикратное исполнение команды NOP, каждая из которых исполняется за 4 мкс

$$5 \times 4 = 20 \text{ (мкс)}$$

или ввести четырехкратное исполнение «пустой» команды MOV A,A, на которую затрачивается 5 мкс. Приняв первый вариант, напишем программу задержки по времени на 1 с:

```
Начало: LXI  B,41 665D ;Загрузить в регистровую пару BC десятичное число
                       ;41 665
Время:  DCX  B         ;Уменьшить содержимое BC на единицу
        MOV  A, B      ;Переслать содержимое B в аккумулятор A
        ORA  C         ;Сравнить содержимое C и аккумулятора
        JNZ  Время     ;При нулевом результате продолжить программу, если
                       ;нет, то возвратиться к метке «Время»
        NOP            ;Пустая операция
        NOP            ;—«—
        NOP            ;—«—
        NOP            ;—«—
        NOP            ;—«—
        OUT  N         ;Выдать импульс временной задержки в порт N
```

При трансляции исходной программы в объектную при помощи ассемблерующей программы в запись иногда вводят дополнительные команды управления ассемблером, называемые директивами ассемблера.

Директивы, подобно комментариям, не отображаются в объектных программах, и не вводятся в двоичном коде в память микроЭВМ, а лишь управляют процессом трансляции программы.

В качестве примера укажем директиву EQU, которая присваивает символическому имени или операнду определенное численное значение. Используя директиву EQU, программу получения импульсов с частотой 250 Гц, можно записать так:

X EQU 555D (присвоить X значение 555)

```
Начало: MVI A,X
Время:  DCR A
        JNZ  Время
        OUT  OH
        JMP  Начало
```

Директива EQU записывается в начале или в конце программы.

Помимо директивы EQU, часто пользуются такими директивами:

ORG — задает ассемблеру адрес ячейки памяти для первой команды транслируемой программы;

DB — указывает, какие данные должны быть записаны в память до начала выполнения программы;

END — используется для указания того, что исходная программа закончилась.

В заключение укажем, что программирование можно успешно освоить, даже не имея глубоких знаний в этой области и специальных навыков. Главное, как можно больше практиковаться в написании программ, а также внимательно и подробно разбираться в готовых, уже составленных программах, используя поле Комментария и таблицу системы команд МП—К580.

Дополнительные сведения о программировании можно узнать из литературы [1, 14, 24, 38, 48].

Язык бейсик

Язык бейсик является **диалоговым языком**, предназначенным для ведения разговора (обмена данными) между пользователем и микроЭВМ при помощи специальных **диалоговых технических средств** (клавиатуры и экранного дисплея, электропишущей машинки и др.). Поэтому изучать бейсик целесообразно только тогда, когда имеются эти диалоговые технические средства.

Следует помнить, что существует большое количество разновидностей (версий) языка бейсик, появившихся от того, что каждый новый тип компьютера фирма обычно наделяет своим бейсиком. Новые версии языка часто существенно отличаются от прежних. Поэтому, изучать нужно только ту версию, интерпретатор (переводчик на машинный язык) с которой заложен в память вашего компьютера.

Учитывая сказанное, приведем лишь самые общие, предварительные сведения о некоторых особенностях языка бейсик, присущих большинству его версий. Бейсик был разработан в 1960 году, как учебный язык для обучения программированию. Позже, в 1975 году, когда появились персональные компьютеры, бейсик стал использоваться в качестве рабочего языка. Он оказался очень удобным для ведения диалога с компьютером и поэтому стал основным языком всех персональных компьютеров.

Учитывая широкое распространение бейсика, знакомство с ним представляет известный интерес.

Название языка бейсик произошло от сокращения английских слов, означающих: «многоцелевой язык символических инструкций для начинающих».

Программу на этом языке пишут в виде строк, подписывая одну строку под другой. В отличие от языка ассемблера, при нумерации строк можно нумеровать строки подряд или с произвольным интервалом.

Обычно принято нумеровать строки с интервалом в 10, записывая:

```
10 . . . . .
20 . . . . .
30 . . . . .
```

Такой вид записи позволяет делать вставки в промежутки между номерами (с 11 по 19, с 21 по 29 и др.) для вписывания дополнительных строк в программу.

Нумеруя строки с желаемым интервалом, нужно следить за тем, чтобы номера строк не повторялись и были записаны в возрастающем порядке.

После номера в строке пишется та «символическая инструкция», которая определяет характер предстоящих действий над некоторыми данными. В литературе вследствие неустоявшейся терминологии эту инструкцию иногда называют командой [41], а иногда — **оператором** [14, 29]. В дальнейшем изложении мы будем пользоваться термином «оператор». В строке можно записывать не один, а несколько операторов. В этом случае они отделяются друг от друга двоеточием и называются составным оператором.

Все, что мы хотим сообщить в операторе, необходимо кодировать с помощью букв, цифр и специальных знаков, составляющих алфавит (азбуку) языка программирования.

Алфавит бейсика состоит из:

1) прописных латинских букв A, B, C, ...;
2) прописных русских букв А, Б, В, Г, ...;
3) цифр Ø123456789 (нуль перечеркивают, чтобы отличить от буквы О);
4) специальных символов:

. — точка;
, — запятая;
: — двоеточие;

; — точка с запятой;
(— левая скобка;
) — правая скобка;
␣ — пробел;
„ — кавычки;
! — восклицательный знак;
? — вопросительный знак;
+ — плюс;
— — минус;
* — звездочка (знак умножения);
/ — знак деления;
∧ — знак возведения в степень;
= — равно, знак присваивания;
> — больше;
< — меньше.

Зарубежные компьютеры используют в качестве символов бейсика еще дополнительно строчные латинские буквы, а также некоторые другие символы.

Символы алфавита бейсик должны располагаться в строке оператора по определенным правилам и в определенной последовательности.

Помимо символов, в операторе можно встретить сокращенные английские слова (например, CLS, LET, PRINT), похожие на мнемонические слова операций в языке ассемблера, и указывающие на выполняемые действия. Эти слова часто называют именем оператора и располагают правее номера строки. После имени следует текст оператора.

Приведем пример бейсик-программы и поясним ее:

```
10 CLS
20 LET   A = 3
30 LET   B = 2
40 LET   C = A + B
50 PRINT C
60 END
```

Эта программа содержит шесть строк, по одному оператору в каждой строке.

Строка с номером 10 содержит правее номера имя оператора CLS, означающее действия по очистке экрана дисплея от ненужного нам текста. В строке под номером 20 помещен оператор с именем LET, указывающий на то, что символу A присваивается значение десятичной тройки. Это значение (A = 3) помещается в ячейку памяти для хранения. Аналогично, в строке под номером 30 имя оператора LET присваивает символу B значение двойки и помещает эту двойку в другую ячейку памяти. В следующей строке с номером 40 в соответствии с названием оператора LET из памяти извлекаются значения символов A и B, производится операция их сложения, после чего символу C присваивается результат сложения и этот результат помещается в третью ячейку памяти.

Очередная строка, имеющая после номера 50 имя оператора PRINT, предназначена для вывода на экран дисплея результата сложения. При этом значение символа C (пятерка) извлекается из ячейки памяти и отображается на экране в виде числа 5. Последняя строка под номером 60 сообщает при помощи имени оператора END об окончании исполнения программы.

Приведем краткое описание наиболее часто используемых операторов языка бейсик, помня при этом, что под словом оператор здесь понимается его название (имя, тип).

Оператор INPUT позволяет приостановить выполнение программы и ввести данные с помощью клавиатуры. Если данные состоят из нескольких чисел, то числа отделяются друг от друга запятой. Например, по ходу выполнения программы прогноза погоды, в какое-то место программы нужно ввести два числа. Первое число X — о температуре (X = 20), а второе Y — о барометрическом давлении (Y = 745). Для этого программист вводит в определенную строку программы оператор INPUT

```
140 INPUT X,Y
```

Когда микроЭВМ, выполняя программу, встретит оператор INPUT, то она приостановит выполнение программы, выдаст на экран знак вопроса (?) и перейдет

в режим ожидания ввода данных. Пользователь должен нажать клавиши 2 и 0, вводя этим число 20 (температура), затем нажать клавишу со знаком запятой и три клавиши 7, 4 и 5, означающие ввод числа 745 (барометрическое давление). После чего следует нажать клавишу ВК. При этом числа 20 и 745 будут введены в нужное место программы и микроЭВМ продолжит ее выполнение.

С оператором PRINT мы уже встречались, рассказывая о программе, выполняющей вывод на экран результата сложения чисел. Помимо этого действия, оператор PRINT может выполнять и ряд других операций. Он используется для следующих целей:

1) вывода на экран или на печать результатов вычислений;
2) выполнения вычислений;
3) вывода на экран или на печать текста, указанного в кавычках и помещенного правее оператора;
4) пропуска строки.

Поясним сказанное примером написания оператора PRINT

```
110  PRINT  3 + 2
120  PRINT  «Сложение чисел»
130  PRINT
```

После выполнения строки под номером 110 на экране дисплея или на листе бумаги появится число 5. После выполнения строки с номером 120 на экран будет выведен или отпечатан на бумаге текст: СЛОЖЕНИЕ ЧИСЕЛ. После выполнения строки с номером 130 произойдет пропуск строки.

Оператор GOTO N является оператором перехода. Он применяется для того, чтобы перейти к выполнению оператора, имеющего номер N (номер N пишется правее имени оператора GOTO)

Пример:

```
110 GOTO 200
```

Выполнив строку 110, программа перейдет к выполнению строки 200.

Операторы COSUB N и RETURN тоже являются операторами перехода. С их помощью переходят к выполнению подпрограммы, начинающейся со строки N. При этом подпрограмма должна обязательно заканчиваться оператором RETURN, который обеспечивает возвращение в то место основной программы, откуда произошел вызов подпрограммы.

Очень часто в программах применяется оператор условного перехода IF X THEN Y. Его действие заключается в проверке условия. Если X истинно (равно единице), то выполняется строка, имеющая номер Y, а если X ложно (равно нулю), то выполняется следующая строка программы.

Оператор POKE X,Y позволяет записать в ячейку памяти с адресом X данные Y.

Оператор OUT X,Y позволяет выдать в порт с номером X данные Y

Оператор STOP приостанавливает выполнение программы.

Оператор PLOT X,Y,Z служит для отображения на экране светящейся (если $Z = 1$) или погашенной (если $Z = 0$) точки, имеющей координаты X и Y. При этом для большинства экранов величина X бывает в пределах от 0 до 128, а Y — от 0 до 64.

Оператор LINE X,Y служит для получения изображения прямой линии на экране с конечными координатами X и Y. Начальные координаты, а также вид линии (засветка или гашение) задаются при помощи оператора PLOT

Пример:

```
10  PLOT  10, 15, 1
20  LINE  40, 80
30  STOP
```

После выполнения этой программы на экране появится светящаяся линия с координатами: начальными 10, 15 и конечными 40, 80.

Помимо перечисленных операторов, в языке бейсик имеются невыполняемые операторы или операторы описания. К таким операторам относятся, например, оператор REM, используемый для вставки в текст программы комментария. В языке ассемблера для комментария отведено крайнее правое поле, а в бейсике комментарий пишется в строке после оператора REM. Когда интерпретатор встречает оператор REM, то он не принимает его во внимание, не замечает, про-

пускает его. Если текст комментария не помещается в одной строке, то следующую строку тоже именуют REM и продолжают писать в ней текст комментария.

Помимо операторов, в строках бейсик-программы можно встретить инструкцию в виде директивы. Директива, в отличие от оператора, предписывает действия не с данными, а с самой программой, управляя ею или видоизменяя.

Опишем некоторые директивы языка бейсик.

Директива NEW стирает в памяти предыдущую программу и подготавливает интерпретатор для ввода новой программы с клавиатуры.

Директива RUN N вызывает исполнение программы, начиная с номера строки N. Если номер строки не указан, то программа выполняется со строки с наименьшим номером.

Директива LIST используется для воспроизведения на экране или печати текста всей программы, группы строк программы или одной строки. Например, директива LIST 20 напечатает только строку 20, LIST 40—60 напечатает строки 40, 50 и 60, а директива LIST напечатает весь текст программы.

Директивы, в отличие от оператора, пишут в программе без указания номера строки.

Пример:

```
     NEW
10   LET    X = 1
20   LET    Y = 3
30   PRINT  X,Y
     RUN
```

Рассмотрим правила написания текста оператора. Этот текст иногда называют выражением или аргументом. Самый простой вид имеют выражения при выполнении математических вычислений. В этом случае они почти не отличаются от обычной записи. Нужно только соблюдать правило: все символы выражения записывать в одну строку. Дело в том, что экран дисплея или электрифицированная пишущая машинка не могут воспроизводить (печатать) символы, расположенные выше или ниже строки. Если машинистка, печатая текст на обычной пишущей машинке, может вручную подкрутить валик и отпечатать 3^2, то подобную запись нельзя получить на экране дисплея или на листе бумаги печатающего устройства. Поэтому приходится видоизменять написание некоторых привычных математических обозначений. Возведение в степень, например, описывается с помощью символа, напоминающего острие стрелки, направленное вверх ($\wedge$). Пользуясь этим символом, 3^2 записывается: $3\wedge 2$. Чтобы отличить знак умножения $\times$ от буквы X употребляют символ в виде звездочки ($*$). При этом точку использовать нельзя, так как с ее помощью отделяют в дроби целое число от дробного, то есть пишут 3.14, вместо обычного 3,14. Запятую же используют для отделения одного числа от другого. Запись 3,14 означает теперь два числа: первое число 3, а второе 14. Знак деления имеет вид наклонной линии /, потому что двоеточие занято для разделения операторов.

Для определения старшинства операций и порядка их выполнения применяют скобки. Выражения внутри скобок выполняются в первую очередь, а затем взятая в скобки величина используется для дальнейших вычислений. Если нет скобок, то операции выполняются в такой последовательности:

1) вычисление функций;
2) возведение в степень;
3) умножение и деление;
4) сложение и вычитание;
5) выполнение логических операций.

Приведем несколько примеров написания программы на языке бейсик для решения математических задач.

Изучая язык ассемблера, мы писали на этом языке программу сложения двух чисел (3 + 2). Эта программа (см. табл. 2.14) занимала восемь строк. На языке бейсик она размещается в одной строке и выглядит очень просто

```
10 PRINT 3 + 2
```

Программа решения более сложных математических задач тоже выглядит довольно просто и понятно.

Например, выражение

$$\frac{ax^2+bx+c}{d-3}$$

для решения на языке бейсик записывается так:

```
20 PRINT(A * X ^ 2 + B * X + C) / (D - 3)
```

Несколько слов о функциях, входящих в состав выражения языка бейсик.

При обычном решении задач без помощи микроЭВМ часто приходится прибегать к различным справочникам и таблицам для нахождения всевозможных функций (например, тригонометрических). А вот, имея микроЭВМ, можно воспользоваться готовыми стандартными подпрограммами функций (библиотекой подпрограмм функций), хранящихся в памяти микроЭВМ. Для обозначения таких функций в бейсике используют трехбуквенные английские наименования. Причем аргумент функции X заключается в скобки.

Помимо общеизвестных тригонометрических функций SIN (X), COS (X) и др., в бейсике применяется большое количество (несколько десятков) и других функций. Наиболее часто пользуются такими функциями:

EXP(X) — для вычисления экспоненты (e^X);

SQR(X) — для вычисления квадратного корня ($\sqrt{X}$);

RND(X) — для выбора случайного числа в пределах от 0 до 1;

INT(X) — для выделения целой части числа, которое сводится к округлению его до ближайшего целого значения. Например:

INT(4.2) — принимается равным 4;
INT(6.8) — —«— 7;
INT (5) — —«— 5.

Функция SPC(X) позволяет вставить в строку X пробелы. Например, строка программы

```
10 PRINT «МикроЭВМ типа»: SPS(6): «есть в продаже»
```

будет отпечатана так:

МИКРО—ЭВМ ТИПА ____ ____ ЕСТЬ В ПРОДАЖЕ, где (__) пробел.

Функция PEEK(X) выдает десятичное число, равное содержимому ячейки памяти с адресом X.

Функция INP(X) выдает десятичное число, равное содержимому порта ввода с номером X.

Функция USR(X) позволяет перейти к выполнению программы на языке ассемблера, которая хранится в памяти с начальным адресом X. Эта функция очень важная. Дело в том, что не все программы можно и нужно писать на языке бейсик. Встречаются программы (или фрагменты программ), которые целесообразно писать на языке ассемблера. Например, программа игры в «крестики—нулики» должна завершаться музыкальным фрагментом, исполняемым в честь победителя. Из-за временных ограничений, накладываемых на интерпретатор, и по ряду других причин написать музыкальную часть программы на бейсике очень сложно, в то время как на ассемблере это выполняется сравнительно просто. Наличие функции USR позволяет в нужный момент перейти на программу ассемблера. При этом следует не забывать в конце программы ассемблера поставить команду RET, после которой осуществляется автоматическое возвращение в программу бейсик.

В заключение укажем, что существуют готовые программы, позволяющие быстро (в течение одного-двух часов) изучить бейсик. При этом обучение становится еще более эффективным, если в этих программах использован метод **«меню»**, заключающийся в том, что микроЭВМ выдает пользователю на выбор (отсюда название «меню») серию различных советов, рекомендаций, наводящих вопросов и указаний.

Для расширения своих знаний в области программирования на языке бейсик читатель может воспользоваться литературой [29, 56, 58, 62].

3. Терминологический словарь

Автокод — разновидность простейшего *языка программирования низкого уровня* [1], то же, что и язык *ассемблера.*

---

[1] Курсивом набраны термины, поясняющиеся в данном словаре.

Автоматизация программирования — совокупность средств и действий, направленных на облегчение и ускорение процесса *программирования.* Основой автоматизации является использование *языков программирования,* близких к естественному человеческому восприятию (см. *Язык программирования высокого уровня*). Кроме того, в целях автоматизации разработано и применяется большое количество вспомогательных (сервисных) *программ,* составляющих так называемую *операционную систему.*

Адрес — номер *ячейки памяти* или *устройства ввода—вывода данных.* Обычно *адрес* располагается на адресной шине в виде двоичного числа, которое можно прочесть при помощи светодиодных или буквенно-цифровых индикаторов, подключаемых к этой шине. Адрес можно записать на бумаге или отобразить на экране *дисплея* в виде двоичного числа (например, 0000101000111110), шестнадцатеричного числа (например, A03E) или символического слова (например, SUMB) условно обозначающего адресное число.

Аккумулятор — один из *регистров общего назначения микропроцессора,* обозначаемый буквой А. Он является основным рабочим *регистром,* участвующим во многих *операциях.* Например, при сложении двух чисел одно из них должно находиться в А. Результат вычисления также размещается в А. При обмене *данными с внешними устройствами* они засылаются сперва в А, потом уже в то место, которое указано в *программе. Содержимое* А можно сдвигать влево или вправо при помощи специальных *команд. Регистр* А получил свое название благодаря тому, что в первых ЭВМ он играл роль накапливающего сумматора, который накапливал (аккумулировал) числа, поступающие в ЭВМ для сложения.

Алгол — *язык программирования высокого уровня,* используемый для написания *программ,* выполняющих большое количество вычислений над небольшим количеством *данных* (инженерные и научно-технические задачи).

Алгоритм — точное описание некоторого процесса, инструкция по его выполнению. Алгоритм указывает, какие действия и в какой последовательности нужно выполнить над исходными *данными,* чтобы решить поставленную задачу.

АЛУ — см. *Арифметическо-логическое устройство.*

Алфавитно-цифровое печатающее устройство (АЦПУ) — технический аппарат, осуществляющий автоматическое печатание на бумаге букв, цифр, разделительных знаков и некоторых специальных символов. Входит в комплект технических средств ЭВМ для вывода *данных* в виде печатного текста, таблиц или графиков. Существуют различные типы *АЦПУ,* основанные на электромеханическом, электрохимическом, ксерографическом и других принципах действия. Для *микроЭВМ* обычно применяют простейшие *АЦПУ* портативного типа и электромеханического принципа действия в виде *электрифицированных пишущих машинок.* В переводной литературе АЦПУ часто называют *принтером.*

Аппаратные средства — совокупность технических средств *(микропроцессор, память, устройство ввода—вывода,* блок питания, пульт управления и др.), входящих в состав *микроЭВМ.*

Арифметическо-логическое устройство (АЛУ) — составная часть *микропроцессора, выполняющая операции* по арифметической и логической обработке данных.

Ассемблер — 1. Название *языка программирования низкого уровня,* на котором пишут *программы* для *микроЭВМ,* отличающегося тем, что действие *(операция)* каждой *команды* условно обозначается мнемоническим сокращением английского слова. 2. *Программа* или техническое средство, выполняющее перевод *(трансляцию) программы,* написанной на языке *ассемблера,* в *машинный код.*

Ассемблирование — процесс перевода *(трансляции) программы,* написанной на *языке ассемблера,* в *машинный код.* Этот перевод может быть выполнен при помощи *программы,* называемой *ассемблер,* или вручную с помощью технических средств. Перевод осуществляется один к одному, когда каждая мнемоническая команда программы заменяется *машинным двоичным кодом.*

АЦП — сокращенное название аналогово-цифрового преобразователя, который преобразовывает непрерывное, аналоговое значение какой-либо физической величины (например, напряжения) в дискретный цифровой код.

АЦПУ — см. *Алфавитно-цифровое печатающее устройство.*

Байт — восьмиразрядное двоичное число, состоящее из восьми *бит* (1 байт = = 8 бит). Иногда *байт* рассматривается как единица обмена *данными* между отдельными устройствами *микроЭВМ.*

Бейсик — *язык программирования высокого уровня.*— Универсальный язык для работы с *микроЭВМ* в *диалоговом* режиме. Относительно прост для освоения. Широко применяется при составлении *программ* для *микроЭВМ.*

Библиотека программ — набор отлаженных стандартных *программ* для многократного использования, составляемых с учетом определенного *микропроцессора* или *микроЭВМ.* Набор включает сотни *программ,* написанных на различных *языках программирования,* доступных для *микроЭВМ* данного типа.

БИС — сокращенное название большой *интегральной схемы,* отличающейся от обычной интегральной схемы большим количеством (сотни тысяч, иногда миллионы) составляющих элементов. БИС широко используется в современной электронной аппаратуре — телевизорах, электронных часах, *микрокалькуляторах, компьютерах* и др.

Бит — разряд двоичного числа или двоичная цифра (0 или 1). Так как данные представляются в двоичных числах, то бит является наименьшей единицей количества *данных.*

Блок-схема алгоритма — условное графическое изображение процесса реализации *алгоритма* на *микроЭВМ* в виде совокупности блоков, соединенных между собой линиями со стрелкой на конце. В каждом блоке приводится описание *операции,* выполняемой блоком. Выполнение *операции* обозначается различной формой блока. В настоящее время блок-схема алгоритма, согласно ГОСТ, именуется *схемой алгоритма.*

Бод — единица измерения скорости передачи *данных* по линиям связи, соответствующая одному *биту* в секунду.

Вектор прерывания — трехразрядный *двоичный код,* используемый *микропроцессором* для формирования начального *адреса* одной из восьми возможных подпрограмм обслуживания запроса на *прерывание.* Обычно *вектор* обозначается отрезком прямой со стрелкой на конце, выражая численное значение и направление математической величины. Здесь же понятие *вектора* равнозначно понятию указателя. *Вектор прерывания* указывает *микропроцессору,* какой *адрес* из восьми возможных нужно использовать, чтобы перейти на выполнение *подпрограммы прерывания.*

Видеотерминал — устройство для ввода, вывода и отображения *данных* на телевизионном экране. Состоит из *дисплея* и *клавиатуры.* Входит в комплект *внешних устройств микроЭВМ.*

Внешнее устройство — см. *Периферийное оборудование.*

Входное устройство — см. *Устройство ввода данных.*

Высокоимпедансное состояние — одно из состояний *тристабильного выхода,* при котором он имеет бесконечно большое сопротивление и отключается от схемы.

Выходное устройство — см. *Устройство вывода данных.*

Данные — сведения, сообщения, известия, факты, необходимые для какого-нибудь вывода или решения, выраженные в таком виде, который позволяет пересылать, обрабатывать и хранить данные при помощи технических средств. В микропроцессорной технике *данные* представляются в виде двоичных чисел, выраженных низким (0) или высоким (1) электрическими уровнями, позволяющими вводить *данные* в *микропроцессор,* обрабатывать их, выводить во *внешнее устройство* и пересылать по линии связи. Вместо слова *данные* часто употребляют слово *информация.*

Двоичный код — *код,* содержащий два кодовых значения (символа), например 0 и 1.

Дешифратор — техническое устройство, позволяющее преобразовать (расшифровать) *код* числа, поступившего на его входы, в сигнал, выдаваемый на один или несколько его выходов. Например, на входы подают двоичный *код* числа, а на одном из выходов получают сигнал, соответствующий десятичному значению этого числа.

Диагностическая программа — служит для определения местонахождения и пояснения причин неисправной работы технических средств *микроЭВМ* или ошибок в *программе.*

Диалоговое взаимодействие — поочередный обмен *данными* между человеком и *микроЭВМ* с помощью *клавиатуры* и телевизионного экрана. В некоторых системах *диалогового взаимодействия,* рассчитанных на опытных *пользователей,* управления диалогом осуществляется преимущественно человеком, а в других случаях,

когда диалог происходит с малоподготовленным *пользователем,* управление берет на себя *микроЭВМ,* своевременно выдавая нужные советы, подсказки и дополнительные наводящие вопросы. Диалоговый режим иногда в литературе называют *интерактивным режимом.*

Динамическое запоминающее устройство — запоминающее устройство, использующее режим регенерации в виде многократной перезаписи данных для их сохранения.

Диспетчер — *программа* и технические средства, регламентирующие последовательность включения различных *программ* в соответствии с их наименованиями, важностью (приоритетом) и указаниями, получаемыми от *управляющей программы.*

Дисплей — устройство для вывода и отображения *данных* в виде слов, чисел и графиков на телевизионном экране. Существует большое количество различных типов *дисплеев.* В некоторых из них телевизионный экран заменяют буквенно-цифровыми индикаторами или плоским, матричным экраном. В целях упрощения и удешевления дисплея его иногда заменяют бытовым телевизором, путем добавления к телевизору специальной электронной приставки.

Драйвер — 1) *Программа,* реализующая обмен порциями *данных* между основной *памятью микроЭВМ* и внешним устройством. 2) Техническое устройство в виде усилителя мощности электрического сигнала.

Загрузчик — *программа,* которая объединяет отдельные *программы,* полученные в результате перевода *(трансляции),* размещает их в нужной области *памяти,* контролирует начальный и конечный *адрес* программ и выполняет ряд других функций по указанию *управляющей программы.*

Запись данных — режим работы *запоминающего устройства,* в процессе которого осуществляется занесение *данных* в *ячейки запоминающего устройства.*

Запоминающее устройство (ЗУ) — техническое средство, реализующее *память.* Предназначено для записи, хранения и считывания *данных.*

Запоминающий элемент — часть *запоминающего устройства,* служащая для записи, хранения или считывания наименьшей единицы *данных* (0 или 1).

Захват шин — режим работы *микроЭВМ,* при которой сигнал из *внешнего устройства* подается на вывод (ЗХ) *микропроцессора* для запроса разрешения использовать общие шины *адреса* и *данных.* При удовлетворении этого запроса *микропроцессор* отключает свои шины *адреса* и *данных* от общей шины и выдает сигнал (ПЗХ), подтверждающий разрешение захвата шин.

Зацикливание — многократно повторяющееся выполнение какого-либо участка (цикла) *программы.* Часто причиной зацикливания является ошибка в *программе,* но иногда его организовывают специально для решения определенных видов задач.

Защита памяти — совокупность аппаратных и программных средств *микроЭВМ,* обеспечивающих сохранение *данных* одной задачи от возможного разрушающего действия других задач, а также предотвращения случайной записи *данных* в занятые *ячейки памяти.*

ЗПВВ — сокращенное обозначение управляющего сигнала, означающего запись (выдачу) *данных* во *внешнее выводное устройство.*

ЗПЗУ — сокращенное обозначение управляющего сигнала означающего запись *данных* в *запоминающее устройство.*

ЗУ — см. *Запоминающее устройство.*

Индентификатор — символы, используемые для обозначения *данных* (массивов, структур, меток и др.) в *языках программирования.*

Имитатор — см. *Эмулятор.*

Индикаторное устройство — специализированные элементы, обеспечивающие наглядное (визуальное) отображение *данных,* выводимых из *микропроцессора.* Существует большое количество различных типов индикаторов. При работе с *микроЭВМ* чаще всего пользуются светодиодами, буквенно-цифровыми и матричными индикаторами, а также телевизионным экраном.

Инициализация — установка в начальное (исходное) состояние различных технических устройств — счетчиков, *регистров, триггеров* и др.

Интегральная схема — миниатюрное электронное изделие, выполняющее функцию сложных электронных схем (усилителей, генераторов и др.), состоящее из десятков (сотен) транзисторов и резисторов, соединенных между собой и занимающих объем менее кубического миллиметра.

Интерактивный режим работы — см. *Диалоговое взаимодействие.*

Интерпретатор — *программа* или техническое средство, служащее для последовательного (пошагового) перевода каждого предложения *(оператора) программы,* написанной на *языке высокого уровня* в серию *машинных двоичных кодов,* которые сейчас же выполняются на *микроЭВМ.* После этого следует новый шаг перевода и новое исполнение переведенного *машинного кода.*

Интерфейс — совокупность правил, технических и программных средств, допускающих согласованное соединение (сопряжение) по временным и электрическим параметрам отдельных устройств *микроЭВМ при их совместной работе.*

Информатика — 1) Наука, изучающая законы, методы и способы накопления, передачи и обработки *информации* при помощи ЭВМ. 2) Область человеческой деятельности, обслуживающая науку, технику, производство, быт и многие другие стороны общественной жизни новыми видами информационной технологии на основе применения современной электронной вычислительной техники.

Информация — одно из наиболее общих, широких и недостаточно точно описанных понятий науки, означающих чаще всего некоторые *данные,* сведения, знания. *Информацию* иногда трактуют как смысл, внутреннее содержание, которое приобретает человек, используя *данные.* В книге вместо слова *информация* авторы чаще пользуются словом *данные,* понимая под этим цифровые данные, буквенный текст, графические или телевизионные изображения, различные виды механических действий, физиологических состояний и других сообщений, условно представленных в виде цифрового *двоичного кода,* пересылаемого по шине *данных* в *микропроцессор* для программной обработки.

Исходная программа — *программа* на одном из *языков программирования,* которая перед загрузкой и выполнением на *микроЭВМ* должна быть переведена в *объектную программу.*

Квитирование — метод взаимодействия *микроЭВМ* с *периферийным оборудованием,* при котором между ними осуществляется обмен управляющими сигналами и сигналами состояния с целью взаимной синхронизированной работы. Метод передачи *данных* с квитированием позволяет согласовать скорости выполнения *операции* в быстродействующих *микропроцессорах* и медленнодействующих *устройствах ввода—вывода данных.*

Клавиатура — система клавишей типа пишущей машинки с надписью над каждой клавишей.

Клавишная панель — панель (пульт), содержащая переключатели, кнопки и клавиши, используемые при работе с *микроЭВМ* для ввода *программы, данных* и управления режимом работы отдельных технических устройств.

Кобол — *язык программирования высокого уровня,* используемый при составлении *программы* для решения бухгалтерско-экономических задач. Применяется в основном при работе с большими и миниЭВМ.

Код — набор правил, раскрывающий способ представления *данных* при помощи условных знаков (кодового набора символов).

Команда — набор двоичных знаков (0 или 1), воспринимаемый *микропроцессором* как приказ для выполнения определенной *операции.* Обычно команда состоит либо из *кода операции,* либо из *кода операции* и *данных,* либо из *кода операции* и *адреса* того места, где хранятся *данные.*

Компилятор — *программа* или техническое средство, выполняющее перевод *(трансляцию)* программы, написанной на *языке программирования высокого уровня* в *машинный двоичный код.* При этом в отличие от *интерпретатора* происходит лишь перевод *программы* без ее немедленного выполнения на *микроЭВМ.*

Компьютер — так в переводной литературе называют электронную вычислительную машину (ЭВМ). Термин *«компьютер»* начинает применяться и на страницах нашей печати, означая при этом в отличие от ЭВМ, электронную машину, выполняющую преимущественно логические, интеллектуальные операции, а не узко вычислительные действия (см. *Персональный компьютер).*

Контроллер — *микропроцессор* или *микроЭВМ,* предназначенные для управления какой-либо системой, *устройством* или *процессом.*

Кросс-программное обеспечение — совокупность взаимосвязанных служебных *программ (трансляторы, редакторы, эмуляторы* и др.), предназначенных для создания рабочих *программ* на *микроЭВМ* (или большой ЭВМ), отличающейся от той, для которой составляется рабочая *программа.*

Курсор — движущийся знак (метка), воспроизводимый на экране *дисплея* для указания места, где будет высвечиваться очередной символ или выполняться другая *операция* редактирования текста.

Листинг — печатная копия *программы*, выдаваемая *транслятором* с помощью *электрифицированной пишущей машинки*, а также выдача этой копии на экран *дисплея*.

Логический элемент — простейшее в функциональном отношении устройство, выполняющее определенную логическую операцию (И, ИЛИ, НЕ) над сигналами, согласно правилам алгебры логики.

Магистраль — см. *Шина*.

Маркер — условный знак, наносимый на магнитную ленту, барабан, перфоленту или другой носитель *данных* для выполнения некоторых служебных функций. Например, поиска начала или конца нужной зоны данных, распознавания характера данных и др.

Массив — упорядоченный набор *данных* одинакового типа. Например, массив целых чисел, массив логических значений и др.

Машинное слово — набор *битов*, рассматриваемый как единое целое, которое *микропроцессор* обрабатывает за один шаг. Обычно машинное слово кратно *байту* (восьми битам) и занимает одну *ячейку памяти*.

Машинный язык — см. *Язык машинный*.

«Меню» — один из способов общения *пользователя* с *микроЭВМ*, при котором микроЭВМ выводит на экран *дисплея* набор («меню») вопросов и различных вариантов ответов или решений, которые *пользователь* может выбрать по своему желанию.

Метка — условное обозначение начала *подпрограммы*.

Микрокалькулятор — малогабаритное электронное устройство, служащее для повседневных математических вычислений. Имеет клавиши, позволяющие вводить нужные числа и выполнять требуемые операции (сложение, вычитание, умножение и др.). Вводимые числа и результат вычислений отображается на многоразрядном цифровом индикаторе. Часто в состав устройства входит источник питания (батарея, аккумулятор) или блок питания, допускающий питание от сети переменного тока.

Микроконтроллер — см. *контроллер*.

Микропроцессор — миниатюрный электронный прибор, выполненный в виде *БИС* и предназначенный для обработки двоичных цифровых *данных*. Обработка производится с большой скоростью по *программе*, заложенной в *память*. Изменение *программы* позволяет решать различные вычислительные и логические задачи, а также осуществлять управление режимом работы автоматических устройств.

МикроЭВМ — конструктивно законченное электронное изделие, построенное на основе *микропроцессора, памяти, устройства ввода—вывода данных*, пульта управления и источника питания.

Мини-ЭВМ — электронная вычислительная машина, занимающая по своим показателям среднее место между *микроЭВМ* и большой ЭВМ. Часто она выполняется не на *БИС*, а на отдельных дискретных логических микросхемах малой и средней степени интеграции. Мини-ЭВМ находит широкое применение для научно-технических и планово-экономических расчетов невысокой сложности, а также для автоматизации экспериментальных исследований и управления технологическими процессами. Последние годы все чаще заменяется микроЭВМ.

Монитор — 1) *Программа* и технические средства, осуществляющие взаимодействие человека и *микроЭВМ*. *Монитор* является частью *управляющей программы микроЭВМ*. В зависимости от *системы команд микропроцессора*, режима работы *микроЭВМ*, состава *устройства ввода — вывода данных*, а также объема внешней *памяти монитор* способен выполнять большое количество различных управляющих действий, реализуемых *пользователем* путем нажатия функциональных *клавиш*. При этом *монитор* может выводить на экран *дисплея* или на печать *данные* из некоторой области *памяти*, загружать *данные* в некоторую область *памяти*, изменять состав и порядок работы устройств *ввода—вывода данных, сравнивать заданные области памяти ОЗУ* и *ПЗУ* и т. д. *Монитор*, используемый в *микроЭВМ* ограниченного состава, обычно выполняет лишь несколько простейших функций или даже только одну, а более сложный *монитор*, применяемый в современных микропроцессорных системах, способен реализовывать большое

количество (десятки) различных операций, обеспечивающих удобство и оперативность взаимодействия человека и ЭВМ. 2) Вспомогательное контрольное оборудование, служащее для наблюдения за состоянием системы и определения отклонений от заданной нормы. Например, телевизионный *монитор* для проверки качества телевизионного изображения, медицинский монитор для контроля деятельности сердца, кровяного давления, частоты пульса и др.

Мультиплексорная система — система передачи *данных* от нескольких устройств по одному каналу связи.

Нибла — см. *Тетрада.*

НМД — сокращенное название накопителя на магнитном диске, представляющего собой *запоминающее устройство,* в котором носителем *данных* является магнитный диск.

НМЛ — сокращенное название накопителя на магнитной ленте, представляющего собой *запоминающее устройство,* в котором носителем *данных* является магнитная лента.

Общее программное обеспечение — совокупность взаимосвязанных *программ,* предназначенных для управления аппаратными средствами, автоматизации создания *программ* и облегчения техники обслуживания *микроЭВМ.* К *программам общего программного обеспечения* относятся: *трансляторы, редакторы, загрузчики, мониторы, супервизоры,* наладочные тесты, *диагностические программы* и др.

Объектная программа — *программа, которую можно поместить непосредственно в* память микроЭВМ и выполнить без каких-либо изменений. В *объектной программе команды* представляются как последовательность двоичных чисел *(двоичных машинных кодов).*

ОЗУ — см. *Оперативное запоминающее устройство.*

Операнд — *данные,* над которыми выполняется математическая или логическая *операция.*

Оперативное запоминающее устройство (ОЗУ) — *запоминающее устройство,* непосредственно связанное с *микропроцессором* и предназначенное для оперативной записи, хранения и считывания *данных* во время выполнения арифметико-логических операций.

Оператор — 1) Обобщенная *команда* (предписание) на *языке программирования высокого уровня.* 2) Представитель персонала, обслуживающего ЭВМ.

Операционная система (ОС) — комплект *управляющих* и обслуживающих *программ,* предназначенных для *автоматизации программирования* и обеспечения максимальной эффективности работы *микроЭВМ,* путем автоматизации управления ее режимами и предоставления *пользователю* набора определенных услуг.

К программам ОС относятся: *транслятор, компилятор, интерпретатор, редактор, загрузчик, монитор, супервизор* и многие другие. Часто функции программ бывают настолько взаимосвязаны и взаимообусловлены, что их не всегда удается четко разграничить и обозначить.

Краткие пояснения наиболее важных программ ОС приводятся в настоящем словаре.

Операция — совокупность действий, предусмотренных одной *командой микропроцессора.*

ОС — см. *Операционная система.*

Отладка программы — процесс обнаруживания и исправления ошибок в *программе* путем ее прогона на *микроЭВМ* с использованием искусственно составленных или истинных *данных* и применения специальных *диагностических программ.*

Память — часть аппаратных средств *микроЭВМ* в виде *запоминающего устройства,* служащего для записи, хранения и выдачи *данных,* представленных в *двоичном машинном коде* (0 и 1).

Память оперативная — см. *Оперативное запоминающее устройство (ОЗУ).*

Память постоянная — см. *Постоянное запоминающее устройство (ПЗУ).*

Паскаль — *язык программирования высокого уровня* и универсального назначения, пригодный для *программирования* большого класса задач — инженерных, экономических, научных, редакционных и др. Применяется при работе как с большими, так и с малыми ЭВМ.

Периферийное оборудование — комплект технических устройств, расположенных вне *микроЭВМ* и служащих для подготовки, ввода, вывода, хранения, отображения и печати *данных*. К периферийному оборудованию относятся: *дисплей, АЦПУ,* перфоратор, перфосчитыватель, *НМЛ, НМД, видеотерминал, телетайп, экранный пульт,* графопостроитель и др.

Персональная машина — **см.** *Персональный компьютер.*

Персональный компьютер — индивидуальная *микроЭВМ* для личного пользования. Различают профессиональные и бытовые *персональные компьютеры.* Основное назначение *персонального компьютера* состоит в решении задач, усиливающих интеллектуальную (умственную) деятельность человека. *Персональный компьютер* называют сокращенно ПК или, согласно ГОСТ,— ПМ *(персональная машина).*

ПЛ/М — *язык программирования высокого уровня* и универсального назначения. Пригоден для программирования инженерных, экономических, научно-технических и других задач. Индекс М указывает на то, что язык был создан на базе языка ПЛ/1 специально для *программирования микроЭВМ.*

Подпрограмма — *программа,* являющаяся частью основной *программы* и допускающая обращение к ней на любом шаге выполнения основной *программы.* Обычно *подпрограммы* хранятся в специально выделенной области *памяти* и вызываются для решения частных задач путем включения *подпрограмм* в нужные места основной *программы.*

Пользователь — тот, кто пользуется *микроЭВМ.* Эксплуатация больших ЭВМ обеспечивается штатом работников, состоящим из оператора, программиста, инженера и др., а работа *микроЭВМ* обычно обеспечивается одним человеком — *пользователем,* который должен выполнять функции и оператора, и программиста, и инженера.

Порт — техническое приспособление, выполняемое обычно в виде *интегральной микросхемы,* с помощью которой осуществляется связь и согласование действий *микропроцессора* с *внешними устройствами ввода и вывода данных.*

Постоянное запоминающее устройство (ПЗУ) — техническое устройство, в которое можно предварительно записать *данные* и содержать неизменными во время работы микроЭВМ, то есть использовать только для считывания и хранения данных.

Постоянная память — то же, что и постоянное запоминающее устройство.

ППЗУ — см. *Программируемое постоянное запоминающее устройство.*

Прерывание — временное прекращение выполнения *микроЭВМ команд* одной *программы* с целью выполнения более важных и нужных *команд* другой *программы* по специальному сигналу запроса на *прерывание.*

Прикладная программа — *программа,* предназначенная для решения задачи или класса задач в определенной, прикладной области применение *микроЭВМ,* например, для экономических расчетов, бухгалтерских вычислений, управления технологическим процессом и др.

Принтер — см. *алфавитно-цифровое печатное устройство.*

Программа — упорядоченная последовательность *команд* или *операторов,* способная выполнить определенную задачу при помощи *микроЭВМ.* В более широком смысле термин *«программа»* определяется как описание алгоритма решения задачи, доступное для *микроЭВМ.* Это описание представляет собой подробную инструкцию, указывающую, в какой последовательности, над какими *данными* и какие *операции* должна выполнить *микроЭВМ,* а также в каком виде выдать результат. Программа пишется на особом искусственном *языке программирования.*

Программатор — техническое приспособление, служащее для *программирования постоянного запоминающего устройства* (ПЗУ) путем занесения в *ячейки памяти ПЗУ* нужной или стирания записанной там *программы.*

Программирование — теоретическая и практическая деятельность по обеспечению программного управления обработки *данных микроЭВМ,* включающая в себя: постановку задачи, разработку *алгоритма* решения задачи, составление *схемы алгоритма,* разработку организации *данных,* выбор *языка программирования,* написание *программы* на выбранном языке, перевод *(трансляция)* написанной *программы* на *машинный язык двоичного кода, отладка программы* и др.

Программирование — очень сложный и трудоемкий процесс особенно при составлении *программ* для больших ЭВМ. Принимаются меры для *автоматизации программирования.*

Программируемая программа — см. *Транслятор.*

Программируемое постоянное запоминающее устройство *(ППЗУ)* — отличается от обычного *постоянного запоминающего устройства (ПЗУ)* тем, что *пользователь* может самостоятельно запрограммировать *ПЗУ* (ввести в него *программу*) с помощью *программатора.* Различают однократно *программируемое ПЗУ,* содержание *ячеек памяти* которого после записи изменить нельзя, и многократно *программируемое ПЗУ* или *репрограммируемое ПЗУ* (РППЗУ). Эти устройства допускают стирание и записывание *данных.* Например, с помощью ультрафиолетового облучения можно стереть прежние данные, и, подавая электрические импульсы, занести новые.

Программируемый интерфейс — отличается от обычного интерфейса тем, что схема и порядок работы определяется двоичным кодом, подаваемым на специальные (программные) выводы этого устройства.

Программное обеспечение — совокупность взаимосвязанных *программ,* описаний, инструкций и *данных,* позволяющих автоматизировать *отладку программ* и выполнить решение задач на *микроЭВМ.* Различают *общее* (системное) и *специальное программное обеспечение,* а также *кросс-* и *резидентное программное обеспечение.*

Прямой доступ к памяти — один из режимов работы *микроЭВМ,* при котором осуществляется прямой (непосредственный) доступ к *памяти* для взаимного обмена *данными* между *внешними устройствами* и *памятью.* На время этого обмена *микропроцессор* отключает свои адресные шины и шины данных от общей шины *микроЭВМ,* осуществляет обмен данными напрямую, минуя *микропроцессор.*

Регенерация данных — режим работы *запоминающего устройства,* в процессе которого осуществляется многократная перезапись находящихся *данных* с целью их сохранения.

Регистр — электронное устройство, служащее для временного хранения *данных,* представленных в *двоичном коде.* Обычно *регистр* выполняется на *триггерах.* Количество *триггеров* определяет длину (разрядность) *регистра.* Чаще всего длина *регистра* соответствует длине *машинного слова* данного *микропроцессора.*

Регистр признаков — набор триггеров, устанавливаемых в состоянии 0 или 1 в зависимости от результата операции, выполняемой в АЛУ. Условно обозначается буквой F.

Регистр сдвигающий — *регистр,* который, помимо выполнения основной задачи — хранения *данных,* позволяет перемещать (сдвигать) двоичные *данные* влево или вправо на требуемое число разрядов. Таким образом, единица, хранимая, например, в третьем разряде *регистра,* при сдвиге влево перемещается в четвертый разряд, а на ее место перемещается то, что содержалось во втором разряде.

Регистры общего назначения (РОН) — группа *регистров,* размещенных внутри *микропроцессора* и служащие для сверхоперативного, временного хранения *данных,* подлежащих обработке, а также для хранения указателей *адресов ячеек памяти,* двоичных чисел, промежуточных вычислений, счета циклов работы и др. Разные типы *микропроцессоров* содержат различное количество *регистров* в группе *РОН* (от 4 до 32). Отличительной особенностью *РОН* от других *регистров,* находящихся в *микропроцессоре,* являются их доступность программисту, то есть при написании *программ* можно управлять содержимым каждого *регистра РОН,* засылая нужные *данные* и извлекая их, пересылая из *регистра* в *регистр* и др.

Редактор — *программа* и технические средства, позволяющие редактировать текст программы, выводя ее на экран в виде определенных символов или строк и делая необходимые исправления.

Резидентное программное обеспечение — совокупность взаимно связанных служебных *программ (транслятор, редактор, монитор, загрузчик* и др.), создающих рабочую программу непосредственно на собственной *микроЭВМ* — на той *микроЭВМ,* где будет выполняться эта *программа.*

РОН — см. *Регистры общего назначения.*

РПЗУ — сокращенное название *репрограммируемого постоянного запоминающего устройства* (см. Программируемое постоянное запоминающее устройство).

Световое перо — техническое приспособление, используемое в системе отображения *данных микроЭВМ* для редактирования текста и вывода графической *информации* (графиков, чертежей) на экран *дисплея* или *экранный пульт* путем касания кончика *светового пера* электронного экрана. При этом фотодатчик, встроенный в световое перо, взаимодействует с электронным лучом экрана и образует управляющий сигнал, высвечивающий дополнительное изображение или стирающий прежнее, расположенное в точке встречи *светового пера* и электронного луча.

Светодиод — электронный полупроводниковый элемент, излучающий свет при прохождении электрического тока. Используется в качестве индикатора для отображения двоичных чисел, находящихся на шинах *микроЭВМ,* или для указания состояния *триггеров* и *логических элементов* схемы *микроЭВМ*. Обычно свечение обозначает единицу, а отсутствие света — нуль.

Сегмент — 1) Хранимая, обрабатываемая или передаваемая (для удобства работы) часть *данных* или *программы* отдельно, по частям (по сегментам). 2) Элемент цифрового сегментного индикатора.

Сигналы состояния — электрические сигналы в виде 0 или 1, кратковременно выдаваемые *микропроцессором* на шину данных в начале каждого машинного цикла и сообщающие о его состоянии (работа, ожидание, останов) или о тех действиях, которые предполагает совершать *микропроцессор* в данном машинном цикле.

Система команд — перечень (список, набор) *команд,* которые может выполнить данный *микропроцессор.*

Системное программное обеспечение — см. Общее программное обеспечение.

Слово состояния — восемь *сигналов состояния,* кратковременно появляющихся на восьми проводах шины данных в начале каждого машинного цикла и сообщающих о состоянии *микропроцессора* или о его действиях, совершаемых в машинном цикле.

Содержимое — *данные,* которые находятся (содержатся) в *ячейках* памяти, в *регистрах,* на *шине* или в *устройстве ввода—вывода* в виде *двоичного машинного кода* (сочетания нулей и единиц).

Специальное программное обеспечение — совокупность *программ* и *данных,* обеспечивающих решение конкретных задач на *микроЭВМ* (научные и инженерные расчеты, программа для автоматизации управления, бухгалтерские операции и др.).

Стек — специально выделенная область *памяти* для записи *содержимого* внутренних *регистров микропроцессора* и *адреса* последней выполняемой *команды* при необходимости прервать выполнение основной *программы* и перейти на выполнение другой *программы.* Такая *память* необходима, чтобы, вернувшись к выполнению основной программы, можно было восстановить прежнее положение, то есть выбрать из *стековой памяти* и переслать в *регистры* и в *счетчик команд* прежние *данные* и *адрес,* а затем продолжить выполнение основной прерванной *программы.*

Супервизор — часть *управляющей программы,* реализующая ввод и вывод *данных,* обмен с внешними накопителями, распределение *памяти* и управления внешним оборудованием *микроЭВМ (память* на магнитных дисках, *АЦПУ, дисплей* и др.).

Схема алгоритма — см. *Блок-схема алгоритма.*

Счетчик команд — *регистр,* содержимое которого соответствует *адресу* обращения к *памяти* в следующей *команде,* подлежащей исполнению. Содержимое *счетчика команд* после завершения текущей *команды,* как правило, увеличивается на единицу, поскольку последовательные *команды* выполняемой *программы* хранятся в соседних *ячейках памяти.*

Считывание данных — процесс работы *запоминающего устройства,* в результате которого производится считывание (выборка) *данных* из *запоминающего устройства.*

Таймер — 1) Техническое устройство, выдающее сигналы через заданные интервалы времени. 2) *Программа,* выполняющая действия, аналогичные техническому устройству, называемому *таймер.*

Телетайп — разновидность телеграфного аппарата с *клавиатурой* типа пишущей машинки. Применяется в качестве устройства ввода данных в ЭВМ путем предварительного преобразования телеграфного *кода* в нужный *код* для работы с данной ЭВМ.

Терминал — оконечное *устройство ввода—вывода данных.* Основное назначение *терминала* — ввода *программ* и *исходных данных,* вывод результатов и преобразование полученных *данных* из одной формы в другую. *Терминалы* наиболее распространены в микропроцессорных системах коллективного пользования, где абонент *(пользователь)* удален от системы на значительное расстояние. В качестве *терминала* используют *телетайп, АЦПУ, дисплей* и др.

Тест-программа — *программа для проведения* испытаний *микроЭВМ.* Некоторые *тест-программы* являются одновременно и диагностическими (см. *Диагностическая программа).*

Тетрада — четыре *бита* двоичного числа или четырехзначное двоичное число. Иногда *тетраду* называют *нибла.*

Транслятор — *программа* или техническое средство, выполняющее перевод *(трансляцию) программы,* написанной на одном из *языков программирования, в программу* на другом языке, в определенном смысле равносильную первой. Различают *трансляторы* компилирующего типа, которые выполняют только перевод программы, и интерпретирующего типа, которые осуществляют не только перевод, а и немедленное выполнение переведенной программы. Часто *транслятор* выполняет и дополнительные функции, например, диагностирует ошибки, формирует названия сегментов *данных,* выдает текст *программы* на печать и др.

Триггер — электронный элемент, который может находиться в одном из двух устойчивых состояний. В первом случае *триггер* на выходе выдает единицу (высокий логический уровень), а во втором — нуль (низкий логический уровень), что позволяет использовать *триггер* в качестве *запоминающего элемента* для одного разряда *(бита)* двоичного числа. Перевод *триггера* из одного состояния в другое производится путем подачи на его вход электрического сигнала.

Тристабильный выход — выход электронного элемента с тремя состояниями: нуль, единица или *высокоимпедансное состояние,* при котором элемент отключается от нагрузки.

Узел — набор элементов, соединенных между собой по нужной схеме и выполняющий определенную функцию (усилителя, счетчика, генератора и др.).

Указатель стека — *регистр,* который при осуществлении стековой операции (см. *Стек*) служит для хранения *адреса* последней занятой ячейки *стека,* что облегчает условие возврата к основной *программе* при ее дальнейшем выполнении.

Управляющая программа — общий термин, который используется в *операционной системе* для обозначения всех *программ,* предназначенных для управления вычислительным процессом, режимом работы *микроЭВМ,* организацией *данных,* условием обмена с *внешними устройствами,* распределением *памяти* и др.

Устройство — часть машины, изделия, прибора, имеющая определенное функциональное назначение, например, *арифметическо-логическое устройство, устройство управления, устройство ввода—вывода данных* и др.

Устройство ввода данных — служит для ввода *программ* и исходных *данных* в *микроЭВМ.* Существует большое количество типов таких устройств. При работе с *микроЭВМ* чаще всего пользуются: простейшими электрическими переключателями, позволяющими вводить 0 или 1, *клавишной панелью* с шестнадцатью клавишами для ввода *данных* в шестнадцатеричном виде, *клавиатурой* типа пишущей машинки, перфосчитывателями с перфолент, магнитосчитывателями с магнитных лент и дисков. В наиболее совершенных типах *микроЭВМ* применяются речевой ввод данных, читающие автоматы, распознающие текст, рисунки, графики.

Устройство ввода—вывода данных — устройство, служащее для двухстороннего обмена (ввода и вывода), *данными* между *пользователем* и *микроЭВМ.* Например, *телетайп, дисплей, экранный пульт.* В современных *микроЭВМ* используются сложные и пока еще не нашедшие широкого применения устройства речевого обмена *данными.*

Устройство вывода данных — устройство, служащее для вывода *данных* и представления их в форме, удобной для *пользователя.* К ним относятся: свето-

диоды, буквенно-цифровые и цифровые индикаторы, *алфавитно-цифровые печатающие* устройства, телевизионные и матричные, плоские экраны (дисплеи).

Помимо визуальных устройств вывода, применяются также речевые устройства в виде синтезаторов речи.

Устройство отображения информации — часть *устройств вывода данных,* служащая для наиболее удобного визуального восприятия *информации,* выведенной из *микроЭВМ,* на телевизионный экран, световое табло, панель, электромеханические и электромагнитные индикаторы лепесткового типа, графопостроители, большие экраны коллективного пользователя и ряд других устройств, которые чаще всего используются при работе с большими ЭВМ.

Устройство управления — часть *микропроцессора,* вырабатывающая из потока поступающих *данных* и *команд* последовательность управляющих сигналов, координирующих совместную работу всех узлов *микропроцессора* и согласовывающих его действия с *внешними устройствами.*

Файл — совокупность *данных,* рассматриваемая как единое целое, относящееся к одной категории, теме. Примером *файла* может быть совокупность анкетных данных кадровых работников предприятия. Как правило, *файл* содержит большие объемы *данных* и размещается на внешних носителях *памяти* (магнитных дисках, лентах). Кроме записи *данных, файл* снабжается служебными метками, указывающими номер *файла,* дату записи, условное название *содержимого* и др.

Формирователь шинный — электронный элемент, выполненный в виде *интегральной микросхемы* и включаемый в линию шин с целью изменения направления пересылки *данных.* Если на управляющий вывод *шинного формирователя* подать напряжение высокого логического уровня (1), то реализуется режим чтения *данных,* при котором данные передаются из *памяти* в *микропроцессор,* а если подать напряжение низкого логического уровня (0), то осуществляется режим записи *данных,* при котором *данные* пересылаются из *микропроцессора* в *память.* Помимо этого, *шинный формирователь* имеет еще один управляющий вывод. При подаче на этот вывод высокого логического уровня (1) выход *шинного формирователя* переходит в *высокоимпеданское состояние* и отключается от шины. *Шинный формирователь* часто служит также усилителем мощности проходящих через него сигналов.

Фортран — *язык программирования высокого уровня,* который применяется для написания *программ,* требующих выполнения большого количества вычислений над сравнительно небольшим количеством данных. *Фортран* чаще всего используется при работе с большими ЭВМ.

Флаг — специально выделенный **триггер,** сигнализирующий своим состоянием (выходным логическим уровнем) о происшедшем событии или определенном условии: например, переполнение *регистра,* возникшей ошибке, получения результата вычислений больше или меньше нуля и др. Если на выходе *триггера* возникает единица (условный сигнал о случившемся), то образно говорят, что «флаг поднят», а если нуль — «флаг опущен». Отсюда и произошло название термина.

ЦАП — сокращенное название *цифро-аналогового преобразователя,* который преобразовывает дискретный цифровой *код* в соответствующее ему аналоговое, непрерывное напряжение.

ЧТВВ — сокращенное название одного из управляющих сигналов схемы *микроЭВМ,* означающего режим чтения данных из внешнего устройства.

ЧТЗУ — сокращенное название одного из управляющих сигналов схемы *микроЭВМ,* означающего режим чтения данных из памяти (запоминающего устройства).

Шаговый режим работы — режим работы *микропроцессора,* позволяющий выполнять *команды программы* в желаемом темпе, что бывает очень полезным при *отладке программ* и контроле правильности работы *микроЭВМ.* Реализуется этот режим путем нажатия специальной управляющей кнопки «шаг», в результате чего *микропроцессор* выполняет один машинный цикл команды, а затем останавливается и ждет следующего нажатия кнопки.

Шина — совокупность электрических проводов (линия связи), по которым передаются *данные, адрес* или *управляющие сигналы* между отдельными устройствами *микроЭВМ.* Различают шину данных (ШД), адресную шину (ША) и шину управления (ШУ). Иногда их объединяют в одну общую шину, по которой в

течение некоторого времени передается адрес, затем в следующий промежуток времени — данные, а потом управляющие сигналы. Такое многократное использование линий приводит к уменьшению общего количества проводников в шине.

Шинный формирователь — см. *Формирователь шинный.*

Шифратор — техническое устройство для *кодирования* сигналов, позволяющее преобразовать входной сигнал, поступающий на один из его входов, в серию *кодовых двоичных чисел,* возникающий на его выходах и соответствующих входному сигналу. Например, *шифратор,* преобразующий нажатие кнопки, подключенной к одному из входов, в двоичный код, соответствующий значению этой кнопки.

Экранный пульт — *устройство ввода—вывода данных,* состоящее из телевизионного экрана, *светового пера* и *электрифицированной пишущей машинки,* и позволяющее визуально контролировать вводимые с *клавиатуры данные,* корректировать их и выводить на печать. Отличается от *видеотерминала* наличием устройства для печати и возможностью выполнения некоторых видов самостоятельной работы, независящей от *микроЭВМ.*

Элемент — простейшая электронная схема, выполняющая элементарную функцию (логический элемент, *триггер,* элемент временной задержки и др.).

Элемент памяти — см. *Запоминающий элемент.*

Эмулятор — *программа* и технические средства, которые заменяют *программу интерпретатора в случае ее исполнения на микроЭВМ,* имеющую другую *систему команд.*

Энергонезависимая память — *память,* содержимое которой сохраняется при отключенном электропитании.

Язык машинный — *язык программирования,* который непосредственно воспринимает *микропроцессор.* Этот язык представляет собой *систему команд* и *данных, выраженных в двоичном коде.* Каждый тип *микропроцессора* имеет свой *машинный язык,* иногда очень близкий к языку другого типа *микропроцессора.*

Язык программирования — язык, предназначенный для представления *программ* и состоящий из определенного набора символов и правил, управляющих способом и последовательностью соединения символов в осмысленные сообщения. Существует большое количество типов языков программирования. Их условно можно разделить на две группы: машиннозависимые (низкого уровня) и машиннонезависимые (высокого уровня). Каждый из языков имеет свои достоинства и недостатки и применяется в зависимости от типа *микроЭВМ* и задач, решаемых на ней.

Язык программирования высокого уровня — *язык программирования,* который существенно облегчает написание *программы,* так как позволяет составлять их в лаконичной и сжатой форме путем замены последовательности *команд* последовательностью *операторов,* представляющих собой комбинацию нескольких *команд.* Кроме того, используя *язык высокого уровня,* программист может не знать *системы команд* и устройства *микропроцессора,* для которого пишется *программа.* Перевод *программы* с *языка высокого уровня* на *машинный язык* осуществляется при помощи *программы,* называемой *транслятором.*

Язык программирования низкого уровня — *язык программирования,* близкий к *машинному языку.* Для написания *программы* на *языке низкого уровня* нужно знать *систему команд* и устройство *микропроцессора.* Одним из характерных *языков низкого уровня* является язык *ассемблера.* При переводе *программы* с *языка низкого уровня* на *машинный язык* существенно упрощается устройство перевода — *транслятор.*

Ячейка памяти — совокупность *запоминающих элементов,* реализующих *ячейку памяти,* в которой может разместиться *машинное слово.* Обычно количество *элементов* в *ячейке памяти* кратно *байту* и состоит из 8 или 16 элементов для восьми- или шестнадцатиразрядных типов *микропроцессора.*

Таблица 7

Наименование и назначение выводов микропроцессора К580ИК80 имеющего 40 или 48 выводов

Наименование выводов	Назначение выводов	Номера выводов МП—К580, имеющего	
		40 выводов	48 выводов
1	2	3	4
A0	Адресные выводы	25	7
A1		26	8
A2		27	9
A3		29	11
A4		30	12
A5		31	13
A6		32	14
A7		33	15
A8		34	16
A9		35	19
A10		1	17
A11		40	20
A12		37	18
A13		38	23
A14		39	22
A15		36	21
D0	Выводы данных	10	35
D1		9	34
D2		8	33
D3		7	32
D4		3	26
D5		4	27
D6		5	28
D7		6	31
C	Сигнал синхронизации	19	45
ПМ	—«— приема данных	17	43
ВД	—«— выдачи данных	18	44
ГТ	—«— готовности внешних устройств	23	3
ОЖ	—«— подтверждения режима ожидания	24	4
ЗХ	—«— запроса захвата шин	13	38
ПЗХ	—«— разрешения захвата шин	21	1
ЗПР	—«— запроса прерывания	14	39
РПР	—«— разрешения прерывания	16	42
СБР	—«— сброса счетчика команд	12	37
Ф1	Тактовый сигнал	22	2
Ф2	—«—	15	40
+5 В	Вывод для подключения первого источника питания	20	48
+12 В	—«— второго —«—	28	6
—5 В	—«— третьего —«—	11	24
ОВ	Общий вывод для подключения всех трех источников питания	2	25

Продолжение таблицы 7

1	2	3	4
—	Незадействованные выводы	—	5
—		—	10
—		—	29
—		—	30
—		—	36
—		—	41
—		—	46
—		—	47

Таблица 8

Потребляемый ток и нумерация выводов питания микросхем, применяемых для сборки микроЭВМ

Тип микросхемы	Потребляемый ток в мА			Номера выводов питания				Примечание
	+5В	+12В	−5В	0В	+5В	+12В	−5В	
К155ЛА2	6	—	—	7	14	—	—	
К155ЛА3	22	—	—	7	14	—	—	
К155ЛА8	22	—	—	7	14	—	—	
К155ЛЛ1	38	—	—	7	14	—	—	
К155ЛЕ1	27	—	—	7	14	—	—	
К155ЛН1	33	—	—	7	14	—	—	
К155ТМ2	30	—	—	7	14	—	—	
К155ТМ7	53	—	—	12	5	—	—	
К589АП16	130	—	—	8	16	—	—	
К580ГФ24	115	12	—	8	16	9	—	
К580ИК80	70	50	1	25	48	6	24	48 выводов
КР580ИК80А	110	85	1	2	20	28	11	40 выводов

Таблица 9

Виды обозначений выводов микросхем

Обозначение выводов, принятых в		Наименование выводов
книге	других изданиях	
ПМ	DB	Прием
ВД	WR	Выдача
ГТ	RA, RDY	Готовность
ОЖ	WAIT	Ожидание
ЗХ	HOLD	Запрос захвата шин
ПЗХ	HLDA	Разрешение захвата шин
ЗПР	INT	Запрос прерывания
РПР	INTE	Разрешение прерывания
СБР	R, RES	Сброс
С, СИН	SYN	Синхронизация

Продолжение таблицы 9

Обозначение выводов, принятых в		Наименование выводов
книге	других изданиях	
ОВ	GND	Общий вид, масса, земля
Ф	F	Тактовый сигнал
ВМ, ВК	CE, CS	Выбор микросхемы (кристалла)
ВШ, ВР	MO, DIEN	Выбор режима
ЧТ	RD	Чтение
ЗП	WR	Запись
ШД	D	Шина данных
ША	A	—«— адреса
—	◇, EZ	Вывод с состоянием высокого импеданса
—	◇	—«— с открытым коллектором

Таблица 10

Условное обозначение микросхем на электрических принципиальных схемах

Обозначение	Наименование
F	Шинный формирователь
CPU	Микропроцессор
RAM	ОЗУ
ROM	ПЗУ
DC	Дешифратор
CD	Шифратор
CT	Счетчик
TT	Триггер
RG	Регистр
G	Генератор
G1, ⊓	Одновибратор
&	Логический элемент И
1	—«— ИЛИ

Таблица 11

Коды команд микропроцессора К580ИК80 в шестнадцатеричном виде

Коды команд пересылки данных

пересылка		пересылка		непосредств. загрузка	
MOV— A,A	7F	MOV— E,A	5F	MVI— A,D8*)	3E
A,B	78	E,B	58	B,D8	06
A,C	79	E,C	59	C D8	0E
A,D	7A	E,D	5A	D,D8	16
A,E	7B	E,E	5B	E,D8	1E
A,H	7C	E,H	5C	H,D8	26
A,L	7D	E,L	5D	L,D8	2E
A,M	7E	E,M	5E	M,D8	36

MOV—	B,A	47
	B,B	40
	B,C	41
	B,D	42
	B,E	43
	B,H	44
	B,L	45
	B,M	46
MOV—	C,A	4F
	C,B	48
	C,C	49
	C,D	4A
	C,E	4B
	C,H	4C
	C,L	4D
	C,M	4E
MOV—	D,A	57
	D,B	50
	D,C	51
	D,D	52
	D,E	53
	D,H	54
	D,L	55
	D,M	56
MOV—	H,A	67
	H,B	60
	H,C	61
	H,D	62
	H,E	63
	H,H	64
	H,L	65
	H,M	66
MOV—	L,A	6F
	L,B	68
	L,C	69
	L,D	6A
	L,E	6B
	L,H	6C
	L,L	6D
	L,M	6E
MOV—	M,A	77
	M,B	70
	M,C	71
	M,D	72
	M,E	73
	M,H	74
	M,L	75
	XCNG	EB

LXI —	B,D16*)	01
	D,D16	11
	H,D16	21
	SP,D16	31

загрузка-хранение

LDAX B	0A
LDAX D	1A
LHLD ADR	2A
STAX B	02
STAX D	12
SHLD ADR*	22
LDA ADR	3A
STA ADR	32

Коды арифметических и логических команд

сложение

ADD—	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC—	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

вычитание

SUB—	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB —	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

увеличение

INR—	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX —	B	03
	D	13
	H	23
	SP	33

уменьшение

DCR—	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX—	B	08
	D	18
	H	28
	SP	38

специальные

DAA	27
CMA	2F
STC	37
CMC	3F

сдвиг

RLC	07
RRC	0F

ANA—	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA—	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA—	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP—	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	DE

двойное сложение

DAD—	B	09
	D	19
	H	29
	SP	39

RAL	17
RAR	1F

непосредств. операции

ACID8	CE
ADID8	C6
SUI	D6
SBI	DE
ANI	E6
XRI	EE
ORI	F6
CPI	FE

Коды команд перехода, ввода, вывода и управления

Переходы

JMP ADR	C3
JNZ ADR	C2
JZ ADR	CA
JNC ADR	D2
JC ADR	DA
JPO ADR	E2
JPE ADR	EA
JP ADR	F2
JM ADR	FA
PCHL	E9

Вызовы

CALL ADR	CD
CNZ ADR	C4
CZ ADR	CC
CNC ADR	D4
CC ADR	DC
CPO ADR	E4
CPE ADR	EC
CP ADR	F4
CM ADR	FC

Возвраты

RET	C9
RNZ	C0
RZI	C8
RNC	D0
JRC	D8
RPO	E0
RPE	E8
RP	F0
RM	F8

Прерывания

RST—	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	E7
	7	FF

Ввод—вывод

OUT N	D3
IN N	D8

Операции со стеком

PUSH—	B	C5
	D	D5
	H	E5
	PSW	F5
POP—	B	C1
	D	D1
	H	E1
	PSW	F1

Управления

D1	F3
EI	FB
NOP	00
HLT	76

*) Условные обозначения в таблице 11

D8 — однобайтный операнд
D16 — двухбайтный операнд
ADR — двухбайтный адрес
N — номер порта

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Абрамов С. А. Элементы программирования.— М.: Наука, 1982.— 162 с.
2. Агаханян Т. М. Интегральные микросхемы.— М.: Энергоатомиздат, 1983.— 133 с.
3. Алексеев С. Применение микросхем серии К155//Радио.— 1977.— № 10.— С. 39—41; 1978.— № 5.— С. 37—38; 1982.— № 2.— С. 30—32; 1986.— № 5.— С. 28—31; № 6.— С. 44—45; № 7.— С. 32—34.
4. Бирюков С. А. Радиолюбительские цифровые устройства.— М.: Радио и связь, 1982.— 72 с.
5. Борисов В., Партин А. Основы цифровой техники//Радио.— 1985.— № 1.— С. 50—52; № 2.— С. 51—53; № 3.— С. 50—52; № 4.— С. 52—53; № 5.— С. 51—53; № 7.— С. 50—51; № 8.— С. 51—52; № 9.— С. 54—55; № 10.— С. 51—54; № 11.— С. 49—50; № 12.— С. 49—50.
6. Вдовикин А. И. Занимательные электронные устройства.— М.; Радио и связь, 1981.— 80 с.
7. Верлань А. Ф., Касаткин В. Н. Основы информатики и вычислительной техники.— К.: Рад. шк., 1985.— 144 с.
8. Волков А. Ваш помощник — компьютер//Моделист-конструктор.— 1987.— № 2.— С. 19—22; № 4.— С. 24—27.
9. Вуд А. Микропроцессоры в вопросах и ответах.— М.: Энергоатомиздат, 1985.— 182 с.
10. Галатузова М. А., Комский Д. М. Первые шаги в электротехнику.— М.: Просвещение, 1979.— 106 с.
11. Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры.— М.: Мир,— 1983.— 464 с.
12. Гилмор Ч. Введение в микропроцессорную технику.— М.: Мир, 1984.— 463 с.
13. Глушков В. М. Что такое кибернетика? — М.: Просвещение, 1975.— 185 с.
14. Григорьев В. Л. Программное обеспечение микропроцессорных систем.— М.: Энергоатомиздат, 1983.— 136 с.
15. Громов Г. Персональные вычисления — новый этап информационной технологии//Микропроцессорные средства и системы.— 1984.— № 1.— С. 37—50.
16. Гутер Р. С., Полунов Ю. Л. От абака до компьютера.— М.: Знание, 1981.— 191 с.
17. Деге В. ЭВМ думает, считает, управляет.— М.: Мир, 1974.— 227 с.
18. Депман И. Мир чисел.— Л.: Дет. лит., 1975.— 128 с.
19. Дмитриенко А. Н. Электронная автоматика.— М.: Изд-во ДОСААФ, 1973.— 76 с.
20. Долгий А. Если нет КР580ВГ75...//Радио.— 1987.— № 5.— С. 22—24; № 6.— С. 33—34.
21. Дробница Н. А. Электронные устройства для радиолюбителей.— М.: Радио и связь, 1983.— 47 с.
22. Ерошов А. Монтаж микросхем при макетировании//Радио.— 1984.— № 11.— С. 44.
23. Житомирский В. Г., Шеврин Л. Н. Математическая азбука.— М.: Педагогика, 1984.— 46 с.
24. Зеленко Г., Попов С., Панов В. Радиолюбителю о микропроцессоре и микроЭВМ//Радио.— 1982.— № 9.— С. 32—36; № 10.— С. 24—28; № 11.— С. 38—40; № 12.— С. 31—34; 1983.— № 2.— С. 40—43; № 3.— С. 31—35;

№ 4.— С. 27—31; № 6.— С. 42—46; № 7.— С. 23—27; № 8.— С. 23—27; № 9. С. 32—35; № 10.— С. 28—31; № 11.— С. 31—34; № 12.— 24—28.

25. Зеленко Г., Панов В., Попов С. Бейсик для «Микро-80»/Радио.— 1985.— № 1.— С. 31—36; № 2.— С. 39—42; № 3.— С. 42—45.

26. Иванов Б. С. Электроника в самоделках.— М.: Изд-во ДОСААФ, 1981.— 191 с.

27. Кальнин Б. Основы вычислительной техники//Радио.— 1979.— № 5.— С. 29—32; № 6.— С. 26—28; № 7.— С. 24—26; № 8.— С. 26—28; № 9.— С. 27—31; № 10.— С. 40—42; № 11.— 23—27; № 12.— С. 21—24.

28. Касаткин В. Н. Введение в кибернетику.— К.: Рад. шк., 1986.— 176 с.

29. Кетков Ю. Л. Программирование на Бейсике.— М.: Статистика, 1976.— 158 с.

30. Корнейчук В. И., Тарасенко В. П., Мишинский Ю. Н. Вычислительное устройство на микросхемах.— К.: Техніка, 1986.— 262 с.

31. Коффрон Дж. Технические средства микропроцессорных систем.— М.: Мир, 1983.— 344 с.

32. Лавров С. Кому и для чего нужна персональная вычислительная машина?//Микропроцессорные средства и системы.— 1984.— № 1.— С. 34—36.

33. Логические пробники//Радио.— 1980.— № 3.— С. 30—36.

34. Любимов Г. В., Новиков С. М. Знакомимся с электрическими цепями.— М.: Наука, 1981.— 126 с.

35. Макетные платы ПР//Радио.— 1986.— № 9.— С. 58.

36. Мальцева Л. А., Фромберг Э. М., Ямпольский В. С. Основы цифровой техники.— М.: Радио и связь, 1986.— 126 с.

37. Мерфи Дж. Как устроены и работают электронные цифровые машины.— Мир, 1965.— 389 с.

38. МикроЭВМ/Под ред. А. Дирксена.— М.: Энергоиздат, 1982.— 328 с.

39. Морисуэ М., Есикава Т. МикроЭВМ за три дня.— М.: Мир, 1981.— 183 с.

40. Мясников В. А., Майоров С. А., Новиков Г П. ЭВМ для всех.— М.: Знание, 1984.— 244 с.

41. Основы информатики и вычислительной техники/Под ред. А. П. Ершова и В. М. Монахова: В 2 ч.— К.: Рад. шк., 1985—1986.— Ч. 1.— 96 с.; Ч. 2.— 144 с.

42. Отряшенков Ю. М. Юный кибернетик.— М.: Дет. лит., 1978.— 576 с.

43. Пекелис В. Д. Маленькая энциклопедия о большой кибернетике.— М.: Дет. лит., 1973.— 336 с.

44. Персональная микроЭВМ «ИРИША»/ В. Барышников, В. Быстров, М. Воронов и др.//Микропроцессорные средства и системы.— 1985.— № 3.— С. 5—8; 1986.— № 1.— С. 61—72; № 2.— С. 52—62; № 3.— С. 59—62.

45. Персональный радиолюбительский компьютер «Радио-86 РК»/Д. Горшков, Г. Зеленко, Ю. Озеров, С. Попов//Радио.— 1986.— № 4.— С. 24—26; № 5.— С. 31—34; № 6.— С. 26—28; № 7.— С. 26—28; № 8.— С. 23—26; № 9.— С. 27—29.

46. Растригин Л. ЭВМ — системы—сети//Радио.— 1986.— № 6.— С. 22—24; № 8.— С. 20—22; № 9.— С. 24—26; № 10.— С. 32—34; № 11.— С 23—25; № 12.— С. 12—14.

47. Салтовский А. Н., Первин Ю. А. Как работает ЭВМ.— М.: Просвещение, 1986.— 161 с.

48. Салтыков А. И., Семашко Г. А. Программирование для всех.— М.: Наука, 1980.— 159 с.

49. Сворень Р. Л. ЭВМ: приглашение к знакомству//Радио.— 1978.— № 3.— С. 54—57; № 4.— С. 51—54; № 5.— С. 50—52; № 6.— С. 51—53.

50. Сворень Р. Л. Электроника шаг за шагом.— М.: Дет. лит., 1979.— 430 с.

51. Седов Е. А. Занимательно об электронике.— М.: Мол. гвардия, 1979.— 352 с.

52. Сетевые блоки питания//Радио.— 1984.— № 10.— С. 63.

53. Сикорук Я. Физика для малышей.— М.: Педагогика, 1983.— 186 с.

54. Стейнберг У., Форд У. Электро- и радиотехника для всех.— М.. Сов. радио, 1972.— 242 с.

55. Стрыгин В. В. Основы автоматики и вычислительной техники.— М.: Энергоиздат, 1981.— 134 с.

56. Уорт Т. Программирование на языке Бейсик.— М.: Машиностроение, 1981.— 255 с.

57. Условные графические обозначения: Элементы цифровой техники// Радио.— 1986.— № 10.— С. 54—56.

58. Уэзерелл Ч. Этюды для программирования.— М.: Мир, 1982.— 267 с.

59. Федорова С. Простые устройства на логических элементах. В помощь радиолюбителю.— М.: Изд-во ДОСААФ, 1981.— Вып. 74.— С. 51—66.

60. Фомин С. В. Системы счисления.— М.: Наука, 1980.— 41 с.

61. Хейзерман Д. Применение интегральных схем.— М.: Мир, 1984.— 208 с.

62. Хелмс Г. Языки программирования: Краткое руководство.— М.: Радио и связь, 1985.— 174 с.

63. Хокингс Г. Цифровая электроника для начинающих.— М.: Мир, 1986.— 232 с.

64. Эймишен Ж. Электроника? Нет ничего проще! — М.: Энергия, 1975.— 247 с.

65. Энциклопедический словарь юного техника.— М.: Педагогика, 1980.— 512 с.

66. Юшин А. Микропроцессорные БИС К580, КР580//Радио.— 1984.— № 9.— С. 59—61; № 10.— С. 59—61; № 11.— С. 59—61; № 12.— С. 55—56; 1985.— № 4.— С. 59—61.

ОГЛАВЛЕНИЕ

Учебное издание
Серия «Когда сделаны уроки»

Тищенко Владимир Германович, Тищенко Георгий Владимирович
МИКРОЭВМ — СВОИМИ РУКАМИ
Для старшего школьного возраста
Киев, «Радянська школа»
Заведующий редакцией математики *Н. Е. Зубченко.* Художественный редактор *В. А. Пузанкевич.*
Технический редактор *Л. Б. Ланцман.* Корректоры *И. Н. Ситниченко, Л. С. Командир.*

ИБ № 6751

Сдано в фотонабор 28.07.88. Подписано в печать 07.04.89. БФ 05533. Формат 60 × 90/16. Бумага тип. № 2. Гарнитура литературная. Печать высокая с ФПФ. Усл. печ. л. 13 + 2 вкл. Усл. кр.-отт. 25.50. Уч.-изд. л. 15,34 + 2,11 вкл. Тираж 100 000 экз. Изд. № 32862. Заказ № 8-835. Цена 80 к.

Издательство «Радянська школа» 252053, Киев, Ю. Коцюбинского, 5.

Харьковская книжная фабрика им. М. В. Фрунзе,
310057, Харьков, Донец-Захаржевского, 6/8.