БУРЕЕВ ЛЕВ НИКОЛАЕВИЧ
ДУДКО АЛЕКСЕЙ ЛЬВОВИЧ
ЗАХАРОВ ВАЛЕРИЙ НИКОЛАЕВИЧ

Простейшая микро-ЭВМ. Проектирование. Наладка. Использование

ПРЕДИСЛОВИЕ

Появление микропроцессоров сыграло важную роль в развитии вычислительной техники, средств обработки информации и управляющих устройств, являющихся основой автоматизации в различных сферах человеческой деятельности. Неослабевающий интерес к микропроцессорам объясняется такими их особенностями, как низкая стоимость, высокая надежность, компактность и значительные функциональные и вычислительные возможности, позволяющие применять их даже там, где использование средств цифровой обработки информации ранее считалось нецелесообразным. В настоящее время как у нас в стране, так и за рубежом издается весьма обширная литература по микропроцессорной технике и возможностям ее применения. И все же книг с описанием реально построенных конструкций и устройств на базе микропроцессоров явно недостаточно для удовлетворения постоянно растущего спроса на такие публикации. Именно это побудило авторов взять на себя смелость написать нечто вроде руководства, ориентированного на читателя, пожелавшего ознакомиться с работой простейшей микро-ЭВМ или заняться ее изготовлением.

Конечно, эта книга — не инструкция по изготовлению микро-ЭВМ в полном смысле этого слова, хотя в ней довольно подробно описываются конструкция вычислительной машины и ее работа. Основная цель книги — помочь разобраться в том, что такое микропроцессор, как он работает, как необходимо его программировать и как на его основе можно создавать разнообразные устройства, применяющиеся в технике, в быту, в повседневной практической деятельности.

Современный микропроцессор - довольно сложное устройство, работу которого не удается описать в деталях вне связи с системой, в составе которой он функционирует (в отличие от других более простых электронных приборов, таких например, как электронная лампа). К сожалению, в специальной литературе микропроцессор в подавляющем большинстве случаев описывается автономно. Из такого описания не всегда понятны детали его работы и особенности применения. Предлагаемая читателям книга в некоторой степени восполняет этот пробел. Авторами описывается схема простейшей микро-ЭВМ как пример простейшей микропроцессорной системы. Будет или не будет читатель строить эту микро-ЭВМ — не так уж и важно. Важно, что на базе этой конкретной микропроцессорной системы он получит необходимые сведения о работе микропроцессора и сможет построить в дальнейшем аналогичные системы по своему выбору и вкусу.

Выступая в 1970 г. с докладом о перспективах развития и применения вычислительной техники на конференции в Московском физико-техническом институте, академик В. М. Глуш-ков говорил о том, что недалек тот день, когда вычислительная техника шагнет в повседневную жизнь и буквально каждая семья сможет получить доступ к вычислительным ресурсам. Во времена господства универсальных вычислительных машин-гигантов это высказывание представлялось по меньшей мере весьма смелым прогнозом. Но прошли годы и вот уже микроэлектроника стучится в двери наших квартир, появляется на рабочих местах в учреждениях, приближая тот день, когда вычислительные машины станут для нас столь же привычными в быту, как холодильники, стиральные машины и цветные телевизоры.

Отсюда ясно, как важна популяризация тех знаний, которые раньше были необходимы лишь специалистам. В особенности это касается знаний в следующих трех областях: математической логике, программировании и электронике. Но даже специалисты, равным образом ориентирующиеся в указанных трех областях, в настоящее время встречаются не так уж часто. Пользователи ЭВМ прошлых поколений практически никогда не сталкивались с аппаратной реализацией своих программ, а специалисты в области электроники, как правило, не занимались программированием. Поэтому широкая подготовка специалистов нового типа — насущная проблема сегодняшнего дня.

Данная книга вовсе не претендует на роль учебного пособия с изложением основ математической логики, электроники и программирования. Цель у книги другая — привлечь широкий круг читателей к относительно новому, увлекательному миру конструирования микроэлектронных устройств на базе микропроцессорной техники, сфера применения которых не ограничивается традиционными вычислительными задачами. Создание программируемых устройств с широкими функциональными возможностями - микроэлектронных помощников

(пусть на первое время совсем простых), повышающих эффективность интеллектуальной деятельности на производстве и дома, - вот, может быть, самая интересная и многообещающая область исследований в наш век всеобщей компьютеризации.

Необходимо заметить, что авторы вовсе не хотели бы склонить будущих конструкторов к попытке воспроизвести копию промышленной микро-ЭВМ. Любительским конструкциям трудно тягаться с изделием, выпускаемым промышленностью. Тем не менее широкое привлечение любителей к микроэлектронному конструированию позволит в ряде случаев найти те оригинальные технические решения, которые в дальнейшем могут быть использованы целиком в конструкциях соответствующих промышленных изделий или положены в их основу. Для чтения книги не требуется специальных знании в области микропроцессорной техники. Тем не менее предполагается, что читатель сможет, пользуясь приведенными в книге рекомендациями самостоятельно собрать простую ЭВМ из малодефицитных деталей, отладить ее, проделать на ней ряд упражнении по программированию решения различных задач, а также изучить способы подсоединения дополнительных внешних устройств, значительно расширяющих возможности построенной машины. Хотя описываемая микро-ЭВМ построена по универсальной схеме, допускающей наращивание аппаратуры до широких пределов (скажем, до масштабов персональной ЭВМ), основное назначение ее - учебное, т. е. позволяющее в максимально короткое время получить навыки основ программирования и проектирования микропроцессорных систем. С этой целью читателю дается весь необходимый материал, приводятся реальные схемы с реальными характеристиками. В отличие от большинства подобных изданий в книге описываются не только отдельные узлы машины, но и целиком вся ее схема. Выводы всех микросхем промаркированы, и каждая микросхема описана в деталях. Поэтому читателю нет необходимости обращаться к зачастую труднодоступным справочным источникам. Главы 1-4, а также приложение 1 (система команд микро-процессора КР580ВМ80А) написаны В.Н. Захаровым, гл. 5-7 написаны А. Л. Дудко, а гл. 8-10 - Л. Н. Буреевым. Принципиальная электрическая схема описываемой микро-ЭВМ (приложение 2) разработана А. Л. Дудко, а описываемые схемы сопряжения микро-ЭВМ с дополнительными внешними устройствами (в том числе с бытовыми телевизором и магнитофоном) , а также схема статического аппаратного эмулятора разработаны Л. Н- Буреевым. Предисловие к книге написано авторами совместно.

Авторы позволяют себе надеяться, что книга окажется полезной не только будущим конструкторам микро-ЭВМ, но и всем тем, кто стремится расширить свои знания в области применения микропроцессорной техники.

Авторы выражают признательность рецензенту канд. техн. наук В. Ф. Корнюшко и редактору проф. Д. А. Поспелову за доброжелательную критику и замечания, которые способствовали улучшению содержания и стиля книги.

Все замечания по содержанию книги, методике изложения, а также все предложения по усовершенствованию схемы и конструкции описываемой машины авторы примут с благодарностью. Пожелания и замечания просьба направлять по адресу: 113114, Москва, М-114, Шлюзовая наб., 10, Энергоатомиздат.

Авторы

1

ЧТО ТАКОЕ МИКРО-ЭВМ?

1.1. ТИПЫ МИКРО-ЭВМ И ОБЛАСТИ ИХ ПРИМЕНЕНИЯ

Понятие микро-ЭВМ отнюдь не означает, что пользователь имеет дело с упрощенным вариантом обычной ЭВМ, обладающим весьма ограниченными возможностями В истории создания вычислительных машин десятилетие с 70-х по 80-е годы сыграло важную роль. Благодаря успехам микроэлектронной технологии появилась возможность конструировать вычислительные машины небольших габаритов, с малым потреблением электроэнергии и в достаточной мере "производительные". Вычислительные возможности современных микро-ЭВМ не уступают возможностям средних ЭВМ начала 70-х годов. Если понятие ЭВМ неразрывно связано с понятием вычислительного центра, крупного предприятия или сложной технической системы, то микро-ЭВМ - это массовое изделие, доступное не только небольшим производственным коллективам но и отдельным лицам вследствие невысокой стоимости, малой материалоемкости, низкого энергопотребления, высокой надежности. Микро-ЭВМ может быть использована для управления производством, а также отдельной, в ряде случаев сложной технической системы, как элемент оборудования рабочего места конструктора-исследователя или научного работника, в быту и во многих других сферах.

Переворот в технике конструирования ЭВМ произошел вследствие перехода к изготовлению основных узлов вычислительной машины (и в частности, ее главного узла - центрального процессорного элемента) в габаритах одной микросхемы или нескольких микросхем с площадью размещения активных элементов в каждой микросхеме порядка 100 мм^2 и менее. При разработке такого процессора, получившего название "микропроцессор" (МП), было учтено требование максимального использования аппаратурных возможностей выполнения им вычислительных или логических функций.

В конструктивном отношении микропроцессоры могут быть однокристальными (выполненными в виде одной микросхемы), многокристальными (выполненными в виде нескольких разнотипных микросхем, каждая из которых представляет собой функционально законченную часть логической схемы процессора) и секционными многокристальными (выполненными в виде нескольких однотипных микросхем, представляющих собой отдельные секции, позволяющие построить процессор с числом разрядов, пропорциональным числу используемых секций).

Кроме МП, предназначенных для обработки дискретной информации, существуют аналоговые микропроцессоры (АМП), предназначенные для обработки аналоговой информации. В их структуру включены аналого-цифровые (аналого-дискретные) и цифро-аналоговые (дискретно-аналоговые) преобразователи, т. е. устройства, преобразующие аналоговый сигнал (например, непрерывно меняющееся входное напряжение) в цифровой (набор напряжений двух фиксированных уровней, представляющих двоичный код) и обратно. Обработка аналоговой информации, преобразованной в дискретную, производится в АМП, как и в обычном микропроцессоре. Кроме однокристальных микропроцессоров существуют однокристальные микро-ЭВМ (ОМ-ЭВМ), представляющие собой микросхему, объединяющую в своем составе все основные устройства, необходимые для ее функционирования.

При использовании ОМ-ЭВМ необходимо добавить источник питания, внешние устройства и в ряде случаев дополнительное внешнее запоминающее устройство. На базе МП или ОМ-ЭВМ может быть построена одноплатная микро-ЭВМ, представляющая собой законченный конструктивный элемент. Одноплатная микро-ЭВМ может входить в состав многоплатной микро-ЭВМ, включающей, кроме того, платы сопряжения с внешними устройствами, а также источник питания, пульт управления, аппаратуру индикации ("голая микро-ЭВМ"). Если "голую микро-ЭВМ" "одеть" внешними устройствами (алфавитно-цифровой клавиатурой, дисплеем, накопителем на гибком магнитном диске и т. п.), то получится вычислительный комплекс.

Одноплатная микро-ЭВМ, ОМ-ЭВМ и "голая микро-ЭВМ" могут быть использованы в составе управляющих систем или измерительных комплексов. Примерами ОМ-ЭВМ являются однокристальные восьмиразрядные микро-ЭВМ серии К 1816, однокристальные четырехразрядные микро-ЭВМ серий К 1820 и К 1814. К одноплатным машинам относится, например, микро-ЭВМ "Истра", а к многоплатным — микро-ЭВМ "Электроника 60", ЕС-1840, "Квант" и др.

По способу реализации системы команд микро-ЭВМ разделяются на два типа. В микро-ЭВМ первого типа система команд является постоянной (фиксированной), а в микро-ЭВМ второго типа - изменяемой (программируемой на уровне микрокоманд). Более простыми, дешевыми и распространенными являются машины первого типа.

К наиболее распространенным микро-ЭВМ, выпускаемым отечественной промышленностью, относятся персональные (ПЭВМ) и профессиональные персональные (ППЭВМ) машины семейства "Электроника", а также машины: ЕС 1840, "Искра 1030", "Нейрон И9.66", "Агат", "Корвет", СМ-1810 и др.

Семейство микро-ЭВМ "Электроника" - это ряд универсальных программно-совместимых машин различной производительности. Наиболее производительные машины этого ряда сравнимы по параметрам с развитыми мини-ЭВМ. Семейство микро-ЭВМ "Электроника" - это ряд машин, ориентированных на использование в управлении технологическими процессами, для сбора и обработки данных, для обработки сообщений и управляющей информации в системах связи и контрольно-измерительных системах. Отдельные модели ряда могут быть встроены в соответствующие подсистемы управления и контроля Модели "Электроника" - это микро-ЭВМ универсального применения, которые с успехом могут быть использованы в системах автоматизированного управления. Одним из важнейших достоинств этой серии является программная совместимость с отечественными мини-ЭВМ СМ-3, СМ-4, а также с зарубежными мини-машинами семейства PDP-11, что позволяет использовать разработанное ранее программное обеспечение. Машины "Электроника 85" и "Электроника БК-0010" относятся к классу персональных компьютеров.

Предназначенная в основном для тех же целей микро-ЭВМ СМ-1810 является машиной, программно-совместимой с микро-ЭВМ, построенными на базе микропроцессора 8080 фирмы Intel.

Кроме перечисленных микро-ЭВМ отечественной промышленностью выпускается большой ассортимент диалоговых вычислительных комплексов, например ДВК-1 - ДВК-3, с высокой производительностью, что позволяет использовать программное обеспечение этой машины, а также программное обеспечение мини-ЭВМ "Электроника 100/25". Операционная система вычислительного комплекса ДВК допускает использование языков БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, КОБОЛ, ПЛ/1, что предоставляет большие возможности для программирования.

Современные микро-ЭВМ обладают несравненно большими возможностями, чем многие вычислительные машины прошлых поколений. Дешевизна, надежность и доступность микро-ЭВМ позволяют использовать их для решения таких задач, для которых применение средств вычислительной техники ранее было неоправданным.

В сфере промышленного производства микро-ЭВМ могут использоваться в составе информационно-управляющих вычислительных систем (ИУВС), в системах технического управления объектами и технологическими процессами и в системах организационно-технического управления цехами, предприятиями, отраслями и т. п. В таких системах микро-ЭВМ используются для сбора и обработки данных, выполнения сложных экономических и технических расчетов, планирования, управления и контроля. В управлении сложными техническими системами микро-ЭВМ чаще всего используются в составе встроенных средств управления и контроля. Замена высокопроизводительной и дорогостоящей ЭВМ, используемой в качестве центрального управляющего органа, сетью микро-ЭВМ повышает надежность, эффективность и гибкость управления сложной технической системой, позволяет организовать управление в реальном времени и снижает стоимость общих затрат на управление.

Применение микро-ЭВМ в машиностроении позволяет перейти от существующих конструкций станков с числовым программным управлением к более совершенным высокопроизводительным робототехническим конвейерным системам и к организации на их основе гибких автоматизированных производств.

Расширению сферы использования ЭВМ (особенно в последние годы) способствовало появление нового класса микро-ЭВМ — персональных ЭВМ (ПЭВМ). Под ПЭВМ подразумевается микро-ЭВМ, предназначенная для индивидуального пользования (подобно пишущей машинке, телевизору, магнитофону), но со значительно более широкими функциональными возможностями, позволяющими использовать ее для решения самых разнообразных задач — от сложнейших профессиональных расчетов до самых мелких бытовых. Обычно ПЭВМ так и классифицируются: профессиональные и бытовые. Профессиональные ПЭВМ используются профессионалами-конструкторами, технологами, инженерами, научными работниками, журналистами, редакторами и т. п. Они оказываются полезными при индивидуальной обработке технической, экономической, медицинской и другой информации, в преподавательской деятельности; позволяют обеспечить оперативный доступ к отраслевым, региональным информационным источникам через локальные сети ЭВМ. Бытовые ПЭВМ могут быть использованы в качестве домашнего информационного центра. С их помощью можно проводить развлекательные и познавательные игры, организовывать учебные курсы (например, по изучению иностранных языков или курсов по школьной программе), обеспечивать доступ к справочной информации: адресам, телефонам, рецептам и т. п. Микро-ЭВМ, выпускаемые промышленностью, являются слишком сложными, чтобы брать их за образец при попытке самостоятельного построения. Возникает вопрос, можно ли вообще самому построить хоть какой-нибудь простейший вариант вычислительной машины?

1.2. МОЖНО ЛИ САМОМУ ПОСТРОИТЬ ЭВМ?

Еще 15 лет назад человека, задавшего такой вопрос, посчитали бы не совсем нормальным. Действительно, до появления микросхем большой и сверхбольшой степени интеграции это было абсолютно бессмысленной затеей. Благодаря достижениям в области микроэлектроники последних лет стало возможным массовое производство в виде микросхем сложнейших устройств, таких как центральный процессор вычислительной машины, оперативное и постоянное запоминающие устройства и т. д. Поскольку в большинстве случаев электрические параметры и функциональное назначение выходов и входов этих устройств (блоков) стандартизованы, их довольно легко соединять друг с другом. К тому же при разработке этих блоков, как правило, предусматривается стандартный вариант их применения, использование которого значительно упрощает проектирование устройств на их основе. Проектирование и построение микро-ЭВМ напоминает игру в детский конструктор, где все детали подходят друг к другу и можно воспользоваться руководством, в котором предложены некоторые типовые варианты узлов и изделий из его элементов. Для построения простейшей машины потребуется всего несколько узлов, создать которые не так уж трудно.

Итак, построение простейшей микро-ЭВМ оказывается сейчас возможным и не очень сложным делом. По крайней мере оно не сложнее постройки любительских конструкций в области радио, телевидения или звукозаписи.

"А можно ли построить самому не простейшую, а более сложную микро-ЭВМ?" — спросит заинтересованный читатель.

Простейший вариант микро-ЭВМ допускает возможность усложнения и усовершенствования конструкции путем замены или установки дополнительных микросхем или новых дополнительных плат с микросхемами. Можно повысить быстродействие микро-ЭВМ, увеличить объем памяти или заставить ее выполнять новые, не предусмотренные первоначальной конструкцией функции. К существенному расширению возможностей простейшей микро-ЭВМ приведет, например, включение в ее состав перепрограммируемой памяти, т. е. постоянной памяти, сохраняющей информацию при выключении питания и программируемой пользователем, с возможностью стирания информации и повторного программирования. Поскольку более сложная микро-ЭВМ, как мы увидим из дальнейшего изложения (см. § 2.3), отличается от простейшей, кроме всего прочего, развитой периферией, можно заняться совершенствованием ее внешних устройств. Однако внешние устройства самому построить довольно сложно. Вряд ли, например, кто-нибудь захочет взяться за конструирование хорошего печатающего устройства. Изготовление подобного устройства под силу лишь промышленности. Вот подсоединить к простейшей микро-ЭВМ имеющиеся внешние устройства можно, в том числе некоторые бытовые приборы, такие как домашний телевизор или кассетный магнитофон. О том, как это сделать, вы узнаете в гл. 10.

КАКУЮ МИКРО-ЭВМ МЫ БУДЕМ СТРОИТЬ?

2.1. ОСНОВНЫЕ БЛОКИ МИКРО-ЭВМ

Будем представлять описываемую далее микро-ЭВМ системой вложенных друг в друга блоков наподобие матрешек и открывать каждый раз лишь тот из них, который будет нужен в момент изложения соответствующего материала. Так, например, сейчас нас будет интересовать только внешний блок (собственно микро-ЭВМ), имеющий вполне определенное число входов и выходов. Следующий, расположенный внутри него блок назовем пока центральным блоком. О содержимом центрального блока и о том, как он связан с внешним блоком, будет показано чуть позже.

Основным назначением внешнего блока является преобразование дискретной информации. Общий вид простейшего преобразователя информации представлен на рис. 2.1,д. На его входы поступает исходная информация, а на выходах появляется информация, преобразованная в соответствии с законом, реализуемом в преобразователе.

В простейших преобразователях закон преобразования информации остается неизменным и применяется к любому конкретному виду информации, на работу с которой рассчитан преобразователь данного вида. Более широкими функциональными возможностями обладают преобразователи с законом преобразования, изменяемым путем подачи специальных управляющих воздействий. На рис. *2.1,6* представлен общий вид такого преобразователя, отличающегося от простейшего наличием специальных управляющих входов.

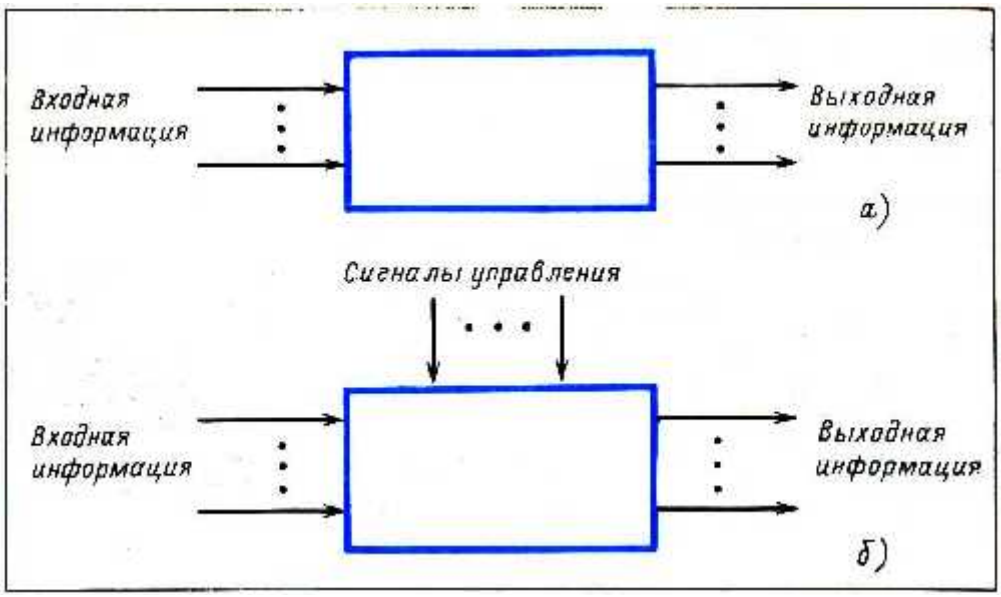


Рис. 2.1. Преобразователи информации: *а* - простейший; *б* – управляемый

Различают два типа управляемых преобразователей. В преобразователях первого типа управляющие воздействия неизменны в течение всего времени преобразования поступившей информации. В преобразователях второго типа в процессе преобразования управляющие сигналы могут изменяться, настраивая каждый раз преобразователь на выполнение какой-то одной функции. Для преобразования дискретной информации, особенно в том случае, когда сложный процесс преобразования может быть разбит на ряд этапов, каждый из которых характеризуется вполне определенной функцией преобразования, как правило, используются преобразователи второго типа.

Любую вычислительную машину можно рассматривать как управляемый преобразователь входной информации в выходную со следующей оговоркой. В процессе многоэтапного преобразования информации настройка преобразователя выполняется автоматически по заранее составленной пользователем схеме (детальной последовательности преобразований) с учетом результатов преобразований на каждом из этапов. Отсюда следуют два важных факта.

1. Пользователь, решающий на вычислительной машине свою задачу, должен заранее составить эту детальную последовательность преобразований исходных данных, называемую программой решения задачи.

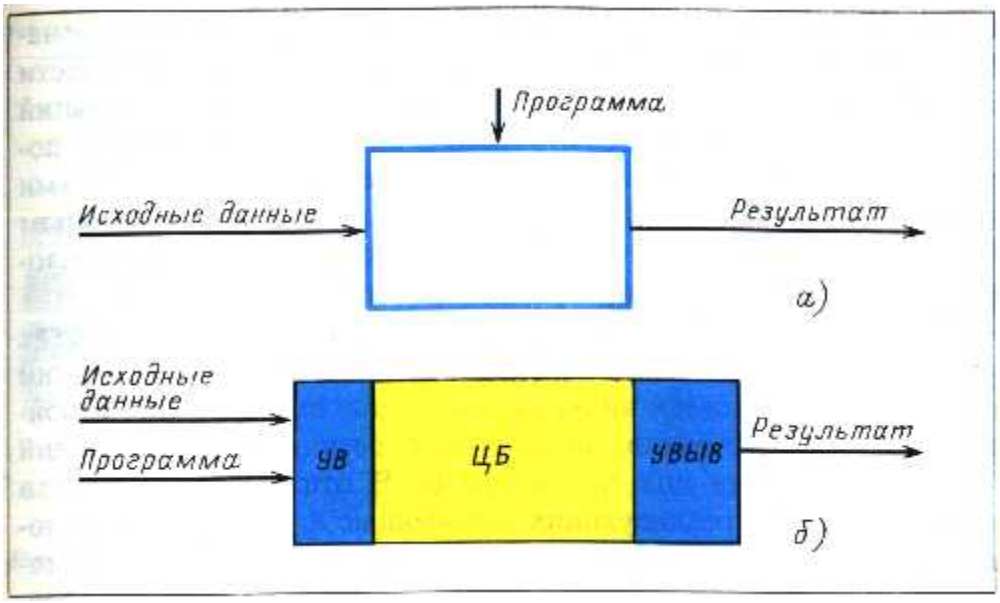


Рис. 2.2. Микро-ЭВМ как преобразователь

2 Чтобы преобразование выполнялось по мере решения задачи автоматически, программа решения задачи должна быть введена в машину до начала ее работы над задачей и должна храниться там в течение всего времени вычислений. Кроме того, должна быть предусмотрена возможность хранения тех промежуточных результатов вычислений, от которых зависит настройка преобразователя (работа машины). С учетом этих замечаний схема преобразования информации с помощью машины приобретает вид, указанный на рис. 2.2,д. А наш внешний блок кроме отмеченного раньше центрального блока (ЦБ) должен содержать устройство ввода (УВ) для ввода данных и программы и устройство вывода (УБЫВ) для выдачи результатов вычислений (рис. 2.2,6). В чем же состоит работа центрального блока и какие устройства в него входят?

2.2. СОДЕРЖИМОЕ ЦЕНТРАЛЬНОГО БЛОКА

Все уже привыкли к тому, что ЭВМ предназначена для вычислений. Об этом говорит ее название Тем не менее это справедливо лишь отчасти. С расширением области применения микро-ЭВМ собственно вычислительные функции в ее работе занимают довольно скромную долю среди всех остальных функций. Действительно, такие задачи, как анализ текстовых и речевых сообщений, поиск требуемых данных в массиве, преобразования массивов данных, распознавание образов и обработка изображений, строго говоря, отнести к вычислительным нельзя, хотя сам процесс преобразований информации можно рассматривать как вычислительный, поскольку в нем реализуются операции над двоичными кодами или числами. По существу микро-ЭВМ является универсальным преобразователем дискретной информации, причем преобразователем особого вида — программируемым.

Процесс преобразования информации начинается в устройстве ввода. Устройство ввода предназначено для преобразований входной информации к виду, удобному для вьиислений. Устройство вывода, напротив, преобразует результаты вычислений к виду, удобному для пользователя. В отличие от устройств ввода и вывода, реализующих неизменные функции преобразования, центральный блок микро-ЭВМ реализует самые разнообразные функции и является поэтому универсальным преобразователем, осуществляющим программный принцип обработки информации.

Любой достаточно сложный процесс преобразования дискретной информации можно разбить на отдельные этапы или акты. Элементарный неделимый акт обработки информации называют *операцией,* а управляющее слово, вызывающее выполнение этой операции, — *командой.* Последовательность команд, реализующих требуемый процесс преобразования информации, составляет *программу* обработки исходных данных. Программный принцип обработки информации позволяет использовать одно и то же устройство — универсальный преобразователь — для решения самых разнообразных задач при помощи составленных пользователем последовательностей команд или программ преобразования. Как уже отмечалось, программа должна быть введена в машину до начала вычислений. В связи с этим следующий блок микро-ЭВМ (центральный) должен содержать по крайней мере следующие два функциональных блока: процессор, реализующий операции преобразования, и память, хранящую программу и результаты вычислений (рис. 2.3).

Благодаря тому что все осуществляющие управление преобразованием команды записываются в память, программный способ преобразования информации является очень гибким. Процессор, извлекая из памяти команды (рис. 2.3), может оперировать с ними как с числами и, изменив, возвращать их обратно в память. Это позволяет реализовать сложные схемы вычислений путем использования команд, которые в процессе вычислений "сами себя меняют", вследствие чего меняется весь ход вычислительного процесса.

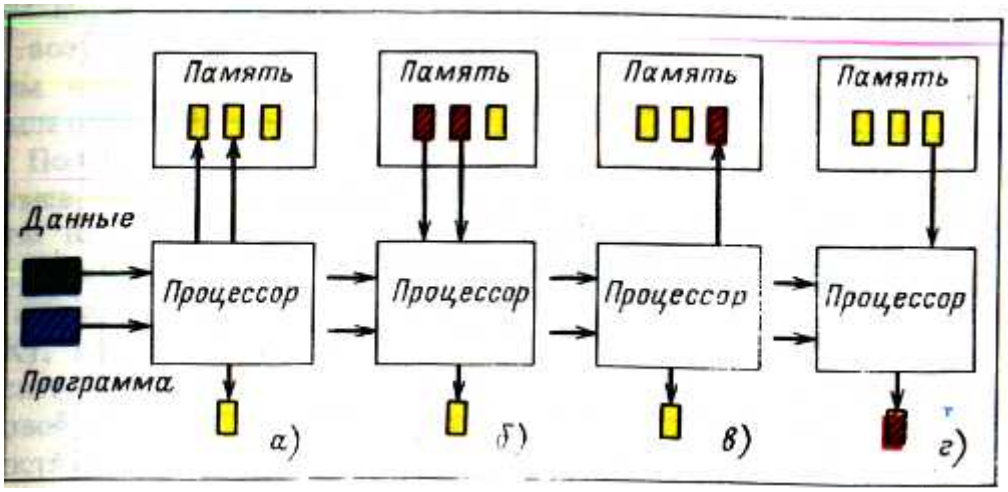


Рис. 2.3. Схема процесса решения задачи:
а — программа и данные на входе микро-ЭВМ; *б* — программа и данные в памяти машины; в - результат вычислений занесен в память; *г* -результат на выходе микро-ЭВМ

Память, содержимое которой изменяется процессором и в которую записываются команды и данные, а также заносятся промежуточные и окончательные результаты вьиислений, называется *оперативной памятью* или оперативным запоминающим устройством. Кроме нее в составе центрального блока микро-ЭВМ должна быть память с неизменным содержанием. Эта память, используемая только для считывания хранимой в ней информации, называется *постоянной памятью* или постоянным запоминающим устройством. Ее содержимое не пропадает при выключении питания, и изменить его с помощью каких-либо команд пользователь не может. Чтобы это сделать, необходимо воспользоваться специальным устройством, называемым *программатором.*

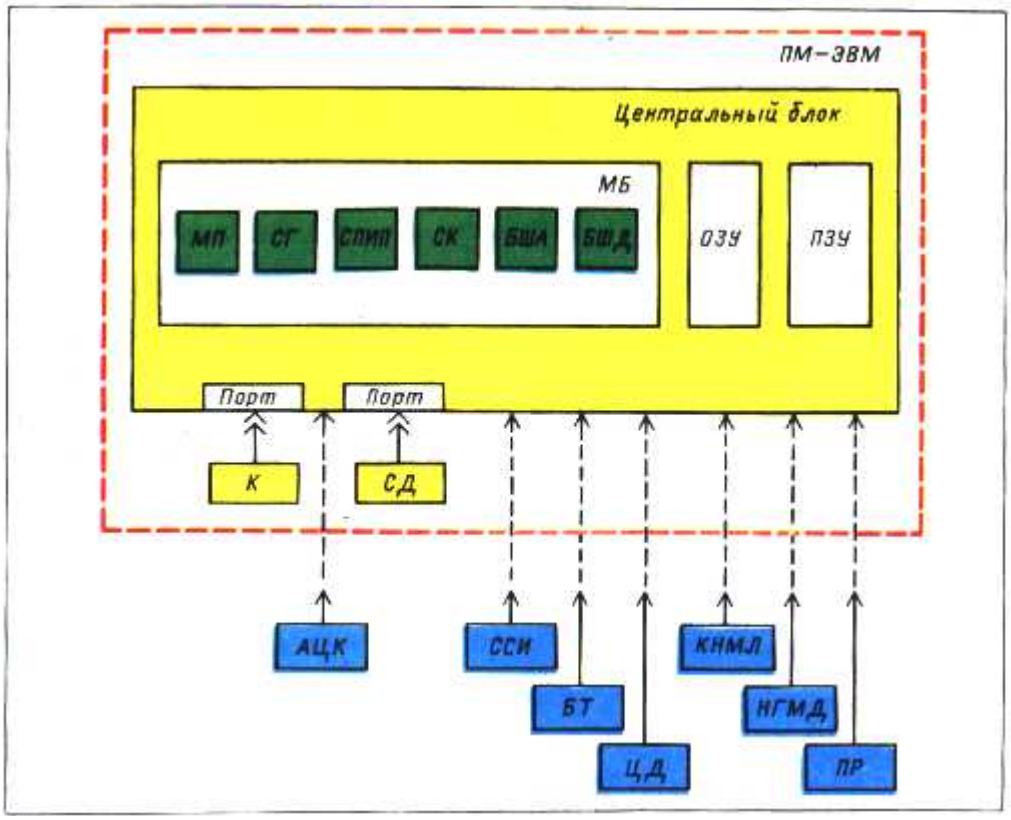


Рис. 2.4. Простейшая микро-ЭВМ и некоторые возможности ее расширения

Непосредственное управление процессом обработки информации в соответствии с командами программы пользователя осуществляется специальной схемой, входящей в состав процессора. Кроме того, в процессе управления микро-ЭВМ принимают участие следующие специальные устройства: синхро-генератор, синхронизирующий работу всех блоков вычислительной машины, системный контроллер (устройство, формирующее сигналы управления из сигналов процессора) и схема пошагового исполнения программы.

Итак, следующий рассматриваемый нами блок (рис. 2.4), называемый центральным блоком микро-ЭВМ, состоит из микропроцессорного блока (МБ), оперативного запоминающего устройства (ОЗУ), постоянного запоминающего устройства (ПЗУ) и вводных и выводных согласующих устройств связи, называемых портами. Порт — это устройство сопряжения, с которым микропроцессорный блок обменивается информацией аналогично обмену с устройствами памяти. В свою очередь МБ содержит: микропроцессор (МП), синхрогенера-тор (СГ), схему пошагового исполнения программы (СПИП), системный контроллер (СК) и специальные устройства, называемые буферами тин адреса (БША) и данных (БШД) и описываемые ниже.

2.3. КАКУЮ МИКРО-ЭВМ МЫ БУДЕМ НАЗЫВАТЬ "ПРОСТЕЙШЕЙ"?

Даже простое перечисление основных узлов микро-ЭВМ говорит о том, что современный дискретный универсальный вычислитель — это довольно сложное устройство. Ниже опишем простейший вариант такого вычислителя, который будем именовать простейшей микро-ЭВМ и возможность построения которого конструктором-любителем из доступных деталей, выпускаемых промышленностью, была оговорена ранее.

Под простейшей микро-ЭВМ (далее ПМ-ЭВМ) будем подразумевать микро-ЭВМ на одной или нескольких платах, построенную на базе микропроцессора КР580ИК80А (КР580ВМ80А), с минимальным (определяемым ниже) объемом ОЗУ и ПЗУ и с простейшими устройствами ввода/вывода в виде клавиатуры (К), включающей 16 клавиш и 24 светоизлучающих диода (светодиода СД). На рис. 2.4 ПМ-ЭВМ обведена красной штриховой линией. На этом же рисунке указаны некоторые возможности функционального расширения ПМ-ЭВМ.

Микро-ЭВМ может быть оснащена алфавитно-цифровой клавиатурой (АЦК), семисегментными светоизлучающими индикаторами (ССИ), специальным цветным дисплеем (ЦД) или дисплеем на базе бытового телевизора (БТ), внешними запоминающими устройствами — кассетным магнитофоном (кассетным-накопителем на магнитной ленте КНМЛ или накопителем на гибких магнитных дисках НГМД), печатающим устройством — принтером (ПР) и некоторыми другими устройствами. Подключение перечисленных дополнительных внешних устройств потребует разработки специальных согласующих схем, не показанных на схеме ПМ-ЭВМ (см. гл. 10).

Описываемая простейшая микро-ЭВМ может быть использована для приобретений навыков программирования на языке Ассемблер в мнемонических кодах. Она может также оказаться полезной при отладке небольших программ, для макетирования простейших управляющих устройств и, как уже отмечалось, для несложных расчетов в качестве программируемого калькулятора. Такая несложная машина может найти применение в школе, профессиональном техническом училище, на факультетах переквалификации специалистов с высшим образованием.

Однако следует заметить, что изготовление этой простейшей вычислительной машины все же потребует известного напряжения ума и сил. Поэтому, если читателю необходимо лишь средство для выполнения простейших расчетов, овчинка выделки не стоит. В таком случае лучше закрыть эту книгу и приобрести один из калькуляторов, имеющихся в продаже. Если же вы хотите приобщиться к увлекательному миру микроэлектроники и построить себе электронного помощника, функции которого можно будет в дальнейшем расширять путем постепенного совершенствования вашей конструкции, приступайте к чтению следующей главы.

3

НЕКОТОРЫЕ ОБЩИЕ СВЕДЕНИЯ О РАБОТЕ МИКРО-ЭВМ

3.1. ДАННЫЕ И ПРОГРАММЫ

Содержание этой главы носит в основном справочный характер. В ней приведены лишь основные понятия и самые общие сведения о работе микро-ЭВМ, без которых неподготовленному читателю будет трудно проследить работу основных ее узлов, описанию которых посвящены последующие главы книги. Читатель, знакомый с основами построения ЭВМ и программирования, может сразу перейти к следующей главе.

Как уже говорилось в предыдущей главе, чтобы решить задачу на машине, нужно ее запрограммировать; т. е. составить определенную последовательность команд (программу), которая, вместе с данными должна быть введена в память машины. В процессе решения задачи центральный процессор обращается к памяти машины, выполняет команды, извлеченные из памяти, обрабатывает в соответствии с этими командами данные, извлеченные из той же памяти и полученные от внешних устройств, и в зависимости от результатов обработки переходит к выполнению одной или нескольких других команд. Последовательность команд, приводящая к решению задачи, называется программой решения задачи. Каждая машина характеризуется определенным набором операций (системой команд), или машинным языком, которые должны быть известны пользователю, решающему свою задачу на данной машине.

Чтобы машина могла воспринимать передаваемые ей команды и данные, они должны быть представлены в двоичной форме. С этой целью каждой команде ставится в соответствие двоичный код, а все числовые значения выражаются в двоичной системе счисления.

Под двоичной системой счисления подразумевается позиционная весомозначная система с основанием 2 и с цифрами 0,1. Термин "позиционная весомозначная" означает, что в зависимости от положения цифры в числе ей приписываются разные значения, или вес. В наиболее распространенных системах счисления этот вес равен степени основания, показатель которой ра-вен n — 1, где *n* — номер разряда, отсчитываемый справа налево. Системы счисления получают наименование в зависимости от основания. Так, в десятичной системе счисления основанием является 10, в двоичной — 2, в восьмеричной — 8, в шестнадца-теричной — 16 и т. п. При этом количество используемых цифр для представления чисел равно основанию системы счисления. В двоичной системе используются всего две цифры: 0 и 1. В десятичной системе используются 10 цифр от 0 до 9, в восьмеричной — восемь цифр, т. е. первые восемь цифр десятичной системы (от 0 до 7). В шестнадцатеричной системе счисления используются все цифры десятичной системы, а в качестве недостающих шести цифр используются первые шесть букв латинского алфавита: A, B, C, D, E, F. Примеры записи десятичных чисел от 0 до 16 и от 248 до 255 в двоичной, восьмеричной и шестнадцатеричной системах счислений приведены в табл. 3.1. Десятичное число 255 является наибольшим представимым в двоичной системе счисления при условии использования всего восьми разрядов для записи чисел. Для представления чисел, больших 255, необходимо в двоичной системе счисления использовать большее число разрядов.

Чтобы отличить число, записанное в той или иной системе счисления, от числа, записанного в другой системе счисления, в конце записи обычно ставят соответствующие используемому основанию цифры или буквы (иногда в виде индексов, например 38_{10}). Для десятичной системы счисления часто используется буква D, для двоичной — B, для восьмеричной — Q, для шестнадцатеричной — H. Например:

216 D = 11011000 B = 330 Q = D8 H.

Как следует из табл. 3.1, наиболее громоздким из рассмотренных представлений, в особенности для больших чисел, является двоичное представление, а наиболее компактным — шестнадца-теричное представление, не совсем удобное для восприятия. Чтобы воспользоваться числовыми результатами в шестнадцатеричной системе счисления, необходимо перевести их в более привычную десятичную форму. Способы перевода чисел из одной системы счисления в другую неоднократно описывались в литературе и здесь рассматриваться не будут.

Таблица 3.1

Представление чисел			
Десятичное	Двоичное	Восьмеричное	Шестнадцате-ричное
0	00000000	000	00
1	00000001	001	01
2	00000010	002	02
3	00000011	003	03
4	00000100	004	04
5	00000101	005	05
6	00000110	006	06
7	00000111	007	07
8	00001000	010	08
9	00001001	011	09
10	00001010	012	OA
11	00001011	013	0B
12	00001100	014	OC
13	00001101	015	OD
14	00001110	016	OE
15	00001111	017	OF
16	00010000	020	10
248	11111000	370	F8
249	11111001	371	F9
250	11111010	372	FA
251	11111101	373	FB
252	11111100	374	FC
253	11111101	375	FD
254	11111110	376	FE
255	11111111	377	FF

Необходимое для выполнения программы на ЭВМ двоичное представление данных и команд для пользователя неудобно именно своей громоздкостью. Написание программы непосредственно в двоичных кодах утомительно и нередко приводит к ошибкам. Шестнадцатеричное представление компактно, но

не совсем удобно ввиду непривычности использования буквенных обозначений числовых величин.

Весьма распространенным способом представления данных и команд является их восьмеричное представление. Оно довольно компактно и легко переводимо в двоичную форму. Для этого достаточно запомнить двоичные коды трех первых двоичных разрядов (см. табл. 3.1). Перевод двоичного числа в восьмеричное производится следующим образом. Двоичный код разбивается справа налево на триады, каждая триада считается самостоятельным двоичным кодом трехразрядного числа и заменяется соответствующей цифрой от 0 до 7. Перевод восьмеричного кода в двоичный производится аналогичным образом: каждая цифра от 0 до 7 заменяется соответствующим двоичным кодом.

Пусть, например, требуется записать восьмеричный код восьмиразрядного двоичного числа 10110101 В. Разбитый на триады двоичный код будет 10 НО 101 В. После замены каждой триады двоичным кодом получим 265 Q. Двоичное представление восьмеричного числа 312Q получается также просто: 11 001 010 В или 11001010 В.

Представление двоичных чисел в табл. 3.1 в виде восьмиразрядных кодов приведено не случайно. Наименьшей единицей информации, которая может быть представлена в ЭВМ, является двоичный разряд или *бит.* Бит может иметь два значения: О или 1, а соответствующий ему электрический сигнал — два уровня напряжения. Группа из восьми двоичных разрядов называется *байтом.* Более крупными единицами информации являются килобит ($1024 = 2^{10}$ бит, или двоичных разрядов), килобайт ($1024 = 2^{10}$ байт) и мегабайт ($1048576 = 2^{20}$ байт). Иногда выделяют группу из четырех двоичных разрядов, называемую *ниблом.* Этим понятием пользуются при рассмотрении процессов выполнения арифметических операций в двоично-десятичных кодах.

Остановимся теперь на определении понятия машинной команды как элемента записи машинной программы.

Под командой подразумевается указание, записанное на машинном языке микро-ЭВМ и определяющее ее действия при выполнении отдельной операции или части вычислительного процесса. Команда может быть представлена в символической форме или в форме кодов (шестнадцатеричных, восьмеричных, двоичных). Таким образом, под программой подразумевается фиксированная последовательность команд, воспринимаемых машиной как единая целая группа указаний, позволяющая решить поставленную пользователем задачу.

В качестве формального языка для описания данных и программ их обработки на микро-ЭВМ используются не только машинный язык, но и специальные языки программирования, такие как, например, БЕЙСИК. Программа на таком языке, называемом иногда *языком высокого уровня,* не может непосредственно восприниматься микропроцессором. В совершенных ЭВМ специальная программа, называемая *интерпретатором* или *компилятором,* преобразует программу на языке высокого уровня в эквивалентную ей программу на языке машины.

Программа, написанная на языке Ассемблер, переводится пользователем в машинные восьмеричные коды с помощью таблицы кодов (табл. 4.1 — 4,3) и вводится вместе с данными также в восьмеричных кодах в ПМ-ЭВМ с клавиатуры и размешается в оперативном запоминающем устройстве, в котором выделяется специальная область, где пользователь может разместить свою программу.

Процесс размещения программы в памяти может быть легко представлен с помощью простой модели запоминающего устройства, имеющей вид обычного книжного стеллажа или открытой полки, разделенной на отдельные ячейки. Каждой ячейке присвоен номер, называемый адресом. В ячейку можно поместить данные ограниченного объема: один байт или восьмиразрядный двоичный код. Команда, занимающая три машинных слова (три байта), может быть размещена лишь в трех ячейках памяти. При этом первый байт размещается в ячейке по указанному адресу, а остальные два — в последующих ячейках памяти. Важно при этом правильно поместить начальную команду программы, которая должна находиться в той ячейке, с которой микропроцессор начинает свою работу.

Программа для ПМ-ЭВМ представляет собой последовательность указанных кодов, из которых левый код — адрес ячейки памяти, а правый — команда или данные.

Предположим, что соответствующие программе и исходным данным найденные коды введены в ПМ-ЭВМ, На чем основаны процессы преобразования всей этой информации в машине?

3.2. ОСНОВНЫЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Любое преобразование кодов в машине основано на выполнении над ними логических операций. К логическим относятся операции над наборами двоичных (принимающих только значения 0 и 1) аргументов, в результате выполнения которых получают единственное значение: 0 или 1. Логические операции могут быть описаны с помощью специальных функций, принимающих 0 и 1 на наборах двоичных переменных и называемых *булевыми функциями.* Число всевозможных булевых функций от n переменных равно 2 и, следовательно, растет очень быстро с ростом n. Для двух переменных число всех булевых функций равно 16, а для одной переменной — 4. Функции одной и двух переменных играют важную роль в теории переключательных схем (схем, реализующих логические преобразования дискретных сигналов).

Таблица 3.2

Значение аргумента	Значения функций			
X	f_1	f_2	f_3	f_4
0	0	1	0	1
1	0	1	1	0

Множество всех булевых функций от и переменных может быть описано конечной таблицей, число строк которой равно 2", а число столбцов — 2^2 . Например, все четыре булевы функции от одной переменной могут быть описаны табл. 3.2.

Первые две функции не требуют для своей реализации специальных аппаратурных затрат, так как первая f_1 (носящая название "константа 0") соответствует разрыву цепи передачи сигнала, а вторая f_2 ("константа 1") — постоянному соединению. Третья функция f_3 носит название "функции повторения" (имеется в виду повторение значения аргумента), а последняя f_4 — "функции отрицания", или "функции инверсии".

В электрических цепях элемент, называемый *буфером* и служащий для развязки цепей и согласования нагрузок, может реализовывать функцию f_3, а так называемый *инвертирующий буфер* — функцию инверсии f_4. Функцию f_3 могут выполнить два последовательно включенных инвертора, т. е. элементы, реализующие функцию инверсии сигнала f_4. Это свойство инвертора использовано в схеме буфера шины адреса ПМ-ЭВМ (см. §6.3).

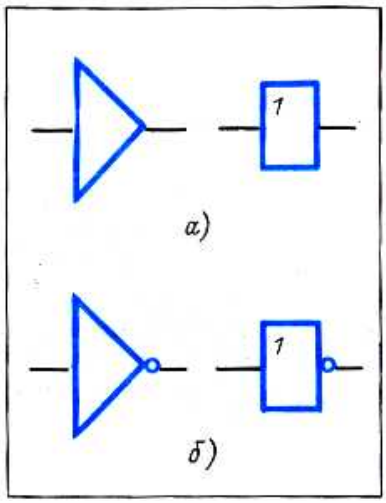


Рис. 3.1. Схемные обозначения буфера *(а)* и инвертора *(б)*

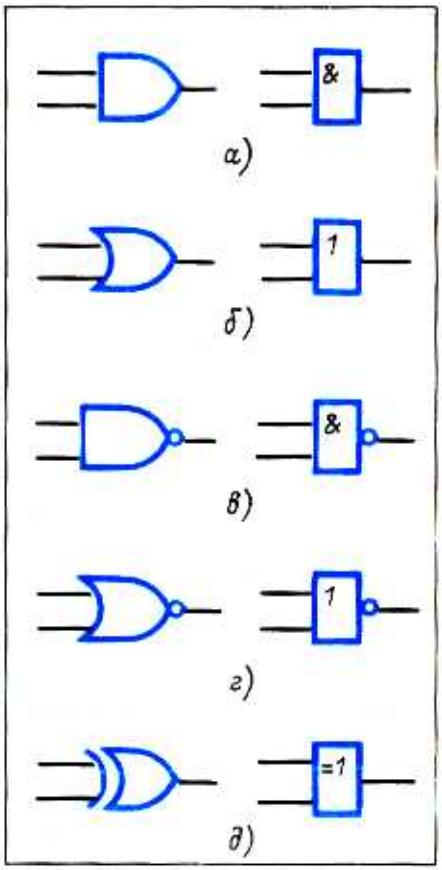


Рис. 3.2. Схемные обозначения:
а — элемента И; *б* — элемента ИЛИ; в - элемента И-НЕ; г -элемента И ЛИ-НЕ; *д* — элемента ИСКЛЮЧАЮЩЕЕ ИЛИ

Схемное обозначение буфера приведено на рис. 3.1,*а,* а инвертора — на рис. *3.1,6.* Инвертор является весьма распространенным элементом логических схем, поскольку входит в состав других логических элементов, реализующих более сложные булевы функции. На схеме рис. 3.1 приведены по два схемных

обозначения буфера и инвертора. Первое обозначение используется в основном в англоязычной литературе, второе — в отечественной и рекомендовано ГОСТ.

Среди всех 16 возможных булевых функций от двух переменных наибольшее распространение получили следующие пять функций, описываемые в табл. 3.3.

Функция f_5 носит название *функции логического умножения (конъюнкции),* или функции И. Она реализуется с помощью логического элемента, называемого схемой И. Графическое обозначение этого элемента приведено на рис. *3.2,а.* К особенностям работы элемента И относится то, что сигнал на его выходе, соответствующий логической единице, появляется только в том случае, если аналогичные сигналы присутствуют одновременно на его двух входах. Если же хотя бы на одном входе элемента И сигнал нулевой, выход также принимает значение 0. Понятие функции (схемы) И естественным образом расширяется на *n* переменных (входов).

Функция f_6 носит название *функции логического сложения (дизъюнкции),* или функции ИЛИ. Она реализуется с помощью логического элемента, называемого схемой ИЛИ. Графическое обозначение элемента ИЛИ приведено на рис. *3.2,6.* В отличие от схемы И на выходе элемента ИЛИ единичный сигнал появляется при наличии аналогичного сигнала на любом входе. И только если на обоих входах элемента ИЛИ присутствуют нулевые сигналы, на его выходе формируется тоже нулевой сигнал. Аналогичным образом вводится понятие и-входового элемента ИЛИ.

Таблица 3.3

Значения аргументов		**Значения функций**				
X_1	X_2	f_5	f_6	f_7	f_8	f_9
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Значения функции f_7 противоположны значениям функции f на одних и тех же наборах входных сигналов. Поэтому функция f_7 называется функцией И-НЕ. Графическое обозначение соответствующего элемента дано на рис. 3.2,в. Его название - элемент И-НЕ.

Значения функции f_8 противоположны значениям функции f_6 на одних и тех же наборах. Поэтому функция f_8 носит название функции ИЛИ-НЕ. Она реализуется элементом с тем же названием, представленным на рис. *3.2,г.*

Наконец, последняя функция f_9 отличается от функции f_6 значением выходного сигнала на последнем наборе. Эта функция называется *логической неравнозначностью* или *суммой по модулю 2.* Выходной сигнал у соответствующего элемента, реализующего эту функцию (рис. 3.2,д), принимает значение, равное 1, в том и только в том случае, когда сигналы на его входах имеют противоположные значения. Элемент, реализующий функцию сложения по модулю 2, является одним из основных логических элементов схемы микропроцессорного блока микро-ЭВМ, поскольку, как мы увидим в дальнейшем, он является базой построения всех суммирующих схем. Кроме того, если выход элемента, реализующего функцию сложения по модулю 2 (другое название этого элемента - ИСКЛЮЧАЮЩЕЕ ИЛИ), подать на вход инвертора, то на выходе последнего сигнал l будет формироваться только при условии совпадения значений сигналов на входах элемента ИСКЛЮЧАЮЩЕЕ ИЛИ. Такой схемой можно воспользоваться для сравнения значений двух одноразрядных чисел. Эта операция в микро-ЭВМ используется очень часто.

Все элементы, реализующие рассмотренные основные функции, обладают одной характерной особенностью. Сигналы на выходах этих элементов полностью определяются входными сигналами, т. е. сигналами, присутствующими в рассматриваемый момент времени на их входах. От сигналов, подававшихся на входы элементов в более ранние моменты времени, выходные сигналы не зависят. Поэтому эти элементы относятся к классу комбинационных схем, или к классу схем, не обладающих памятью.

С помощью комбинационных элементов в ЦГО микро-ЭВМ реализуются все основные логические операции над 8-разрядными словами данных и 16-разрядными словами адресов. К ним относятся операции поразрядного логического умножения (поразрядное И), сложения (поразрядное ИЛИ), инвертирования (поразрядное НЕ), сравнения. С их помощью реализуются арифметические операции (см. § 3.3). Однако многие из этих операций были бы практически невыполнимы, если бы схема обработки двоичных сигналов не содержала специальных устройств для временного хранения данных.

К таким устройствам принадлежат регистры. Они состоят из элементов, число которых (или длина регистра) равно числу двоичных разрядов поступающих на них данных. Каждый элемент способен сохранить на своем выходе значение поступившего и затем пропавшего входного сигнала до тех пор, пока не возникнет необходимость сохранения вновь поступившей информации. Эти элементы, называемые

триггерами, относятся не к комбинационным схемам, а к схемам с памятью. Триггер — это схема с двумя устойчивыми состояниями: ВКЛЮЧЕНО, обозначаемое 1, и ВЫКЛЮЧЕНО, обозначаемое 0. Распространенным типом триггера является триггер с двумя входами, обозначаемыми R и S. Такой тип триггера носит название RS-триггера. Вход R (Reset — очистка) предназначен для перевода триггера в состояние 0, или, как говорят, для очистки триггера. При появлении сигнала на этом входе триггер переводится в состояние 0, если до этого он находился в состоянии 1, или остается в состоянии 0, если до этого он находился в состоянии 0. Вход S (Set — установка) предназначен для перевода триггера в состояние 1, или, как говорят, для установки триггера.

При появлении сигнала на этом входе триггер переводится в состояние 1, если до этого он находился в состоянии 0, или остается в состоянии 1, если до этого он находился в состоянии 1. Сигнал, переводящий триггер в то или иное состояние, может быть как единичным (в этом случае вход называется прямым) или нулевым (вход называется инверсным). Так, например, в схеме К155ТМ2 (см. гл. 5) сигналы сброса и установки имеют нулевые значения. В этой схеме триггер остается в состоянии 0 и в том случае, если значение сигнала изменится с 0 на 1. Перевести его в состояние 1 можно только подачей сигнала 0 на вход S (Set — установка). Состояние 1 триггер будет также сохранять при изменении значения сигнала на входе S на единичное.

Триггер типа RS является далеко не единственным и даже не наиболее применяемым в схемотехнике дискретных устройств. Более распространенным, например, является D-триг-гер. Он имеет два входа: D и C. Вход D является информационным, а C — управляющим. Сигнал со входа D переписывается в триггер только при наличии определенного сигнала на управляющем входе C. Некоторые триггеры, реализованные в виде микросхем, имеют входы, соответствующие как RS-триггеру, так и D-триггеру, и могут быть использованы как триггеры любого из указанных двух типов.

Состояние триггера однозначно соответствует сигналу на выходе, который называется прямым выходом. Кроме прямого выхода у триггера имеется так называемый инверсный выход, значение сигнала которого всегда противоположно значению сигнала прямого выхода. Это позволяет, например, иметь на выходах регистра, состоящего из восьми триггеров, одновременно прямой и обратный коды вводимого 8-разрядного двоичного числа.

Поскольку все процессы преобразования сигналов в ЭВМ синхронизированы, в схемах регистров используются синхронизируемые триггеры. В таких триггерах кроме информационных входов имеется специальный вход, на который подаются сигналы синхрогенератора. Изменение состояния триггера происходит только во время появления тактового импульса на его синхронизирующем входе. Изменение значений информационных сигналов в период пауз тактовых импульсов на состояние триггера влияния не оказывает.

Используемые для одновременного хранения нескольких двоичных разрядов регистры состоят из последовательно соединенных триггеров и управляющих связей между ними, позволяющих организовать последовательную или параллельную подачу запоминаемой информации. С помощью управляющих связей можно сдвигать хранящуюся в регистрах информацию на произвольное число разрядов вправо и влево, а также считывать информацию в последовательной и параллельной формах. Такие регистры называются сдвиговыми регистрами. Они широко используются для реализации арифметических функций, в частности умножения и деления.

Итак, в самом общем случае регистры позволяют осуществить следующее:

1) хранить поступившую на них двоичную информацию в течение необходимого времени;
2) осуществлять преобразование последовательных двоичных кодов в параллельные и обратно;
3) сдвигать при необходимости хранимые данные вправо и влево;
4) пользоваться обратным кодом хранимой информации, имеющимся на инверсных выходах триггеров.

3.3. ОСНОВНЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Рассмотренные в предыдущем параграфе логические операции и описываемые ниже арифметические операции реализуются в арифметическо-логическом устройстве (АЛУ) микропроцессора, на котором построена ПМ-ЭВМ. Операции выполняются над 8-разрядными двоичными числами, причем старший (левый) разряд может использоваться для представления знака числа (значение 0 указывает на положительное число, а 1 — на отрицательное).

Сложение двоичных чисел производится по тем же правилам, что и сложение десятичных чисел, за исключением того, что перенос в следующий разряд осуществляется при сумме в данном разряде, равной 2, а не 10. Пусть, например, требуется сложить два числа:

00011 010B = 26D
\+ 00001 100 B = 12D
00 100 110B=38D.

При сложении крайних правых четырех разрядов имели место все четыре возможные комбинации сложения одноразрядных чисел: 0+0=0, 1 + 0 = 1, 0+1 = 1, 1+1=0. В четвертом разряде сумма равна 2D, или 10 В. Следовательно, необходим перенос единицы из четвертого разряда в пятый. Тогда в пятом разряде опять же 1 + 1 = 0 и после переноса единицы в шестой разряд сумма в шестом разряде будет 1 + 0+0=1.

Нетрудно заметить, что реализация этих операций может быть выполнена с помощью всего двух логических элементов: суммы по модулю 2, или ИСКЛЮЧАЮЩЕГО ИЛИ, реализующего функцию f_9, соответствующую выходу суммы S, и элемента И, реализующего функцию f_5, соответствующую выходу переноса в следующий разряд С (рис. 3.3,с). Эта схема сложения одноразрядных чисел без учета входного переноса из предыдущего разряда называется *полусумматором.*

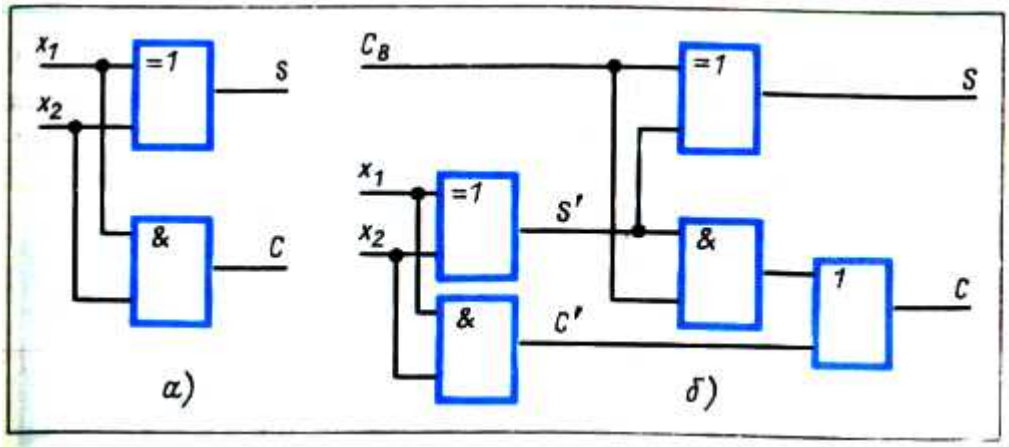


Рис. 3.3. Одноразрядные суммирующие схемы: *а* - полусумматор; *б* - полный сумматор

Чтобы реализовать сложение с учетом переноса из предыдущего разряда, необходимо использовать два полусумматора, соединив их так, как показано на рис. *3.3,6.* Эта схема носит название *полного сумматора,* или *одноразрядного сумматора.* Для сложения двух 8-разрядных двоичных чисел потребуется 8 одноразрядных сумматоров, соединенных таким образом, чтобы сигнал переноса передавался в каждый следующий разряд на вход $C_в$ соответствующего полного сумматора. Сигнал переноса из самого старшего разряда запоминается в специальном триггере, называемом *триггером флага переноса,* для указания переполнения, имевшего место при сложении.

Поскольку, как указывалось выше, старший разряд может отводиться под знак числа, с помощью 8-разрядной суммирующей схемы можно оперировать как с любыми целыми положительными числами в диапазоне от 00 000 000 В = OD до 11 111 111 В = 25 5 D, так и с целыми положительными и отрицательными числами в диапазоне от 01 111 111 В = 127D до 10 000 000 В = -128D (в этом случае единица в старшем разряде указывает на отрицательное число, а нуль - на положительное).

Для вычитания одного числа из другого используется специальное кодовое представление отрицательных чисел, называемое *дополнительным кодом.* Дополнительный код (иначе, дополнение до 2) получается прибавлением единицы к младшему разряду инверсного, или обратного, кода числа (дополнения До 1). Сложение уменьшаемого с вычитаемым, представленным в дополнительном коде, приводит к результату, который получился бы при обычном вычитании. Таким образом, не нужно строить специальную схему для операции вычитания, а можно

воспользоваться все той же схемой 8-разрядного сумматора. Пусть, например, требуется выполнить вычитание: 38 D — 26 D. Перейдем к дополнительному коду для числа — 26. Инвертируя код числа 26D и прибавляя единицу в младшем разряде, получаем:

00 011 010 — прямой код 11 100 101 — обратный код
\+
1
11 100 110 — дополнительный код. Теперь выполним сложение:
00100 110B + 11 100 110B
100001 100 В =12D.

В результате получили двоичный код десятичного числа 12 D и перенос из старшего разряда. Этот сигнал может быть использован при выполнении арифметических операций с 16-разрядными числами по

частям в одном 8-разрядном сумматоре. При этом сначала производится сложение в дополнительном коде младших разрядов суммируемых чисел и запоминание сигнала (единицы) переноса, а затем сложение старших разрядов с учетом этого сигнала переноса.

Для выполнения операций умножения и деления специальных команд в микропроцессоре КР580ИК80А не предусмотрено. Поэтому эти операции в ПМ-ЭВМ выполняются программным путем с использованием операций сложения, сложения с дополнительным кодом числа (вычитания) и сдвига (см. § 9.1).

Арифметические операции можно производить также над десятичными числами, закодированными так называемыми двоично-десятичными кодами (двоично-кодированное представление десятичного числа). При таком представлении чисел каждая десятичная цифра кодируется четырехразрядным двоичным кодом (кодом прямого замещения, или кодом 8421). После сложения двух двоично-кодированных чисел для получения правильного результата необходимо выполнить коррекцию результата этой операции. Для этого используется блок десятичной коррекции. Он осуществляет требуемую коррекцию результата путем исполнения специальной команды "десятичной коррекции" DAA (см. список команд в приложении 1), которую необходимо выполнить после команды сложения.

АРХИТЕКТУРА ПМ-ЭВМ И КОМПОНЕНТОВ

4.1. КОНСТРУКТИВНОЕ ОФОРМЛЕНИЕ ПМ-ЭВМ

Если пользователю микро-ЭВМ для работы с ней при программировании своих задач полезно все же иметь некоторое представление о содержимом того "черного ящика", с которым он имеет дело, то будущему конструктору, конечно уж, необходимо знать абсолютно все его "болты и гайки". В связи с этим в данной главе нам предстоит заглянуть внутрь каждого из блоков будущей ЭВМ и попытаться проанализировать его работу на функциональном уровне.

Как отмечалось в гл. 2, ПМ-ЭВМ состоит из центрального блока и устройств ввода/вывода. В качестве устройства ввода в ПМ-ЭВМ используется клавиатура, а устройства вывода -совокупность светодиодов. Центральный блок машины, как следует из рис. 2.4, состоит из центрального процессорного элемента, оперативного (ОЗУ) и постоянного (ПЗУ) запоминающих устройств и портов ввода и вывода. Работа всех перечисленных блоков синхронизируется сигналами от синхрогенератора, входящего в состав МБ.

Конструктивно ПМ-ЭВМ может быть оформлена в виде настольного прибора, на лицевой панели которого находятся клавиатура (К) и светодиоды (СД). Все компоненты центрального блока могут быть размещены на одной печатной плате (в описываемом варианте ПМ-ЭВМ — одноплатная машина), установленной внутри корпуса прибора.

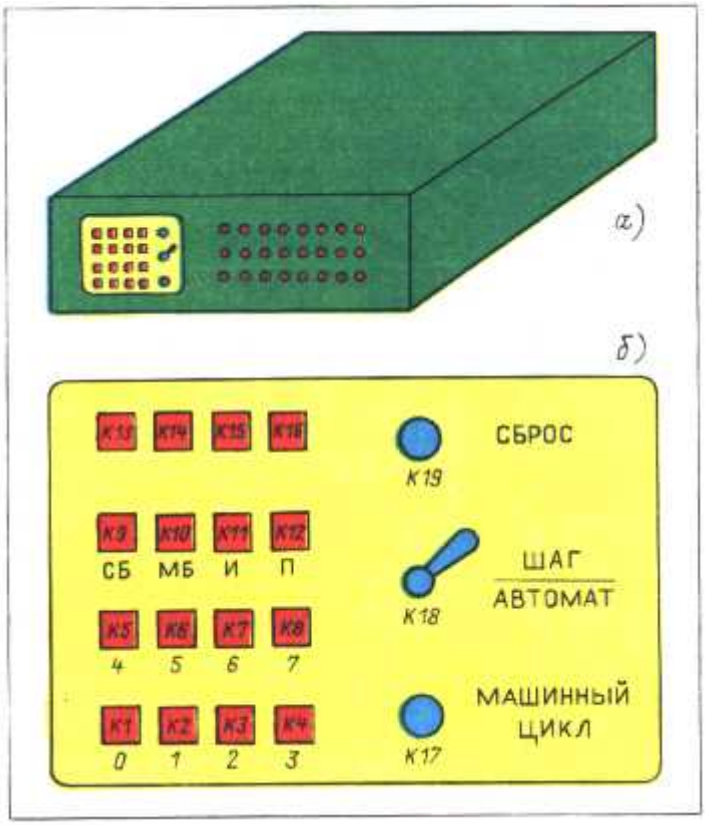


Рис. 4.1. Внешний вид ПМ-ЭВМ *(а)* и шестнадцатеричной клавишной панели ввода информации *(б)*

Внешний вид ПМ-ЭВМ при таком конструктивном оформлении приведен на рис. 4.1,д. На рис. 4.*1,6* указаны основные органы клавиатуры. В более простом варианте конструкции машины светодиоды также размещаются на плате, а клавиатура оформляется в виде отдельной выносной панели, соединенной с машиной с помощью жгута. Разъемное подсоединение *К* к машине может быть рекомендовано лишь при наличии хорошего разъема, обеспечивающего надежное соединение контактов. Если такого разъема нет, жгут лучше подсоединить наглухо путем пайки. В этом случае вся машина "укладывается" в размеры средней толщины папки для бумаг.

Клавиатура содержит 16 кнопок, из которых кнопки *К1- К8* служат для ввода данных, кнопки *К9 — К11 —* для управления вводом данных, а кнопка *К12 —* для управления запуском программы. Кнопки *К13-К16* не задействованы. Кроме того, имеются еще две управляющие кнопки и один переключатель. Кнопка *К17* предназначена для пошагового выполнения программы. При ее нажатии происходит выполнение одного машинного цикла (см. § 4.4). Переключатель *К18* предназначен для выбора режима работы ПМ-ЭВМ. Он имеет два положения: "шаг" и "автомат". В положении "автомат" переключатель замкнут и происходит автоматическое выполнение программы. В положении "шаг" программу можно пропустить в пошаговом режиме путем многократного нажатия кнопки "машинный цикл" *К17.* Кнопка *К19* ("сброс") предназначена для обнуления счетчика команд (см. § 4.3) .

Светодиоды компонуются в три группы по восемь штук в каждой. Они отображают состояния трех портов вывода. Схемы подключения светодиодов и клавиатуры описываются в § 7.3.

Прежде чем переходить к рассмотрению функциональной схемы ПМ-ЭВМ, поговорим о том, как организованы связи между блоками в вычислительной машине.

4.2. ОСНОВНЫЕ СВЯЗИ И СТРУКТУРА ШИН

В микро-ЭВМ, построенных на базе микропроцессоров, все связи между отдельными функциональными блоками осуществляются, как правило, так называемыми шинами. Под шиной подразумевается физическая группа линий передачи сигналов, обладающих функциональной общностью (по каждой линии передается один двоичный разряд информации). Так, например, данные в машине обычно передаются к различным ее функциональным узлам параллельно по восьми линиям. Физически шины реализуются в виде параллельных проводящих полосок печатной платы или в виде связанных в жгут проводов. Соответствующая группа из восьми линий передачи данных называется 8-разрядной шиной данных. Кроме шины данных в микро-ЭВМ выделяют шину передачи адресов, или шину адреса, и шину управления. Микро-ЭВМ с такой организацией связей относят к системам, обладающим архитектурой с тремя шинами.

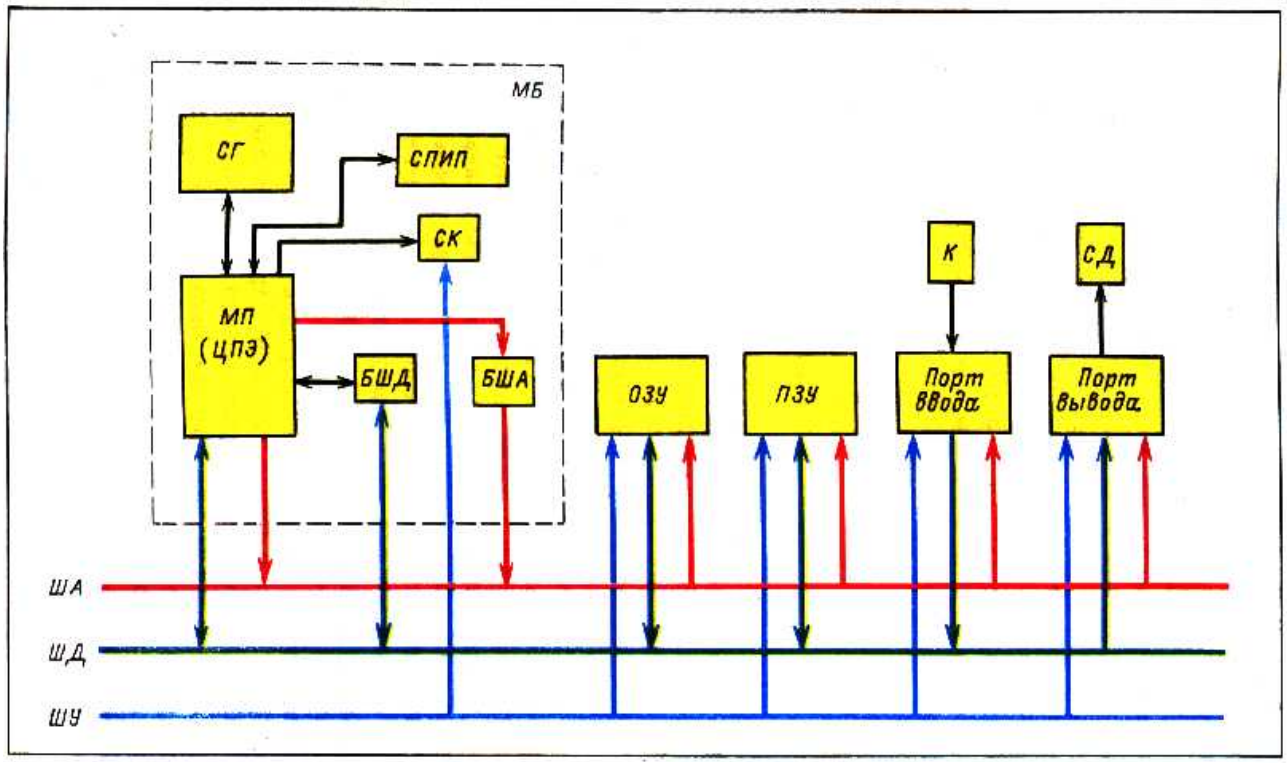


Рис. 4.2. Упрощенная архитектура ЭВМ с тремя шинами

Типовые связи в архитектуре ЭВМ с тремя шинами в общем случае будут иметь вид, представленный на рис. 4.2, если в качестве основных функциональных блоков машины использовать микропроцессорный блок (МБ), *ОЗУ, ПЗУ* и порты ввода/вывода. Линии шин адреса (ША) и управления (ШУ) являются однонаправленными [Здесь не рассматривается режим прямого доступа к памяти]. В них сигналы протекают в одном направлении — от центрального процессорного элемента ко всем остальным блокам. Шина адреса является 16-разрядной. Число линий шины управления определяется составом сигналов, формируемых системным контроллером.

Передаваемые по ША сигналы формируются в МП. Они необходимы для определения пути передачи внутри микро-ЭВМ, в том числе для выбора ячейки памяти, куда необходимо занести или откуда необходимо считать информацию. В определении тракта передачи данных могут принимать участие и управляющие сигналы, подсоединяющие или, напротив, блокирующие те или иные устройства микро-ЭВМ.

В отличие от *ША* и *ШУ* шина данных (ШД) является шиной двунаправленной. Данные по линиям шины могут передаваться от микропроцессора к какому-нибудь устройству микро-ЭВМ либо пересылаться в *МП* от какого-то устройства, доступ к которому обеспечивают сигналы адресной шины.

Естественно, что в каждый момент времени данные могут передаваться лишь в одном направлении, определяемом режимом работы микропроцессора. К основным режимам работы МП можно отнести: запись данных в память машины; чтение данных из памяти машины; пересылку данных в устройство ввода/вывода; чтение данных с устройства ввода/вывода; выполнение операций с содержимым внутренних регистров микропроцессора.

При реализации последнего режима внешние по отношению к МП шины микро-ЭВМ не используются, т. е. все действия происходят внутри МП. Реализация первых четырех режимов оказывает определяющее влияние на работу шины данных.

Работа по реализации программы любой микро-ЭВМ, построенной по типу архитектуры с тремя шинами, состоит в выполнении следующих действий для каждой команды программы.

1. Микропроцессор формирует адрес, по которому хранится код операции команды, переводя в соответствующее состояние шину адреса.
2. Код операции считывается из памяти по сформированному адресу и пересылается в микропроцессор.
3. Команда дешифрируется (идентифицируется) микропроцессором.
4. Микропроцессор "настраивается" на выполнение одного из перечисленных выше пяти основных режимов в соответствии с результатами дешифрирования считанного из памяти кода команды.

Перечисленные выше пять режимов являются основными, но не единственно возможными. Существуют и другие, рассматриваемые в гл. 6.

Перейдем теперь к окончательному оформлению функциональной схемы ПМ-ЭВМ.

4.3. ОБЩАЯ ФУНКЦИОНАЛЬНАЯ СХЕМА ПМ-ЭВМ

Кроме микропроцессорного блока, функциональное назначение которого мы уже определили, в состав ПМ-ЭВМ входят оперативное и постоянное запоминающие устройства и порты ввода/вывода. Все эти блоки связаны между собой через шины адреса и данных. Шина управления в ПМ-ЭВМ состоит всего из четырех линий передачи сигналов, причем в каждом режиме работы микропроцессора активна только одна линия.

Управление работой ПМ-ЭВМ осуществляется двумя блоками: схемой пошагового выполнения программы и схемой системного контроллера. Первая схема полезна для анализа исполнения вводимых команд программы. Системный контроллер необходим для организации трактов передачи информации при реализации основных режимов работы микропроцессора. Используемый в ПМ-ЭВМ микропроцессор КР580ИК80А имеет шины адреса и данных, а также некоторые управляющие сигналы. Сигналы шины управления формируются системным контроллером вне микропроцессора из управляющих сигналов МП. К ним относятся: ЧТЕНИЕ, ЗАПИСЬ, ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО, ВВОД С ВНЕШНЕГО УСТРОЙСТВА (см. § 6.3).

При организации шин необходимо правильно оценить величину токовой нагрузки по каждой шине. Если суммарный ток нагрузки линии в шине превышает допустимую величину на соответствующем выходе микропроцессора, то такую линию необходимо снабдить буфером. Под буфером подразумевается специальная схема, обеспечивающая электрическое согласование цепей передачи сигналов. Как уже отмечалось в гл. 3, простейшая схема буфера, не меняющая значение сигнала, состоит из двух включенных последовательно инверторов. Именно эта схема используется в ПМ-ЭВМ в формирователе сигналов шины адреса.

Линии шины данных в ПМ-ЭВМ также необходимо снабдить буферами. Однако поскольку шина данных является двунаправленной, буфер для нее также должен быть двунаправленным. Поэтому схему из двух последовательно включенных инверторов здесь использовать нельзя. Необходима специальная схема с использованием управляющего сигнала выбора направления передачи данных. Работа схемы двунаправленного буфера шины данных описана в § 6.3.

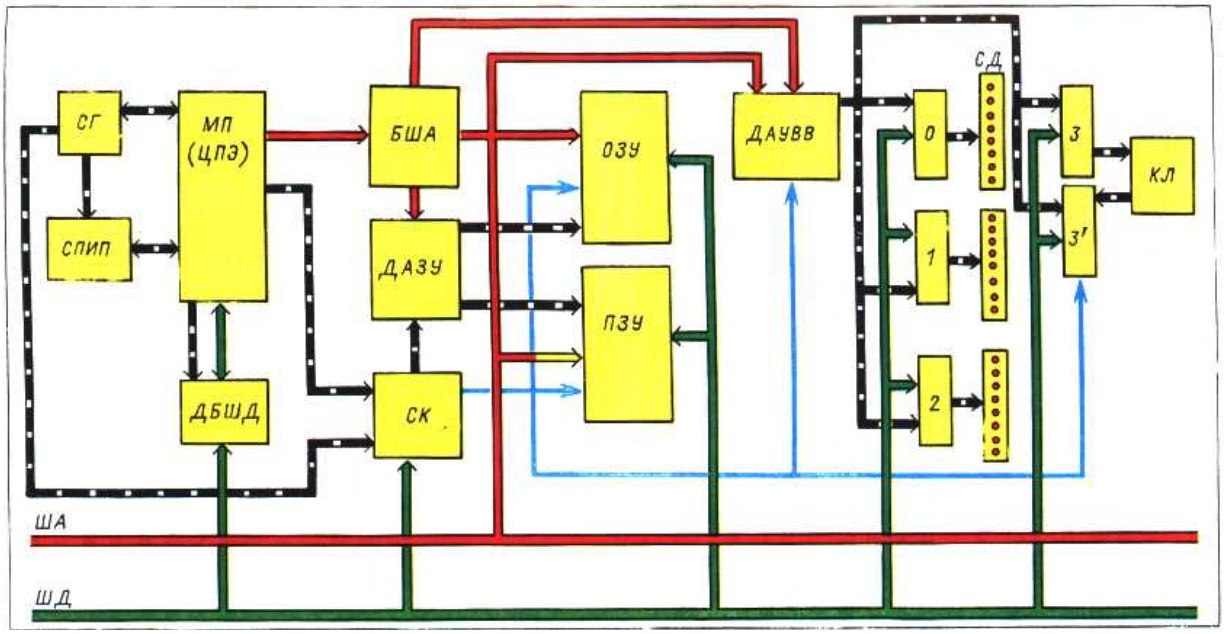


Рис. 4.3. Схема основных функциональных узлов ПМ-ЭВМ и связей между ними:

ДАЗУ, ДАУВВ - дешифраторы адреса ЗУ и УВВ; *ДБШД* - двунаправленный буфер ШД; *КЛ* - клавиатура; *СД* - светодиоды; *СК* — системный контроллер; *СПИП* — схема пошагового исполнения программы; *0, 1,2,3* — порты вывода; *3'* - порт ввода

Для организации правильной адресации к устройствам памяти и к устройствам ввода/вывода в схеме ПМ-ЭВМ используются еще два блока, не указанных на рис. 4.2: дешифратор адреса запоминающих устройств и дешифратор адреса устройств ввод а/выв од а. С учетом этих и всех остальных перечисленных выше дополнительных блоков общая функциональная схема ПМ-ЭВМ принимает вид, представленный на рис. 4.3. На этой схеме двойными линиями указаны шины передачи адреса (красными) и данных (зелеными), а сигналы управления — тонкими линиями. Расположение блоков в точности соответствует расположению их схем на общей электрической принципиальной схеме ПМ-ЭВМ, приведенной в приложении 2. На рис. 4.3 показаны лишь основные связи (например, отсутствуют связи по питанию).

Работа всех приведенных на рис. 4.3 блоков в отдельности описывается в гл. 6 и 7. Перейдем к рассмотрению функциональной схемы микропроцессора и организации его работы.

4.4. ФУНКЦИОНАЛЬНАЯ СХЕМА МИКРОПРОЦЕССОРА

При работе с ПМ-ЭВМ пользователю необходимо иметь информацию о числе и назначении всех регистров, специальных указателей, называемых флагами, и о системе команд МП. Число, назначение регистров, флагов и команд пользователь изменить не может. Он может изменять лишь содержимое регистров и использовать команды в любой нужной ему комбинации.

Как уже говорилось, под регистром подразумевается специальное запоминающее устройство, состоящее из элементов (триггеров) с двумя устойчивыми состояниями. Число элементов (восемь) соответствует одному байту. Большинство регистров микропроцессора 8-разрядные и лишь некоторые 16-разрядные. Все регистры разбиты на группы и отличаются различным функциональным назначением.

Основными блоками МП (рис. 4.4) являются: блок регистров общего назначения со схемой выборки регистров, регистр команд с дешифратором команд и формирователем машинных циклов, арифметическо-логическое устройство (АЛУ) с регистром А (аккумулятором), выполняющее арифметические и логические операции, регистр временного хранения данных W и Z (РВХД), флаговый регистр, устройство управления и синхронизации, буферы шины данных (БШД) и адреса (БША), буфер аккумулятора (БА), схема приращения и уменьшения (СПИУ).

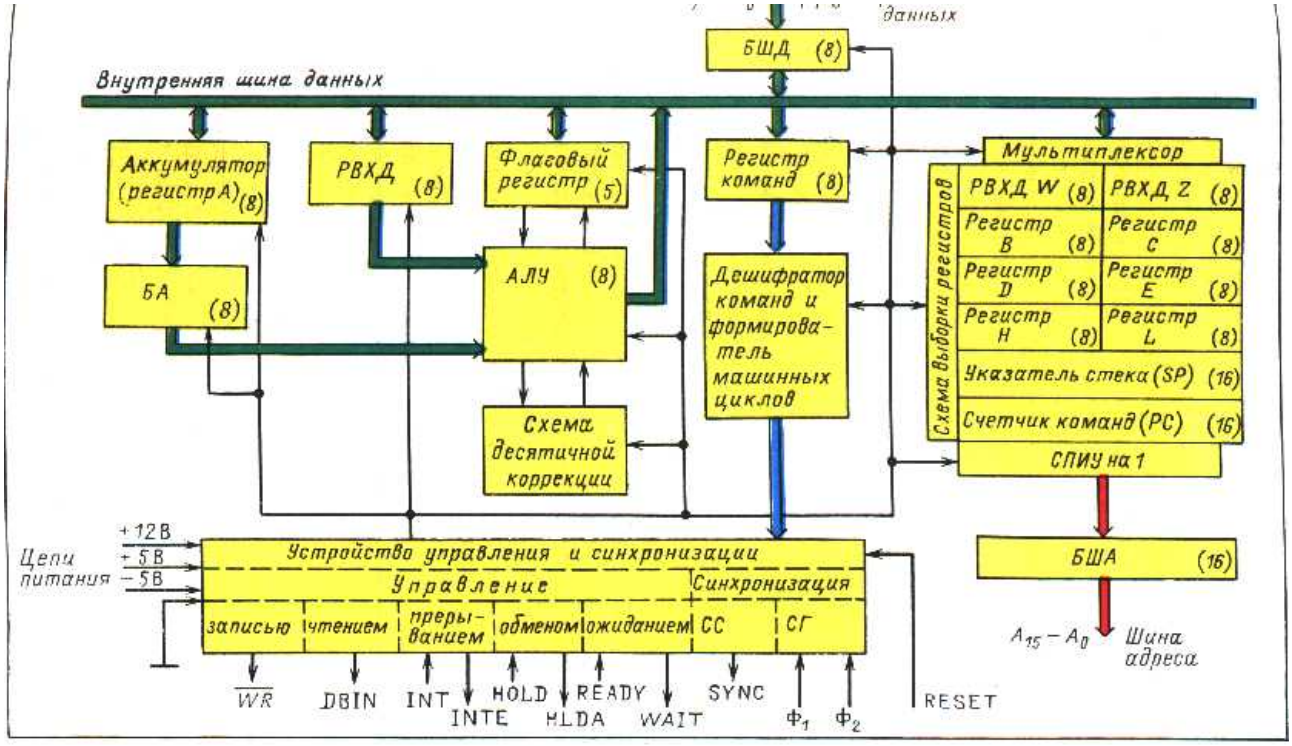


Рис. 4.4. Схема ЦПЭ на базе микропроцессора КР580ИК80А:
СГ - сигналы генератора тактовых импульсов; *СС* — сигнапы синхронизации

Доступными программисту являются следующие регистры: шесть 8-разрядных регистров, адресуемых по одному или парами (регистры B, C, D, E, H, L); один 8-разрядный регистр A, называемый аккумулятором; один 16-разрядный регистр, называемый указателем стека; один 16-разрядный регистр, называемый счетчиком команд, или программным счетчиком.

В некоторых специальных случаях могут быть доступными данные следующих двух регистров: регистра команд (8-разрядного) ; флагового регистра (5-разрядного).

Программно недоступными пользователю являются регистры W и Z. Они используются для временного хранения данных при выполнении команд микропроцессором.

Регистры общего назначения. Эти регистры размерностью в один байт обозначаются B, C, D, E, H, L. Они используются для хранения данных и промежуточных результатов вычислений, выполняемых с помощью арифметическо-логиче-ского устройства. При обработке 16-разрядных слов возможно обращение к парам регистров (B, C); (D, E); (H, L).

Аккумулятор — специальный однобайтовый регистр, обозначаемый A. При выполнении арифметических и логических операций служит источником одного из операндов и местом запоминания результата выполнения операции. Аккумулятор является основным операционным звеном арифметическо-логического устройства. Он служит также местом хранения данных и результатов операций, выполняемых в АЛУ.

Регистр команд — однобайтовый регистр, в котором хранится код выполняемой команды. Этот регистр непосредственно пользователю недоступен. Это означает, что не существует команды, которая бы явным образом могла изменить его содержимое. После выполнения очередной команды в регистр команд автоматически записывается код следующей команды из ячейки оперативной памяти, адрес которой содержится в счетчике команд.

Счетчик команд — двухбайтовый (16-разрядный) регистр, обозначаемый PC, или программный счетчик. Этот регистр хранит адрес следующей команды, которая должна быть выполнена вслед за предыдущей. Счетчик команд автоматически получает приращение хранимого в нем адреса в зависимости от того, какую по длительности команду (в один, два или три байта) микропроцессор считывает из памяти, указывая всегда на 1-й байт следующей команды. На содержимое этого регистра пользователь может повлиять только с помощью команд, изменяющих последовательное выполнение программы (например, команд безусловного перехода), а также с помощью некоторых специальных команд.

Указатель стека-двухбайтовый (16-разрядный) регистр, обозначаемый SP. Этот регистр хранит адрес очередной ячейки стека. *Стеком* называется особым образом организованный участок оперативной памяти, выделяемый программистом для временного хранения содержимого внутренних регистров МП со специальным режимом доступа. Эта область оперативной памяти необходима в том случае, когда нужно прекратить выполнение реализуемой последовательности команд и вернуться к ней позже, например для немедленного выполнения специальной подпрограммы или в результате прерывания программы. Данные от МП поступают в верхнюю часть стековой памяти, и при этом содержимое указателя стека уменьшается на единицу, чтобы всегда указывать на адрес последней заполненной ячейки стека (дно свободного пространства стека). Когда же данные выбираются (считываются) из стека, содержимое указателя стека увеличивается на единицу с каждым выбранным байтом. Эти операции со стеком называются *стековыми.* С их помощью легко организуются многоуровневые (вложенные) прерывания и программы обращения к подпрограммам из подпрограмм (к вложенным подпрограммам).

Флаговый регистр- регистр, содержащий 5 двоичных разрядов, называемых *флагами,* по числу хранимых в нем специальных признаков результатов некоторых операций. Иногда он называется регистром признаков или флаговым регистром битов условий. Значение флага указывает на результат выполнения какой-либо операции. Микропроцессор КР580ИК80А содержит флаговый регистр, состоящий из следующих флагов: флага нуля (Z - zero), флага переноса (C -carry), флага знака (S - sign), флага четности (P -parity), флага дополнительного переноса (AC - auxiliary carry). Флаги всегда устанавливаются или сбрасываются автоматически после выполнения очередной команды, влияющей на флаги, в зависимости от результата операции. При этом флаг считается установленным, если флаговый разряд принимает значение 1, и сброшенным, если значение разряда 0. Состояния флагов используются в командах условного перехода. Результаты выполнения арифметических и логических операций над содержимым аккумулятора, регистров общего назначения или содержимым ячеек памяти оказывают влияние на флаги следующим образом.

Флаг нуля устанавливается в состояние 1, если после выполнения какой-либо команды получен нулевой результат, и сбрасывается в 0 в случае ненулевого результата.

Флаг переноса устанавливается в 1, если в результате операций сложения и сдвига появляется единица переноса из старшего разряда байта данных, а также если появляется заем из старшего разряда после выполнения операций вычитания или сравнения, в противном случае флаг сбрасывается в 0.

Флаг знака устанавливается в 1, если в результате выполнения операций появляется логическая единица в старшем разряде байта данных (указание на отрицательный результат), и сбрасывается в 0 в случае нулевого значения старшего разряда (указание на положительный результат).

Флаг четности устанавливается в 1, если после выполнения операций сумма единиц в байте данных, подсчитываемых с помощью операции сложения по модулю 2, четна (значение суммы по модулю 2 равно 0), и сбрасывается в 0 в противном случае (число единиц — нечетное).

Флаг дополнительного переноса устанавливается в 1, если в результате выполнения команды появляется сигнал переноса из третьего разряда в четвертый в байте данных результата. Если такого переноса нет, флаг дополнительного переноса сбрасывается в 0. Сигнал этого флага используется во многих схемах вычислений, однако он особенно необходим при сложении чисел в двоично-десятичной форме.

Перейдем теперь к рассмотрению того, как микропроцессор выполняет команды.

4.5. КАК МИКРОПРОЦЕССОР ВЫПОЛНЯЕТ КОМАНДУ?

В ПМ-ЭВМ используется микропроцессор КР580ИК80А, структурная схема которого приведена на рис. 4.4. Микропроцессор содержит 16-разрядную шину адреса и 8-разрядную шину данных, способную к передаче сообщений в двух возможных направлениях. Единовременно передаваемая порция информации соответствует одному байту (8 двоичных разрядов). В самом общем случае возможны следующие

передачи сообщений: 1) пересылка байта данных от устройства ввода; 2) пересылка байта данных к устройству вывода; 3) считывание байта данных из памяти или запись в память; 4) генерирование в шину данных специального байта, называемого *управляющим словом* и предназначенного для установления правильного схемного соединения.

Работа МП (или центрального процессорного элемента ЦПЭ) по реализации каждой команды программы пользователя основана на принципе микропрограммного управления. Это означает, что каждая команда реализуется как некоторая последовательность микрокоманд или микроопераций, приводящая к требуемому результату. Считываемая из памяти микропроцессором команда, вернее, ее 8-разрядный двоичный код (код операции), поступает в регистр команд, где и хранится в течение времени выполнения команды. По результату дешифрирования кода команды происходит формирование последовательности микрокоманд (микропрограммы), процесс выполнения которой и определяет все последующие операции, необходимые для выполнения считанной команды.

Таким образом, выполнение каждой считанной ЦПЭ команды программы пользователя осуществляется в строго определенной последовательности, задаваемой кодом команды. При этом выполнение отдельных микроопераций синхронизируется во времени сигналами $Ф_1$ и $Ф_2$ тактового генератора. Одним из важнейших понятий всего процесса выполнения команд программы является понятие *машинного цикла.*

В микропроцессоре КР580ИК80А процесс выполнения каждой команды можно разбить на ряд основных операций. Время, отведенное на выполнение операции обращения к памяти или к устройству вывода, составляет машинный цикл. Таким образом, процесс выполнения команды состоит из стольких машинных циклов, сколько обращений к памяти или к устройствам ввода/вывода требуется для ее исполнения. Машинный цикл в свою очередь состоит из нескольких машинных тактов.

Каждая команда в зависимости от ее вида может занимать от одного до пяти машинных циклов. Микропроцессор КР580ИК80А имеет 10 типов машинных циклов, перечисленных в табл. 6.2, а каждый машинный цикл может состоять из 3 — 5 машинных тактов. Под *машинным тактом* подразумевается интервал времени, соответствующий одному периоду тактовых импульсов, подаваемых от синхрогенератора.

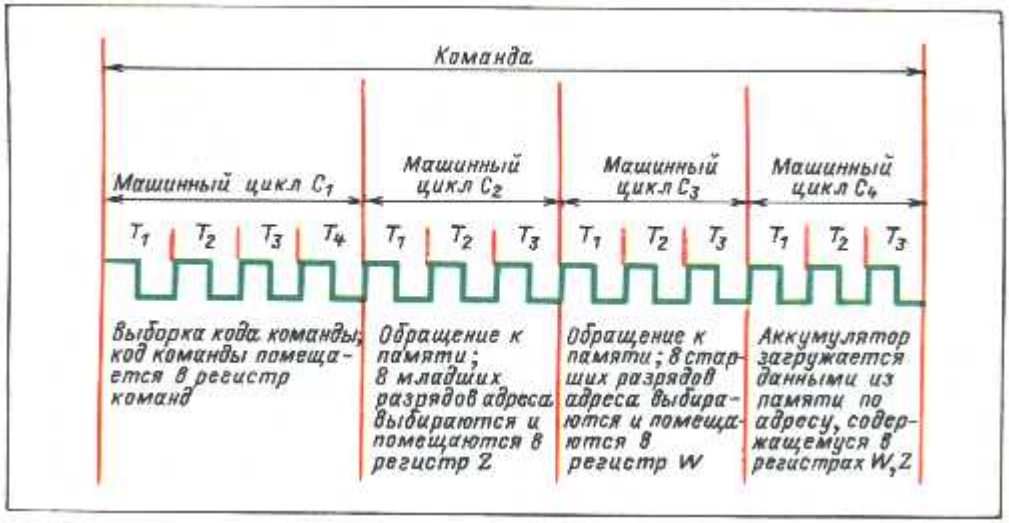


Рис. 4.5. Временная диаграмма для команды, требующей четыре машинных цикла

Для организации машинных циклов требуется формирование строго синхронизированной во времени последовательности управляющих сигналов, обеспечивающих правильные пути прохождения данных в строго определенные моменты времени, и выполнение требуемых микроопераций. Как уже отмечалось, исходными данными для этого являются результаты дешифрирования операционного кода команды. Выполнение команды всегда начинается с цикла обращения к памяти, в результате которого производится считывание кода, интерпретируемого МП как код операции. Будем обозначать машинные циклы одной команды символами Cl, C2 и т. д., а машинные такты одного цикла — символами Tl, T2 и т. д. Таким образом, машинный цикл C1 — это всегда цикл выборки команды. Его длительность обычно — четыре или пять тактов. Последующие циклы C2 — C5 состоят обычно из трех тактов. Для простых операций, таких как арифметические, команды занимают четыре или пять тактов. Команды более сложных операций требуют для выполнения до 18 машинных тактов.

В качестве примера рассмотрим команду LDA (ЗАГРУЗИТЬ данные в аккумулятор), требующую для реализации четыре машинных цикла. На рис. 4.5 приведена временная диаграмма команды LDA. Эта команда переписывает в аккумулятор данные, содержащиеся в определенной ячейке памяти.

Первый машинный цикл C_1, как и следовало ожидать, отведен под выборку команды из памяти. При этом код операции помещается, как уже говорилось, в регистр команд. Выполнение последовательности микрокоманд приводит к следующим действиям. В первом такте T_t содержимое счетчика команд выдается через буфер шины адреса в шину адреса. Второй такт выделен для представления времени для ответа памяти. В третьем такте появившиеся данные в шине данных пересылаются в регистр команд. В четвертом такте выборка команды завершена, операционный код команды передается дешифратору команд для формирования последовательности микроопераций. При этом получает приращение содержимое счетчика команд.

Второй машинный цикл соответствует чтению из памяти восьми младших разрядов адреса, которые в результате выполнения этого цикла должны быть помещены в регистр Z. В первом такте этого цикла содержимое счетчика команд передается через буфер шины адреса в адресную шину. Во втором такте микропроцессор ожидает ответ из памяти. В третьем такте содержимое шины данных передается в регистр Z и счетчик команд опять получает приращение.

Третий машинный цикл аналогичен второму, за исключением того, что из памяти выбираются и пересылаются в регистр W восемь старших разрядов адреса. Последовательность микроопераций та же.

В четвертом машинном цикле происходит окончательное выполнение команды: аккумулятор загружается данными из памяти по адресу, записанному в регистрах W и Z.

В первом такте четвертого машинного цикла содержимое регистров W и Z передается через буфер в шину адреса. Во втором такте предоставляется время для ответа памяти. Наконец, в третьем такте содержимое шины данных пересылается через буфер в аккумулятор. Выполнение команды полностью завершено за 13 машинных тактов.

Перейдем теперь к рассмотрению системы команд микропроцессора КР580ИК80А и особенностей их работы.

4.6. СИСТЕМА КОМАНД И СПОСОБЫ АДРЕСАЦИИ

Система команд микропроцессора КР580ИК80А представлена 244 кодами операций, которые могут быть расклассифицированы по разным признакам. Наиболее существенными для ознакомления с особенностями их использования являются следующие три признака: длина команды, или число занимаемых байтов, функциональный признак, или выполняемые командой операции, и способ адресации. Полный перечень команд приведен в приложении 1. В таблице приведены символическая запись (мнемокод) команды, описание выполняемых ею функций, длина в байтах, обозначения изменяемых флагов и число машинных тактов, занимаемых при ее выполнении. Из 256 возможных кодов (при восьми двоичных разрядах) не используются следующие 12 кодовых комбинаций (в восьмеричной записи): 010, 020, 030, 040, 050, 060, 070, 313, 331, 335, 355 и 375. Этим и объясняется количество всех команд: 244. Команды в таблице разбиты на три группы по числу занимаемых байтов: все команды делятся на однобайтовые, двухбайтовые и трехбайтовые. При этом первый байт всегда отводится под код команды, а второй и третий байты содержат либо данные, либо адрес, по которому они находятся в памяти (рис. 4.6).

Восьмеричные коды всех однобайтовых, двухбайтовых и трехбайтовых команд приведены соответственно в табл. 4.1, 4.2 и 4.3. При этом в табл. 4.1 наряду с обычными указаны обобщенные коды, т. е. коды, содержащие кроме восьмеричных цифр кодовые переменные S, D и A. В табл. 4.1 если код команды содержит только цифры, это означает, что существует единственная команда с соответствующим кодом. Некоторые коды кроме цифр содержат буквы: S (первые И команд), D (третья команда) и A (последняя команда). Таким кодам соответствуют множества команд. При этом каждой команде из первых 11 команд, кроме третьей, соответствует группа конкретных команд, мнемокод которых вместо символа г (регистр) содержит обозначения конкретных регистров (B, C, D, E, H, L), либо символ памяти (M), либо символ аккумулятора (A). Соответствующий конкретный восьмеричный код команды вместо символа S должен содержать одну из цифр: O, 1, 2, 3, 4, 5, 6, 7 соответственно, т. е. символу B в мнемокоде соответствует цифра 0 в восьмеричном коде, символу C в мнемокоде — цифра 1 в восьмеричном коде и т. д,. . ., символу M в мнемокоде — цифра 6 в восьмеричном коде и, наконец, символу A в мнемокоде — цифра 7 в восьмеричном коде. Для третьей команды в табл. 4.1 точно такая же аналогия соблюдается для пар символов $г_ь$ $г_2$ и D, S. Последняя команда таблицы содержит символ A в мнемокоде и в восьмеричном коде. Она соответствует группе из восьми конкретных команд, в каждой из которых символ A принимает одно из значений: O, 1, ...,7.

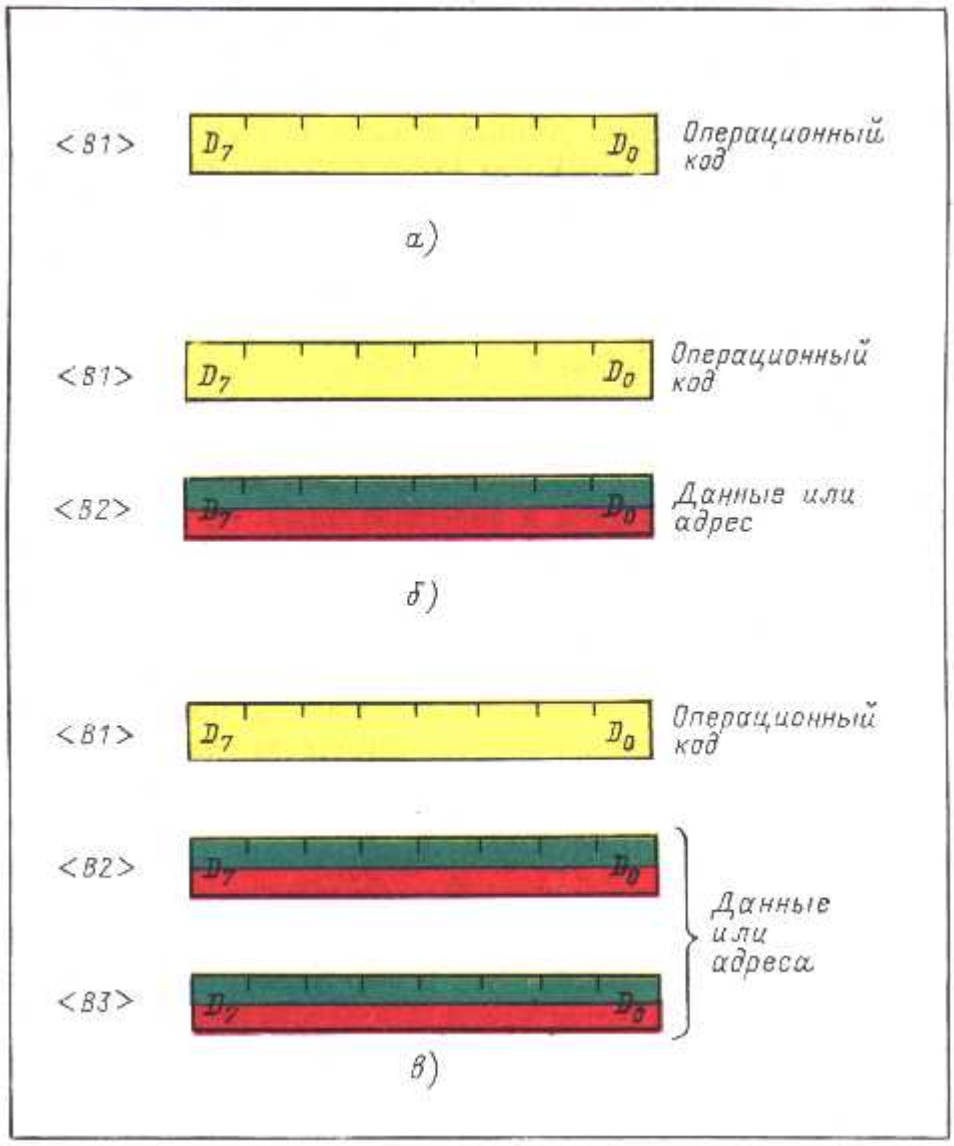


Рис. 4.6. Форматы команд: однобайтовой *(а),* двухбайтовой *(б),* трехбайтовой (в)

Таблица 4.1

Код	Команда		Код	Команда		Код	Команда
OS4	INR	r	051	DAD	H	370	RM
OS5	DCR	r	071	DAD	SP	311	RET
1DS	MOV	r1r2	301	POP	B	007	RLC
20S	ADD	r	321	POP	D	017	RRC
21S	ADC	r	341	POP	H	027	RAL
22S	SUB	r	361	POP	PSW	037	RAR
23S	SBB	r	305	PUSH	B	353	XCHG
24S	ANA	r	325	PUSH	D	343	XTHL
25S	XRA	r	345	PUSH	H	371	SPHL
26S	ORA	r	365	PUSH	PSW	351	PCHL
27S	CMP	r	002	STAX	B	166	HLT
003	INX	в	022	STAX	D	000	NOP
023	INX	D	012	LDAX	B	363	DI
043	INX	H	032	LDAX	D	373	EI
063	INX	SP	300	RNZ		047	DAA
013	OCX	B	310	RZ		057	CMA
033	OCX	D	320	RNC		067	STC
053	OCX	H	330	RC		077	CMC

073	DCX	SP	340	RPO		3A7	RST	A
011	DAD	B	350	RPE				
031	DAD	D	360	RP				

Таблица 4.2

Код	Команда		Код	Команда		
306	ADI	<B2>	323	OUT		<B2>
316	ACI	<B2>	006	MVI	B	<B2>
326	SUI	<B2>	016	MVI	C	<B2>
336	SBI	<B2>	026	MVI	D	<B2>
346	ANI	<B2>	036	MVI	E	<B2)
356	XRI	<B2>	046	MVI	H	<B2>
366	ORI	<B2>	056	MVI	L	<B2>
376	CPI	<B2>	066	MVI	M	<B2>
333	IN	<B2>	076	MVI	A	<B2>

Таблица 4.3

Код	Команда		Код	Команда	
302	JNZ	<B2><B3>	344	CPO	<B2><B3>
312	JZ	<B2><B3>	354	CPE	<B2><B3>
322	JNC	<B2><B3>	364	CP	<B2><B3>
332	JC	<B2><B3>	374	CM	<B2><B3>
342	JPO	<B2><B3>	315	CALL	<B2><B3>
352	JPE	<B2><B3>	001	LXIB	<B2><B3>
362	JP	<B2><B3>	021	LXI D	<B2><B3>
372	JM	<B2><B3>	041	LXIH	<B2><B3>
303	JMP	<B2><B3>	061	LXISP	<B2><B3>
304	CNZ	<B2><B3>	062	STA	<B2><B3>
314	CZ	<B2><B3>	072	LDA	<B2><B3>
324	CNC	<B2><B3>	042	SHLD	<B2><B3>
334	CC	<B2><B3>	052	LHLD	<B2><B3>

Как видно из табл. 4.2, микропроцессор КР580ИК80А может выполнять всего 18 двухбайтовых и 26 трехбайтовых команд, приведенных в табл. 4.3.

Прежде чем переходить к обзору особенностей выполнения различных операций приведенными командами, остановимся на возможных способах адресации. Под способом или режимом адресации подразумевается способ определения данных, участвующих в операциях, или, иначе говоря, способ определения операндов. Для составления программы важно знать особенности процедур, позволяющих преобразовать информацию об адресах команд и данных в физические адреса участков памяти машины.

Для МП КР580ИК80А существуют следующие четыре возможных способа адресации: непосредственная, прямая, регистровая и косвенная.

Непосредственная адресация является наиболее экономичным способом хранения и поиска информации, поскольку необходимые данные содержит сама команда. Эти данные содержатся во втором и третьем байтах трехбайтовой команды или во втором байте двухбайтовой команды. В случае трехбайтовой команды младшие разряды 16-битового числа содержатся во втором байте команды, а старшие - в третьем (рис. 4.7,*а)*.

Менее экономичной, но также довольно простой является *прямая адресация.* В этом случае во втором и третьем байтах команды содержится полный 16-разрядный адрес памяти. Младшим байтом адреса является <B2>, старшим — <B3) (рис. 4.7,6). Таким способом можно адресоваться к любой ячейке адресного пространства памяти.

При *регистровой адресации* код команды содержит указание на регистр или пару регистров, в которых содержатся данные. Используемые в регистровой адресации команды являются однобайтовыми (рис. 4.7,в). Возможность указания пары регистров в однобайтовой команде реализуется за счет того, что адреса регистров являются трехразрядными двоичными кодами.

Косвенная адресация отличается от регистровой лишь тем, что в регистровой паре, определяемой кодом команды, содержатся не данные, а полный 16-разрядный адрес ячейки памяти,в которой имеются эти данные (рис. 4.7,г). Старший байт адреса записывается в первом регистре пары, а младший байт — во втором. Обычно указателем адреса при косвенной адресации являются пара регистров H, т. е. регистры H, L, но иногда используются и пары B и D.

При всевозможных пересылках данных из регистров в регистры или из памяти в регистры или обратно различают регистры — источники данных и регистры — приемники данных. Первые обозначаются символом S (source — источник), вторые — символом D (destination — место назначения). В регистровых парах B, C; D, E и H, L старшими являются первые регистры пар, т. е. регистры B, D, H соответственно. Коды регистров общего назначения, пар регистров и флагов строго фиксированы (табл. 4.4).

В табл. 4.4 обозначение условия в мнемокоде команды в зависимости от значения флагов расшифровывается следующим образом: NZ — результат ненулевой; Z — результат нулевой; NC — нет переноса; C — есть перенос; PO - нет четности; PE — есть четность; P — результат положительный (плюс) ; M -результат отрицательный (минус).

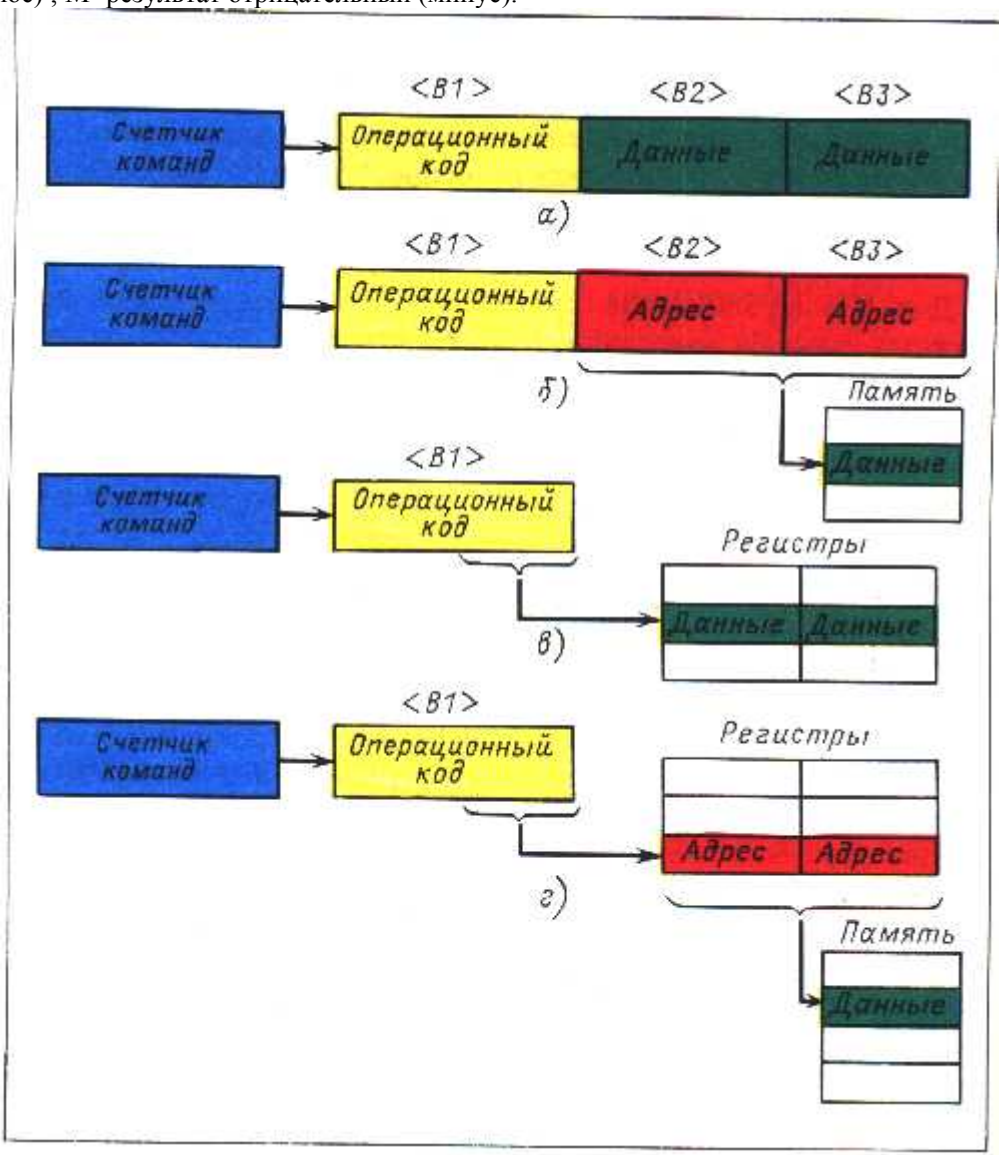


Рис. 4.7. Способы адресации: непосредственная *(а),* прямая *(б),* регистровая (в) и косвенная (г)

Все команды по функциональному признаку могут быть разбиты на следующие пять групп: группа команд пересылки данных, арифметические команды, логические команды, команды переходов и команды управления и работы со стеком. Рассмотрим работу наиболее типичных представителей этих групп команд, многие из которых вошли в описываемую ниже программу-монитор (см. § 7.4).

4.6.1. ГРУППА КОМАНД ПЕРЕСЫЛКИ ДАННЫХ

В этой группе встречаются команды, пересылающие содержащиеся в них данные в регистры и в память, а также команды пересылки данных между регистрами и между регистрами и памятью. Рассмотрим команды пересылки данных из одного регистра МП в другой, из регистра МП в ячейку ОЗУ и из ячейки ОЗУ в регистр МП. В мнемонике команд, используемых в данной книге, порядок пересылки данных всегда предполагается в направлении от крайнего правого операнда к следующему слева операнду, отделенному от первого запятой.

Команда MOV r_t**, r2.** Эта команда предназначена для пересылки данных из регистра S (источник, или r_2) в регистр D (приемник, или r_t). В качестве регистра-источника и регистра-приемника может выступать любой из регистров B, C, D, H, L или аккумулятор A. Содержимое регистра-источника при этом не меняется. В регистре-приемнике просто появляется копия содержимого регистра-источника. На рис. 4.8,а приведен формат этой команды. Чтобы получить конкретную команду, необходимо вместо символов D и S в формате команды рис. 4.8,я проставить из табл. 4.4 коды соответствующих регистров. Например, команда 01 010 000 B пересылает данные из регистра B в регистр D (в регистре D после выполнения команды будут содержаться т© же данные, что и в регистре B). В восьмеричной системе это число (код команды) будет представлено как 120Q (см. список команд в приложении 1).

Команда MOV M, **г.** Эта команда может быть получена из предыдущей заменой символов D кодом M из табл. 4.4. Команда пересылает данные из регистра-источника в ячейку памяти по адресу, указанному в регистровой паре H, L. Регистром-источником может служить любой регистр B, C, D, H, L или аккумулятор A. После выполнения команды в ячейке памяти появится копия содержимого регистра-источника. Содержимое самого регистра не изменится. Код конкретной команды получается фиксацией кода регистра-источника (рис. 4.8,6).

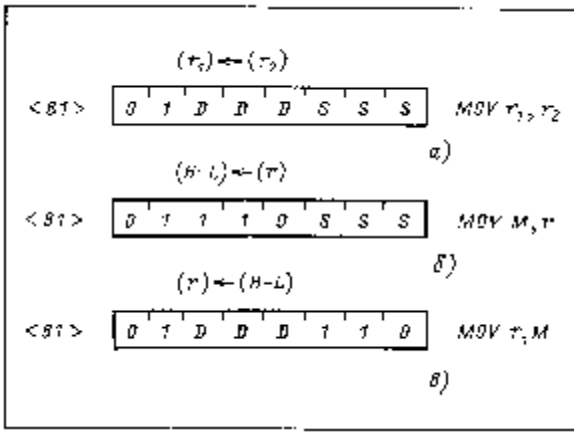


Рис. 4.8. Примеры размещения некоторых команд пересылки данных в байтах

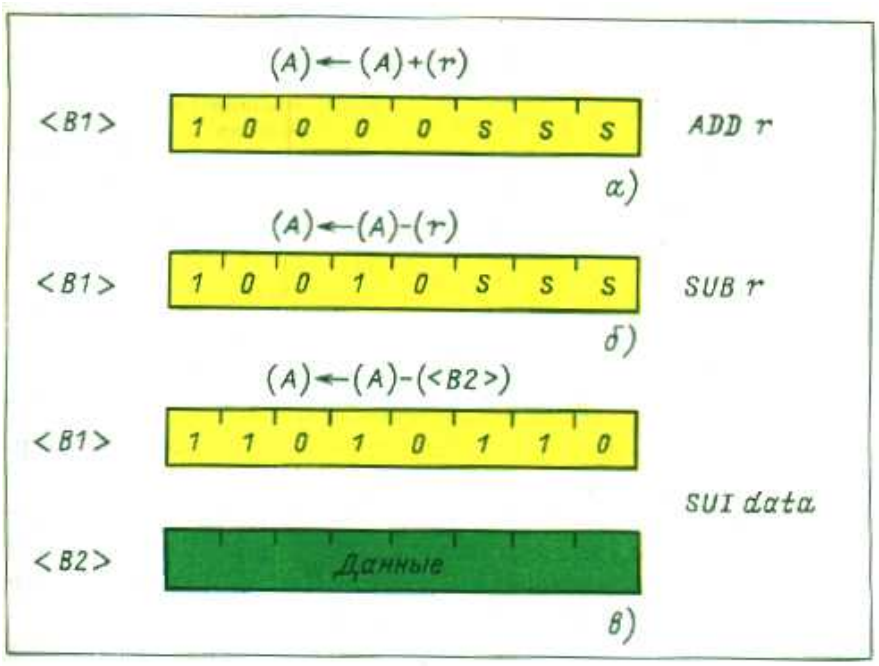


Рис. 4.9. Примеры размещения некоторых арифметических команд в байтах

Команда MOV г, M. Команда также получается из первой заменой символов S кодом M из табл. 4.4. Команда пересылает данные из ячейки памяти по адресу, указанному в регистровой паре H, L, в регистр-приемник, в качестве которого может использоваться любой регистр B, C, D, H, L или аккумулятор A. После выполнения команды в регистре-приемнике появляется копия содержимого ячейки памяти (рис. 4.8,в).

На этом закончим рассмотрение примеров команд этой группы. Отметим лишь, что все команды из группы пересылки данных на значения флагов влияния не оказывают.

4.6.2. ГРУППА АРИФМЕТИЧЕСКИХ КОМАНД

Команды этой группы предназначены для выполнения арифметических операций над данными, хранимыми в регистрах и ячейках памяти. Эти команды в отличие от команд предыдущей группы, как правило, оказывают влияние на значения разрядов флагового регистра, поскольку при выполнении арифметических операций меняются знаки используемых чисел, возникают сигналы переноса, появляются нулевые результаты и т. п. Рассмотрим примеры размещения в байтах некоторых команд этой группы.

Команда ADD r. Эта команда выполняет сложение содержимого регистра-источника S с содержимым аккумулятора А. Результат сложения помещается в аккумулятор. Чтобы получить конкретную команду, необходимо в формате команды вместо символов S проставить код регистра-источника из табл. 4.4. Например, команда 10 000 001 производит сложение содержимого регистра С с содержимым аккумулятора А. Двоичный код этой команды 10 000 001 В; в восьмеричной системе это число представляется кодом 201Q. Это представление является восьмеричным кодом команды ADD С, выполняющей указанное выше сложение (см. список команд в приложении 1). Размещение команды ADD г в байте приведено на рис. 4.9,д.

Команда SUB r. Команда выполняет вычитание содержимого регистра-источника S из содержимого аккумулятора. Результат заносится в аккумулятор (рис. 4.9,6). Например, команда 10 010 ОН В выполняет вычитание содержимого регистра Е из содержимого регистра А и помещает результат в аккумулятор. Код этой команды SUB E — 223 *Q*.

Таблица 4.4

Регистр	Код	Пара регистров	Код	Обозначение условия в мнемокоде	Код
A	111	B (B, C)	00	NZ (Z = 0)	000
B	000	D (D, E)	01	Z(Z=1)	001
C	001	H (H, L)	10	NC (CY - 0)	010
D	010	SP	11	C(CY= 1)	011
E	011			PO (P = 0)	100
H	100			PE(P = 1)	101
L	101			P (S - 0)	ПО
M (память)	110			M (S= 1)	ПО

Команда SUI <B2> производит вычитание содержимого второго байта (команда двухбайтовая) из содержимого аккумулятора и помещает результат в аккумулятор (рис. 4.9,в). Восьмеричный код команды — 326Q (см. приложение 1).

Кроме команд, приведенных в качестве примеров, группа арифметических команд содержит команды сложения и вычитания содержимого аккумулятора с содержимым ячеек памяти, а также команды увеличения и уменьшения на единицу содержимого различных регистров.

4.6.3. ГРУППА ЛОГИЧЕСКИХ КОМАНД

Команды этой группы предназначены для выполнения логических, или булевых, операций над данными, .содержащимися в регистрах, ячейках памяти, а также над флагами условий. К этим операциям относятся операции: логического сложения (ИЛИ), логического умножения (И), суммирования по модулю 2, сравнения, сдвига, дополнения до 1 и до 2. Как и команды предыдущей группы, все логические команды оказывают влияние на флаги.

Команда ANA г выполняет параллельно поразрядное логическое И над содержимым регистра-источника и аккумулятора. Результат операции заносится в аккумулятор (рис. 4.10,а). Например, команда 10 100 100 В выполняет операцию логического умножения поразрядно над содержимым регистра Н и А и заносит результат в аккумулятор. Двоичный код 10 100 100 В соответствует восьмеричному коду 244Q команды ANA H.

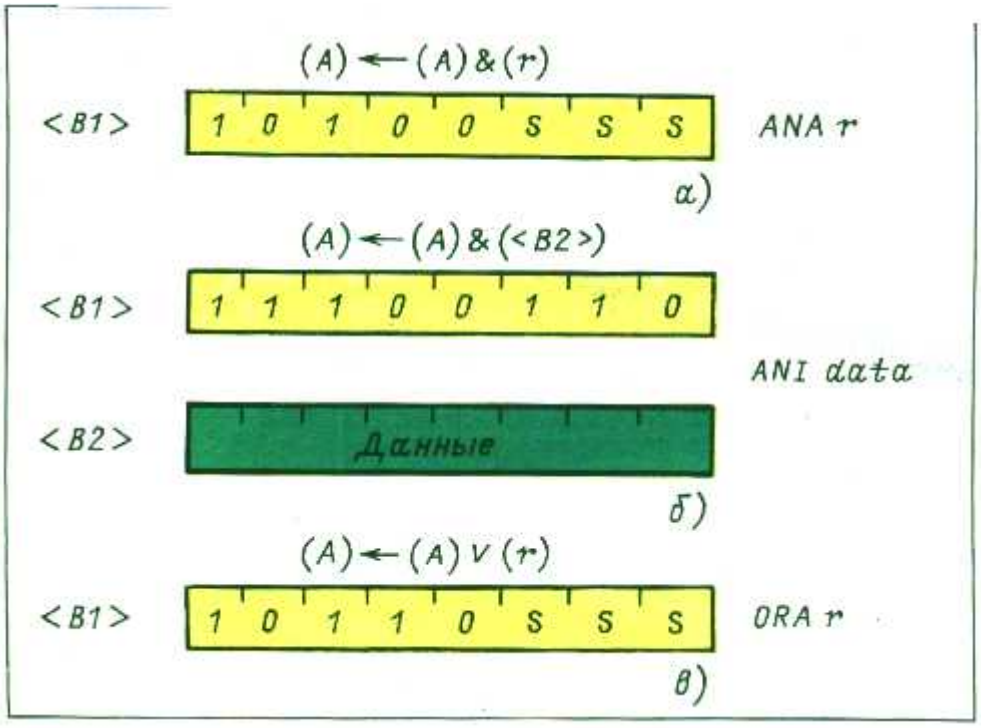


Рис. 4.10. Примеры размещения некоторых логических команд в байтах

Команда ANI < B2 > является двухбайтовой и также выполняет поразрядную операцию логического И, но над содержимым второго байта команды и аккумулятора. Команда имеет восьмеричный код-3460 (рис. 4.10,6). Результат операции заносится в аккумулятор.

Команда ORAr аналогична команде ANA r, но в отличие от нее выполняет операцию поразрядного логического ИЛИ. Результат операции заносится в аккумулятор (рис. 4.10,в).

4.6.4. ГРУППА КОМАНД ПЕРЕХОДОВ

Эта группа команд предназначена для организации правильной последовательности выполнения программы. Сюда входят команды безусловного и условного переходов, команды вызова подпрограммы и возвращения к главной программе. Все команды этой группы на флаги влияния не оказывают. Команды безусловного перехода выполняют специальные операции над содержимым счетчика команд. Команды условного перехода обеспечивают необходимое ветвление программы путем анализа состояния одного из четырех флагов: нуля, знака, четности и переноса, коды которых указаны в табл. 4.4.

Команда JMP <B2> <B3> — трехбайтовая команда передает управление команде по адресу, содержащемуся в третьем и втором байтах текущей команды. Это осуществляется путем записи содержимого третьего и второго байтов команды в счетчик команд (рис. 4.11,а). Как уже отмечалось в § 4.3, счетчик команд представляет собой 16-разрядный регистр, содержащий адрес, по которому можно обратиться для считывания очередного байта команды. Восьмеричный код этой команды - 303 Q.

Команды **CALL и RET** — команды безусловного перехода. Первая из них передает управление подпрограмме, прекращая выполнение основной программы; вторая передает управление главной программе, возвращаясь к ее выполнению (рис. 4.11,£, в). Первая команда — трехбайтовая. Восемь старших разрядов адреса следующей команды пересылаются в ячейку памяти, адрес которой на единицу меньше содержимого указателя стека. Восемь младших разрядов адреса следующей команды пересылаются в ячейку памяти, адрес которой на две единицы меньше содержимого указателя стека. Содержимое указателя стека уменьшается на две единицы. Управление передается команде, адрес которой размещается в третьем и втором байтах команды CALL

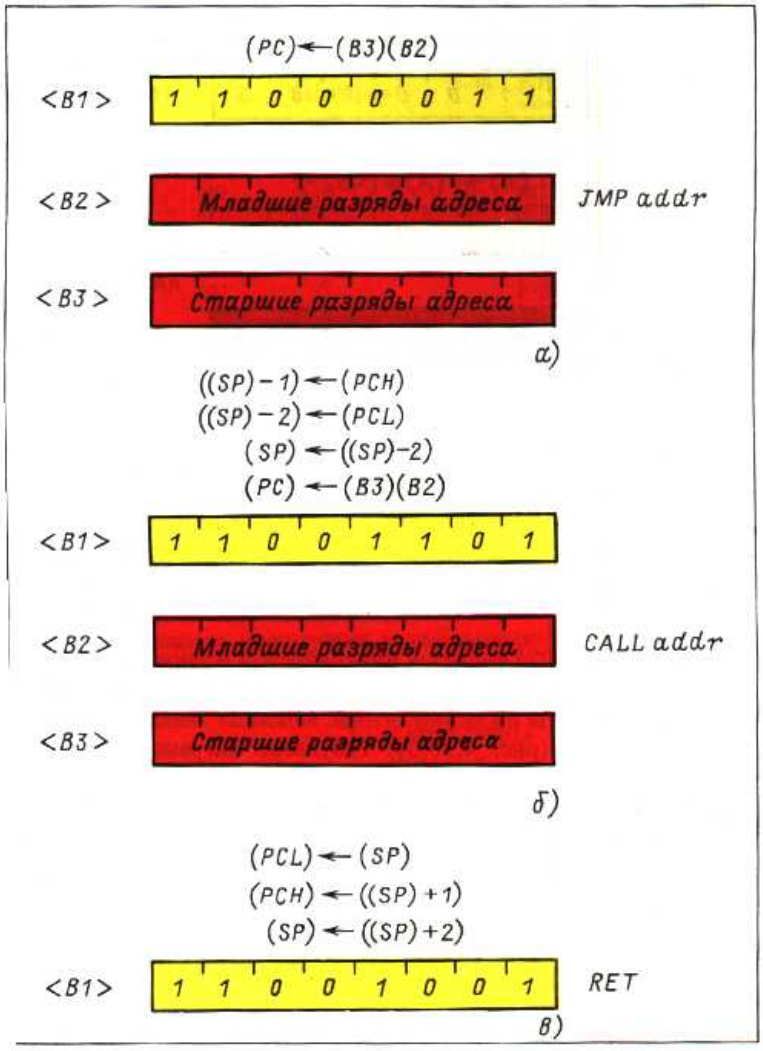


Рис. 4.11. Примеры размещения некоторых команд переходов в байтах

Команда **RET** — однобайтовая. Ее восьмеричный код — 311Q (код предыдущей команды CALL — 315Q). В процессе выполнения этой команды содержимое указателя стека получает приращение на две единицы. Содержимое ячейки памяти по адресу, хранящемуся в указателе стека, пересылается в счетчик команд на место младших восьми разрядов. Содержимое ячейки памяти по адресу, на единицу большему, чем содержимое указателя стека, пересылается в счетчик команд на место старших восьми разрядов. Таким образом, возвращение к главной программе происходит всегда путем обращения к байту команды, непосредственно следующему за байтом, используемым командой CALL

4.6.5. ГРУППА КОМАНД УПРАВЛЕНИЯ И РАБОТЫ СО СТЕКОМ

Команды этой группы предназначены для управления работой микропроцессора, устройствами ввода/вывода и стеком. Команды этой группы не оказывают влияния на флаги. Рассмотрим в качестве примера работу некоторых команд из этой группы.

Команда XTHL — однобайтовая команда с восьмеричным номером 343Q. Является примером наиболее длинных по времени исполнения команд - занимает 18 машинных тактов.

В процессе выполнения команды содержимое регистра L меняется на содержимое ячейки памяти по адресу, содержащемуся в указателе стека, и наоборот. Содержимое регистра H меняется на содержимое ячейки памяти по адресу, на единицу большему, чем содержимое указателя стека, и наоборот (рис. 4.12,я). На это уходит пять машинных циклов.

Команды IN <B2>, OUT <B2> . Эти команды предназначены для ввода данных от входного порта в аккумулятор и вывода данных из аккумулятора в выходной порт соответственно. Первая команда имеет восьмеричный код 333 Q, вторая — 323 Q. Обе команды — двухбайтовые. Второй байт обеих команд отведен под адрес соответствующего входного и выходного портов. В результате выполнения первой команды данные от входного порта по двунаправленной шине данных передаются в аккумулятор. В

результате выполнения второй команды данные выводятся по той же шине данных из аккумулятора в выходной порт (рис. 4.*12,6,* в).

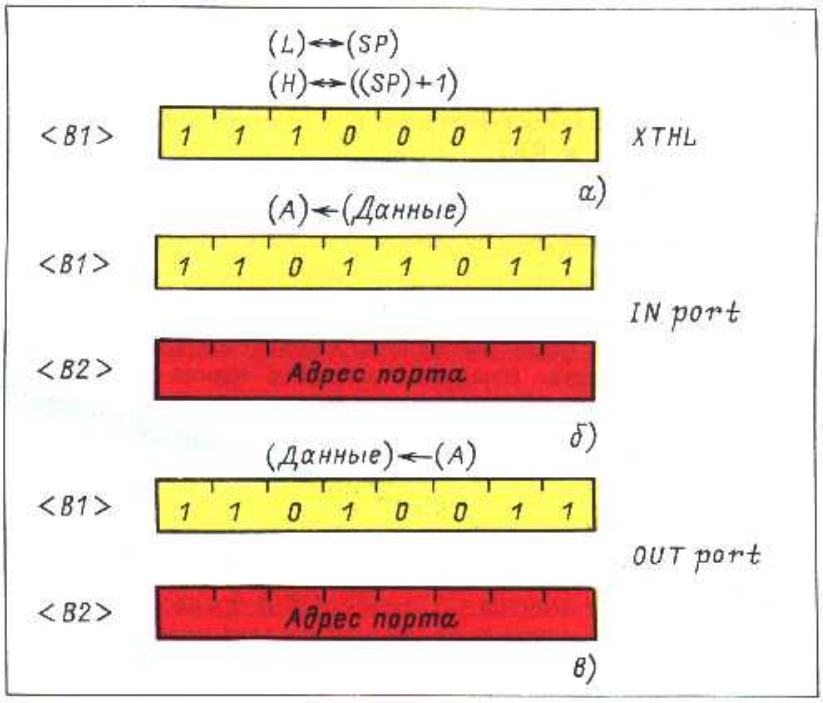


Рис. 4.12. Примеры размещения некоторых команд управления в байтах

На этом закончим рассмотрение примеров размещения в байтах команд описанных выше групп и перейдем к вопросу о том, как составляется программа решения на ПМ-ЭВМ задачи с использованием приведенного в приложении 1 списка команд.

4.7. ПРОГРАММИРОВАНИЕ ПМ-ЭВМ

Как уже отмечалось в гл. 3, ПМ-ЭВМ не имеет программного обеспечения, позволяющего пользоваться для решения задач языками высокого уровня. Поэтому запись программы для ПМ-ЭВМ осуществляется пользователем на машинном языке с применением команд описанных выше групп.

Полный перечень команд, приведенных в сжатой форме в табл. 4.1 — 4.3, должен быть хорошо известен пользователю, решающему свою задачу. Это даст ему возможность варьировать при составлении своих программ различными командами и способами их включения в программы с целью составления программ либо более коротких, либо экономящих используемую память, либо сокращающих время счета.

В ряде случаев способ решения той или иной задачи становится более понятным, если он представлен в виде специальной схемы, называемой схемой алгоритма решения и имеющей вид ориентированной сети с вершинами различных типов, соответствующими используемым операциям.

В качестве примера рассмотрим решение задачи суммирования первых 20 чисел натурального ряда, схема которой приведена на рис. 4.1*3,а.*

Для реализации первых двух операторов (не считая Start) можно воспользоваться двухбайтовой командой MVI r, осуществляющей непосредственную загрузку в какой-либо регистр данных, содержащихся во втором байте команды (в данном случае - чисел OD). Для осуществления операции $S = S + N$ можно воспользоваться командой ADD r, выполняющей суммирование содержимого какого-либо регистра с содержимым аккумулятора и запись результата в аккумулятор. Для выполнения операции $N = N + 1$ в машинном языке ПМ-ЭВМ предусмотрена специальная команда INR r, осуществляющая увеличение на единицу содержимого какого-либо регистра. Остается еще организовать в программе цикл с использованием условного оператора. Для этой цели подошла бы команда JNZ, осуществляющая переход в программе при отсутствии нуля в результате какой-либо предшествующей переходу операции (в нашем случае — операции вычитания: 21 — *N).*

Однако этого можно и не делать, построив несколько иначе схему решения задачи. Вместо того чтобы суммировать последовательно увеличивающиеся числа от O D до 20 D, можно суммировать последовательно уменьшающиеся числа от 20 D до OD. Тогда содержимое регистра, хранящего число *N,* на некотором шаге выполнения программы обратится в нуль и этот результат можно будет использовать при организации условного перехода.

Итак, новая схема алгоритма может быть построена следующим образом.

Шаг 1. Заносим число O D в аккумулятор (A).

Шаг 2. Заносим число 20 D в какой-нибудь регистр, например в регистр D.

Шаг 3. Суммируем содержимое регистров A и D.

Шаг 4. Уменьшаем содержимое регистра Она единицу.

Шаг 5. Если содержимое регистра Вне равно нулю, осуществляем переход к шагу 3; в противном случае переходим к следующему шагу.

Ш а г 6. Переписываем содержимое регистра A в какой-нибудь из портов вывода, например в порт P_0оо-

Ш а г 7. Конец работы программы.

Соответствующая схема представлена йа рис. *4.13,6.*

Приступим к программированию этой схемы. Для этой цели нам понадобятся все указанные выше команды, за исключением команды INR г (увеличение содержимого регистра на единицу). Вместо нее следует взять команду DCR г (уменьшение содержимого регистра на единицу), которая будет использована применительно к регистру D. Кроме того, нам будут необходимы еще следующие две команды: OUT — двухбайтовая команда вывода данных из аккумулятора в порт вывода, определяемый адресом, содержащимся во втором байте команды, и однобайтовая команда HLT — останов программы. Коды всех используемых команд можно взять из таблицы, приведенной в приложении 1.

Для размещения всей программы нам понадобятся 12 ячеек памяти с номерами от 014QOOOQ до 014Q013Q. В первые две ячейки с номерами 014 000 и 014 001 поместим оба байта первой команды MVI A OOOQ загрузки аккумулятора (регистра A) числом OD. При этом в первую ячейку поместим восьмеричный код команды 076, а во вторую — содержимое второго байта этой команды — восьмеричный код числа OD (число 000). В следующие две ячейки с номерами 014 002 и 014 003 поместим два байта следующей команды, загружающей регистр D десятичным числом 20. Восьмеричный код этой операции 026 будет находиться в первой ячейке, а восьмеричный код 024 десятичного числа 20 - во второй и т. д. В последнюю ячейку с номером 014 013 поместим восьмеричный код 166 команды HLT, осуществляющей останов программы. В табл. 4.5 приведен полный текст составленной программы вместе с комментарием. При организации цикла использована специальная метка Ml, указывающая на операцию, которую следует выполнить, если содержимое регистра D не равно нулю.

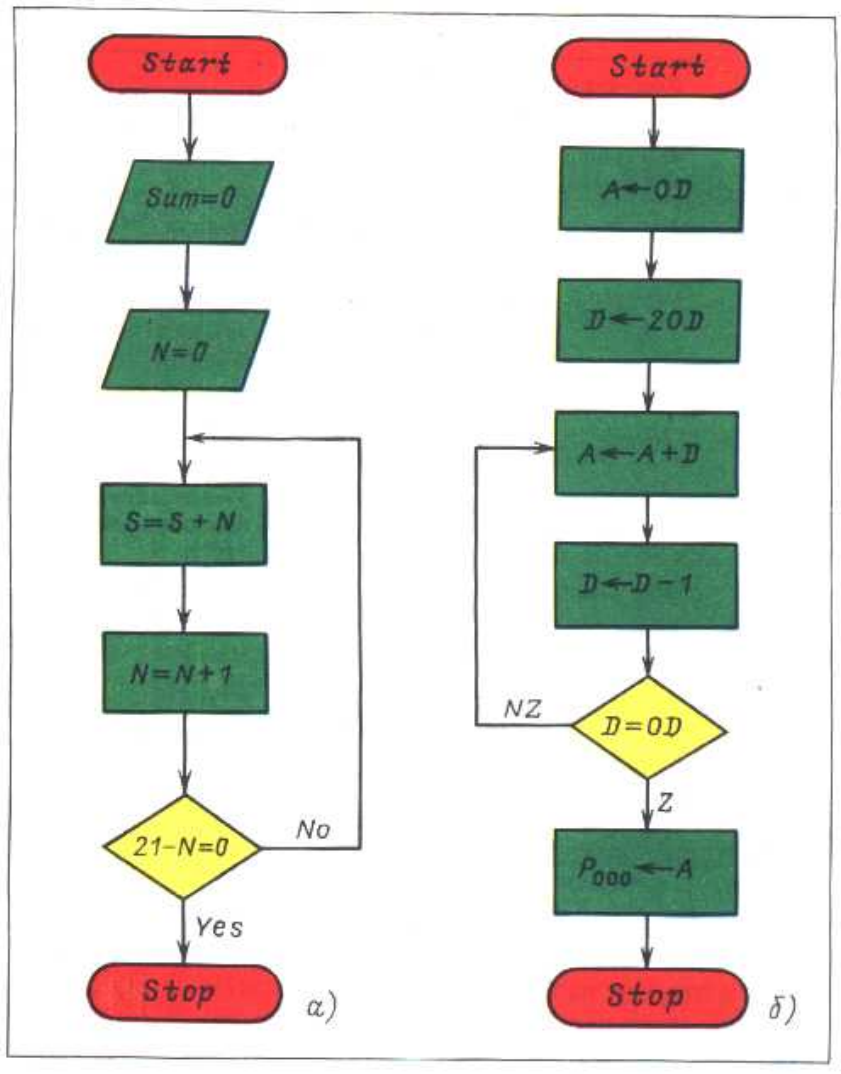


Рис. 4.13. Схемы алгоритмов для задачи сложения первых 20 чисел

Таблица 4.5

Адрес ячейки	Код операции или содержимого второго или третьего байта	Метка	Мнемокод	Комментарий
014 000	076		MVI A OOOQ	Загрузка регистра A числом OD
014 001	000			Число OD
014 002	026		MVI D 024Q	Загрузка регистра D числом 20 D
014003	024			Число 20 D
014 004	202	MI:	ADDD	Сложение содержимого регистров A и D
014005	025		DCRD	Уменьшение на единицу содержимого регистра D
014 006	302		JNZM1	Переход к метке M 1 при отсутствии нуля
014 007	004			Метка M 1 (адрес)
014 010	014			Метка M 1 (адрес)
014 011	323		OUT OOOQ	Вывод данных из регистра A в порт POOO
014 012	000			Номер порта
014013	166		HLT	Останов программы

Аналогичным образом осуществляется составление и других программ, которые читатель встретит в этой книге. Отметим еще раз, что при написании программы совсем не обязательно прибегать вначале к составлению ее операторной схемы. При некотором навыке программа легко может быть составлена сразу после небольшого анализа условий сформулированной задачи.

Перейдем теперь к рассмотрению работы конкретных схем отдельных функциональных блоков ПМ-ЭВМ, назначение которых было определено ранее.

5

ИСПОЛЬЗУЕМЫЕ МИКРОСХЕМЫ

5.1. ОБЩИЕ ВОПРОСЫ

В последующих главах будет подробно описано устройство и функционирование ПМ-ЭВМ и ее блоков. Для понимания этих глав необходимо познакомиться с работой отдельных микросхем, из которых построены блоки ПМ-ЭВМ.

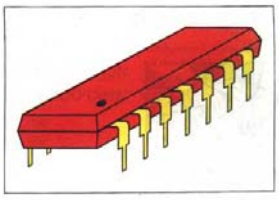


Рис. 5.1. Внешний вид микросхемы в корпусе с двухрядным расположением выводов

Интегральная микросхема - микроэлектронное изделие, выполняющее определенную функцию преобразования и обработки сигнала и имеющее высокую плотность упаковки электрически соединенных элементов. Полупроводниковый кристалл, в объеме и на поверхности которого созданы элементы, составляющие микросхему, обычно помещается в корпус. Корпуса микросхем бывают самых разнообразных форм и изготовляются из керамики, пластмассы, металла и т. д. В ПМ-ЭВМ будут использоваться микросхемы в прямоугольных пластмассовых корпусах (рис. 5.1) с двухрядным расположением выводов. Принято обозначать выводы микросхем арабскими цифрами и буквами русского и латинского алфавитов. При этом для корпусов всех типов используется общее соглашение по обозначению номеров выводов, практически не имеющее исключений. У вывода номер один ставится маркирующая точка, которая называется *ключом.* Начиная от этой точки номера выводов будут возрастать при обходе их против часовой стрелки. На условных графических изображениях микросхем эти номера выводов проставляются над линиями, изображающими выводы. Принято также присваивать выводам буквенные обозначения в соответствии с названиями выполняемых выводамифункций. Существуют две системы буквенных обозначений: отечественная (буквами русского алфавита) и международная (буквами латинского алфавита). В этой книге будет применяться международная система, так как она широко используется во многих отечественных и переводных книгах по электронике и микро-ЭВМ. Выводы будут обозначаться в тексте буквами или номерами (если не принято данный вывод обозначать буквой) при описании микросхем и номерами — при описании электронных схем, чтобы избежать двусмысленности этих описаний.

5.2. ТТЛ-ВХОДЫ И ТТЛ-ВЫХОДЫ

Цифровые микросхемы выполняют преобразование сигналов, изменяющихся дискретно. Такой сигнал может принимать несколько фиксированных значений. Современные микросхемы, как правило, работают с сигналами, которые имеют два значения, и эти значения кодируются двумя различными уровнями напряжения. Обычно эти уровни напряжения расположены в диапазоне от нуля до напряжения источника питания и различны для серий микросхем, отличающихся по технологии изготовления. Широкое распространение в настоящее время получили микросхемы, изготовленные по ТТЛ-технологии, и поэтому часто микросхемы, изготовленные по другим технологиям, имеют выводы с такими же характеристиками и работают с такими же уровнями напряжения (в частности, все микросхемы, применяющиеся в ПМ-ЭВМ). Для того чтобы подавать сигналы с микросхем, имеющих одни уровни напряжения, на микросхемы с другими уровнями напряжения, применяются специальные преобразователи уровня.

Вывод микросхемы, на который сигнал необходимо подавать, называется входом (для ТТЛ-микросхем - ТТЛ-входом), а вывод, на котором сигнал вырабатывается самой микросхемой, называется выходом (для ТТЛ-микросхем - ТТЛ-выходом). В этом параграфе будут подробно рассмотрены характеристики ТТЛ-входов и ТТЛ-выходов.

Входные и выходные уровни напряжения ТТЛ-микросхем имеют определенные значения. Высокий уровень напряжения должен лежать в пределах от +2,4 до +5 В для ТТЛ-выхода и в пределах от +2 до +5 В для ТТЛ-входа, а низкий уровень -от 0 до +0,4 В для ТТЛ-выхода и от 0 до +0,8 В для ТТЛ-входа. Между нижней границей для более высокого уровня и верхней границей для более низкого уровня напряжения ТТЛ-входа имеется диапазон напряжения (от 0,8 до 2 В) шириной 1,2 В. Этот диапазон предназначен для защиты микросхемы от ложных срабатываний при помехах амплитудой менее 1,2 В. Если бы его не было, то любая малая помеха приводила бы к тому, что напряжение сигнала попадало бы из области низких напряжений в область высоких (или наоборот). В этом диапазоне напряжение сигнала может находиться только в момент переключения с одного уровня напряжений на другой.

Как известно, переменные в логике могут принимать два значения - 0 и 1. Возможны два способа кодировки этих значений. Первый способ - это кодировать 0 низким уровнем напряжения, а 1 - высоким уровнем напряжения. Такой способ кодировки называется *позитивной (иногда положительной) логикой.* Второй способ - кодировать 0 высоким уровнем, а 1 - низким уровнем напряжения. Такой способ кодировки называется *негативной (иногда отрицательной) логикой.* Эти названия общеприняты, хотя не совсем удачны. Во-первых логика здесь ни при чем (она одинаковая, разные только способы кодировки), а во-вторых, слова "позитивный" и "негативный" применяются для обозначения способов кодировки с помощью разных уровней напряжения одной полярности. В дальнейшем для краткости вместо слов "сигнал имеющий высокий (низкий) уровень напряжения" будут употребляться слова "сигнал высокого (низкого) уровня" или просто высокий (низкий) уровень", и так как в этой книге принята положительная логика, то будут также употребляться слова "уровень логического нуля (единицы)".

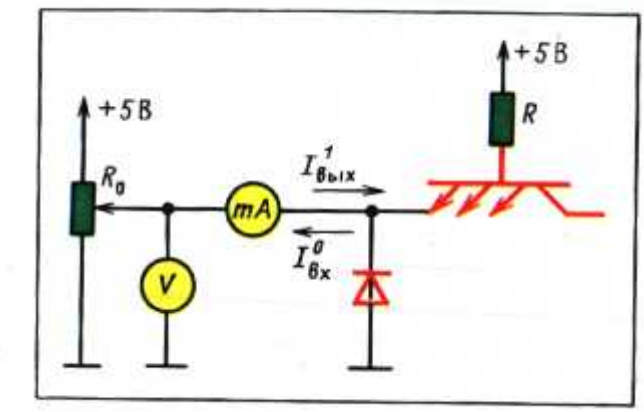


Рис. 5.2. ТТЛ-вход и схема для снятия характеристики

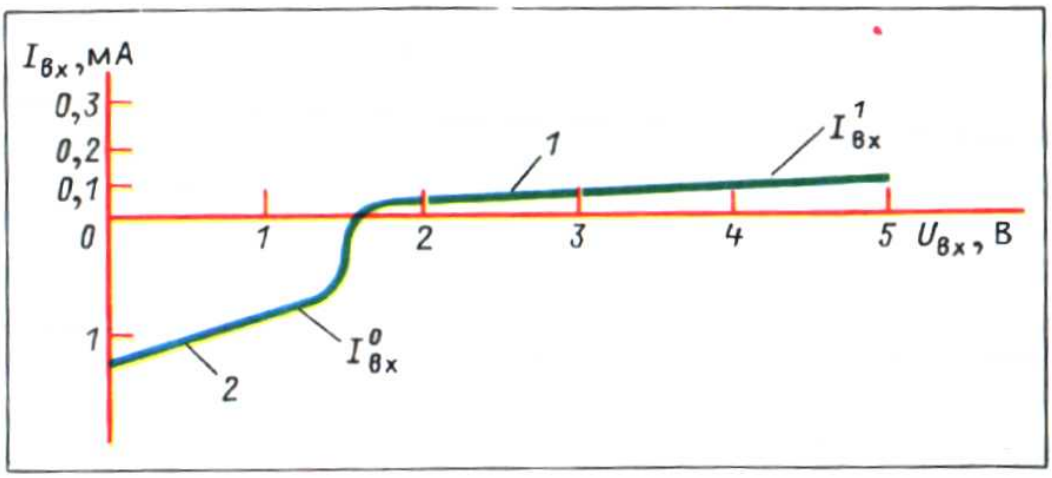


Рис. 5.3. Характеристика ТТЛ-входа

ТТЛ-вход является эмиттером многоэмиттерного транзистора (рис. 5.2). Диод к эмиттеру подключен для того, чтобы ограничивать отрицательные импульсы напряжения. Если поставить эксперимент, изображенный на рис. 5.2, то можно получить входную характеристику ТТЛ-входа, т. е. зависимость тока от подаваемого напряжения (рис. 5.3). Ветви *1* соответствует ток, вытекающий в ТТЛ-вход, который в этом случае работает как *p-n* переход, включенный в обратном направлении, и поэтому значение тока невелико. Этот ток обозначается обычно $I^1_{вх}$. Ветви *2* соответствует ток, который вытекает из ТТЛ-входа, работающего как *p-n* переход, включенный в прямом направлении. В этом случае ток ограничивается сопротивлением *R*. Этот ток обозначается $I^0_{вх}$. В диапазоне напряжений, соответствующем высокому уровню, $I^1_{вх}$ должен быть не больше 0,1 мА, а в диапазоне, соответствующем низкому уровню, $I^0_{вх}$ должен быть не больше 1,6 мА. Направления $I^0_{вх}$ и $I^1_{вх}$ показаны стрелками на рис. 5.2. Вход с такими характеристиками называется *стандартным ТТЛ-входом* или *стандартной ТТЛ-нагрузкой.*

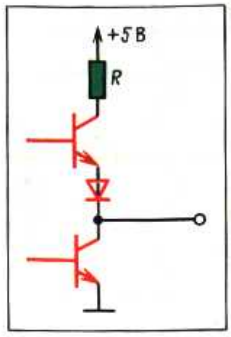


Рис. 5.4. ТТЛ-выход

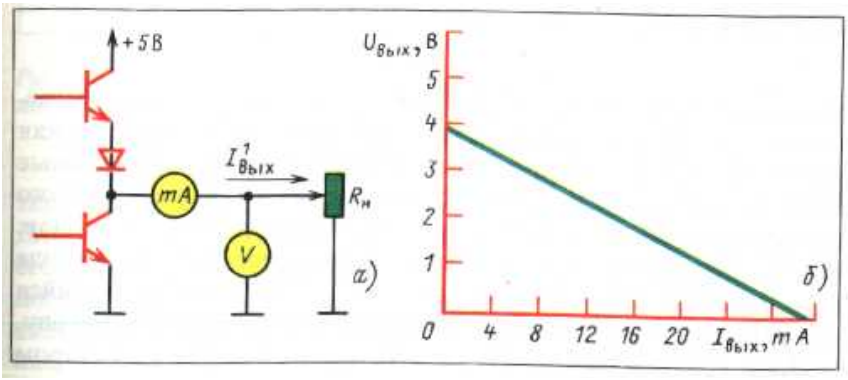


Рис. 5.5. Схема снятия характеристики ТТЛ-выхода при высоком уровне напряжения *(а)*; характеристика *(б)*

ТТЛ-выходы бывают трех типов: нормальный ТТЛ-выход, выход с открытым коллектором и выход с тремя состояниями. Схема *нормального ТТЛ-выхода* приведена на рис. 5.4. Для того чтобы создать на выходе высокий уровень напряжения, с помощью внутренних управляющих цепей открывается верхний транзистор и закрывается нижний. Если поставить эксперимент по схеме, изображенной на рис. 5.5,л, то можно снять выходную характеристику ТТЛ-выхода при высоком уровне напряжения (рис. *5.5,б)*. Из этой характеристики видно, что с возрастанием вытекающего тока $I^1_{вых}$ напряжение на выходе уменьшается. Стандартный ТТЛ-выход должен обеспечивать при $I^1_{вых}$, равном 1 мА, высокий уровень напряжения (не меньше+2,4 В).

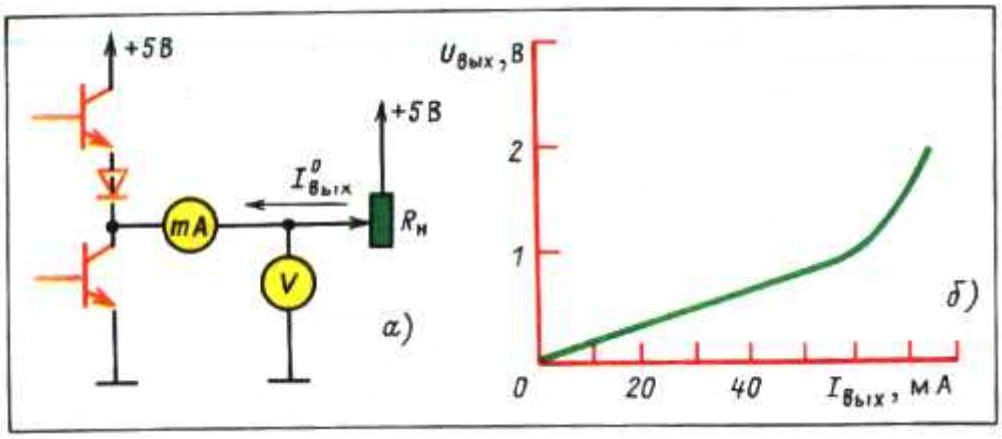


Рис. 5.6. Схема снятия характеристики ТТЛ-выхода при низком уровне напряжения *(а);* характеристика *(б)*

Для того чтобы создать на выходе напряжения низкого уровня, с помощью внутренних управляющих цепей открывается нижний транзистор и закрывается верхний. Схема, с помощью которой можно снять выходную характеристику ТТЛ-выхода, и сама характеристика приведены на рис. 5.6. Из этой характеристики видно, что с возрастанием втекающего тока напряжение на ТТЛ-выходе, находящемся в состоянии низкого напряжения, увеличивается. Стандартный ТТЛ-выход должен обеспечивать при $I^0_{вых}$, Равном 16 мА, высокий уровень напряжения (не более 0,4 В). Если теперь сравнить входные и выходные токи, то видно, что к одному стандартному ТТЛ-выходу можно подключить, не перегружая его, 10 стандартных ТТЛ-входов. Иногда используют термин "нагрузочная способность" выхода или "коэффициент разветвления по выходу", измеряющийся числом входов, которые можно подключить к данному выходу.

Диод между эмиттером верхнего транзистора и коллектором нижнего транзистора включен для того, чтобы избежать открывания верхнего транзистора при низком уровне напряжения на выходе. При переключении выходных транзисторов в какой-то момент времени они оба оказываются открытыми и ток через них резко возрастает (в этот момент ток ограничивается только резистором *К)*. Резкое возрастание тока может создать импульс помехи, распространяющейся по цепям питания. Для подавления таких помех используются развязывающие конденсаторы, которые включаются между линией питания и общим выводом в непосредственной близости от микросхемы. Развязывающий конденсатор должен быть один на группу не более 10 микросхем и должен иметь емкость не менее 0,002 мкФ на микросхему.

Два нормальных ТТЛ-выхода нельзя подключать друг к другу, если возможна ситуация, когда на одном выходе высокий, а на другом — низкий уровень напряжения, потому что тогда ток, протекающий через открытые транзисторы разных ТТЛ-выходов, может превысить допустимое значение.

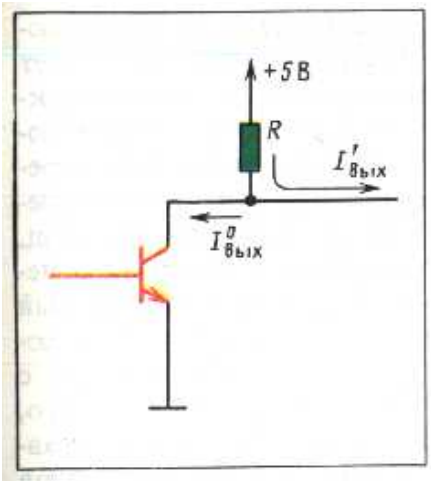


Рис. 5.7. Выход с открытым коллектором

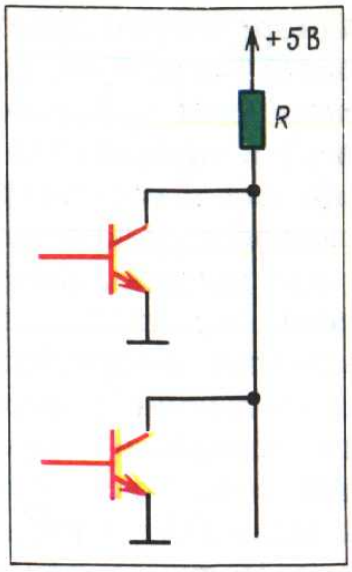


Рис. 5.8. Соединение нескольких выходов с открытым коллектором

Второй тип ТТЛ-выхода — это так называемый *выход с открытым коллектором.* Как следует из названия, выход такого типа представляет собой коллектор транзистора (рис. 5.7). Для того чтобы обеспечить нормальную работу этого выхода, его необходимо соединить через резистор с положительным полюсом источника питания. Тогда если транзистор закрывается с помощью внутренних управляющих цепей, то напряжение на выходе больше +2,4 В, что достаточно для создания стандартного высокого ТТЛ-уровня; если же транзистор открывается, то напряжение на выходе падает до стандартного низкого ТТЛ-уровня. При закрытом выходном транзисторе ток $I^1_{вых}$ течет от положительного полюса источника питания через резистор *R,* при открытом транзисторе ток $I^0_{вых}$ течет через транзистор. Резистор *R* рассчитывается так, чтобы обеспечить необходимые значения токов $I^0_{вых}$ и $I^1_{вых}$.

В отличие от стандартных ТТЛ-выходов выходы с открытым коллектором можно соединять друг с другом (рис. 5.8). В этом случае высокий уровень напряжения поддерживается в точке соединения только тогда, когда закрыты транзисторы всех выходов, поэтому в этой точке реализуются логическая функция И при позитивной логике и логическая функция ИЛИ при негативной логике. Такая схема часто называется "монтажное И" или "монтажное ИЛИ" в зависимости от используемой логики.

Третий тип ТТЛ-выходов - это *выходы с тремя состояниями* или, как их еще называют, выходы с третьим состоянием высокого сопротивления (или просто выходы с третьим состоянием). В отличие от нормальных ТТЛ-выходов оба выходных транзистора такого выхода с помощью внутренних управляющих цепей могут быть закрыты. В этом случае через них может протекать лишь небольшой ток утечки (обычно несколько микроампер) и говорят, что выходы находятся в отключенном состоянии или в состоянии высокого сопротивления. Для перевода выходов микросхемы в состояние высокого сопротивления, как правило, имеется специальный управляющий вход, или если этот переход происходит вследствие каких-либо неуправляемых внутренних процессов, то имеется специальный выход, состояние которого показывает, переведены ли в состояние высокого сопротивления другие выходы. Выходы с третьим состоянием сконструированы специально для того, чтобы можно было подключать несколько выходов для управления состоянием одной линии. Когда несколько выходов с третьим состоянием подключены к одной линии, то в определенный момент времени только один из них может управлять состоянием этой линии (т. е. создавать на ней уровень высокого или низкого напряжения). Остальные выходы должны находиться в состоянии высокого сопротивления. В этом состоянии они не создают дополнительную нагрузку на транзисторы работающего выхода. Естественно, что это требует специальных электронных схем, которые управляют тем выходом, который должен работать в какой-либо определенный момент времени.

Вывод микросхемы может быть входом или выходом и, кроме того, может совмещать эти функции. Это достигается путем объединения внутри микросхемы входов и выходов с тремя состояниями или открытым коллектором. Для управления таким выводом у микросхемы имеется специальный вход, в зависимости от сигнала на котором вывод работает как ТТЛ-вход или как ТТЛ-выход в разные моменты времени. Ниже будут встречаться микросхемы со всеми разобранными типами выводов. Нормальные ТТЛ-входы и ТТЛ-выходы в дальнейшем будем называть просто входами и выходами, а во всех остальных случаях будем указывать определенный тип вывода.

5.3. ВРЕМЕННЫЕ ДИАГРАММЫ

Электрические сигналы, которые соответствуют определенным логическим состояниям, можно наблюдать на выводах работающих микросхем с помощью осциллографа или логического пробника. На рис. 5.*9,а* показано изображение на экране осциллографа электрического сигнала с вывода цифровой микросхемы. Ось времени располагается горизонтально, а ось напряжений — вертикально. Время увеличивается слева направо, т. е. из двух событий правее окажется более позднее. Такое расположение и направление оси времени традиционны и используются во всех осциллографах.

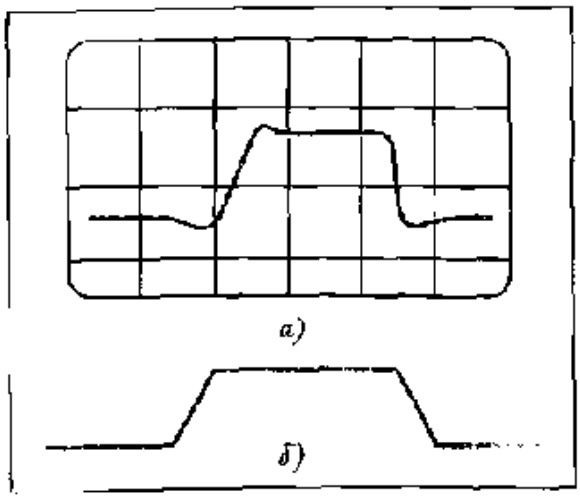


Рис. 5.9. Дискретный сигнал:
а - на экране осциллографа; *б* - условное графическое изоб -ражение

Электрические сигналы с выводов цифровых микросхем принято изображать в процессе их изменений во времени (рис. *5.9,6).* Такие условные графические изображения называются временными диаграммами. На временных диаграммах не наносятся оси напряжения и времени. Нулевая отметка времени также не наносится, так как большинство процессов периодически повторяется и достаточно изобразить интервал времени немного большим периода повторения, чтобы диаграмма содержала всю необходимую информацию о происходящем процессе.

Приведем некоторые соглашения, которые используются при изображении сигналов на временньгх диаграммах. Высокий уровень изображается, как показано на рис. 5.10,я, состояние высокого сопротивления - как на рис. 5.10,в, низкий уровень -как на рис. 5.*10,6.* Штриховые линии на этих рисунках приведены только для сравнения и обычно не наносятся. На рис. 5.11 показано, как изображается переход с одного уровня на другой. Переход от низкого уровня к высокому называется фронтом, а от высокого к низкому — срезом. Иногда употребляют также термины "нарастающий и спадающий фронт", "передний и задний фронт", "положительный и отрицательный фронт" и некоторые другие. Если переход от одного уровня к другому происходит не в конкретный момент времени, а может произойти в любой момент времени в течение некоторого интервала времени, то это изображается как на рис. 5.12.

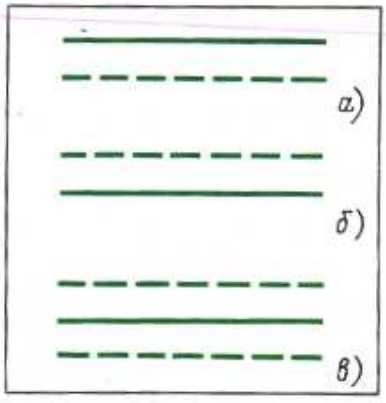


Рис. 5.10. Изображение уровней напряжения:
а — высокий; *б* — низкий; *в* — состояние высокого сопротивления

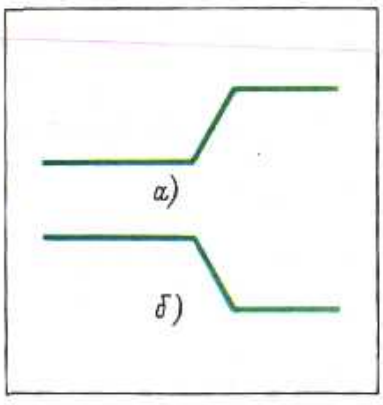


Рис. 5.11. Изображение фронтов:
а - фронт; *б* - срез

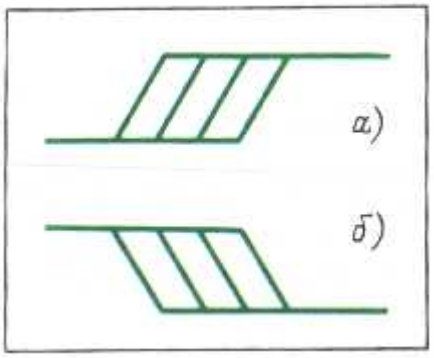


Рис. 5.12. Изображение фронтов в неопределенный момент времени

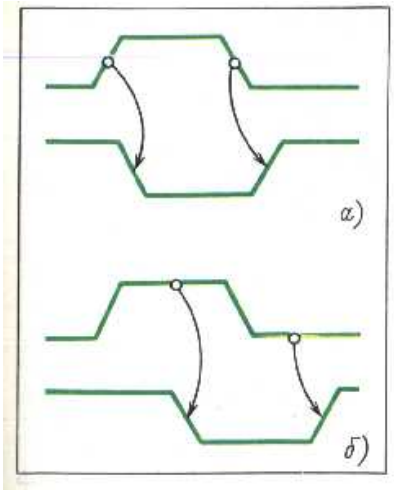


Рис. 5.13. Изображение зависимости одного сигнала от другого

Изменение одного сигнала может быть причиной изменения другого сигнала. Если изменение вызывается фронтом сигнала, то это изображается как на рис. 5.13,я, если уровнем сигнала -то как на рис. 5.13,6. При этом кружок отмечает тот элемент сигнала, который вызывает изменение, а стрелка указывает на изменение зависимого сигнала. Изменение одного сигнала может быть причиной изменений нескольких сигналов, тогда это изображается, как показано на рис. 5.14, и наоборот, изменение некоторого сигнала может вызываться только определенными изменениями нескольких других сигналов (рис. 5.15).

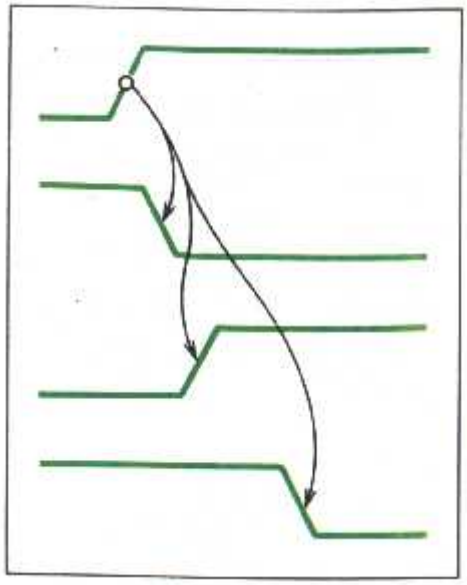


Рис. 5.14. Изображение зависимости нескольких сигналов от одного

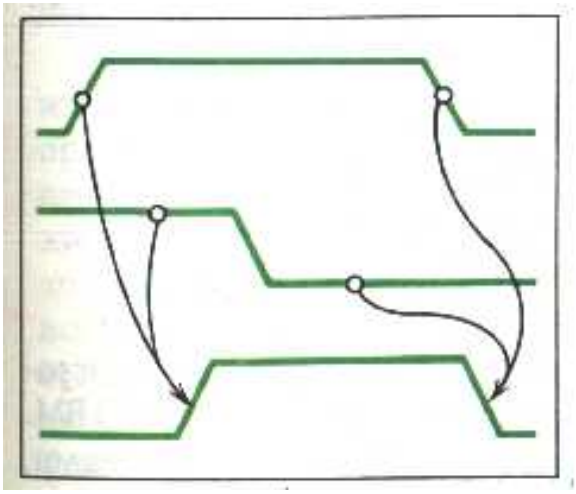


Рис. 5.15. Изображение зависимости одного сигнала от нескольких других

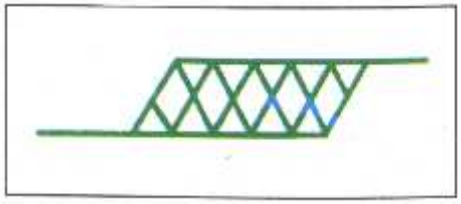


Рис. 5.16. Изображение неопределенного сигнала

Если уровень на входе микросхемы не влияет никак на ее работу или уровень на выходе не определен (не устанавливается), то это изображается так, как показано на рис. 5.16. Если необходимо изобразить на временной диаграмме состояния и Фронты нескольких сигналов, которые ведут себя одинаково с точки зрения переключений, но могут иметь разные уровни, то это делается так, как показано на рис. 5.17. Длительности интервалов времени между какими-либо изменениями сигналов изображаются с помощью букв или цифр на выносных линиях (рис. 5.18). На этом же рисунке показано, где пишутся буквенные обозначения сигналов (слева рядом с соответствующей временной диаграммой).

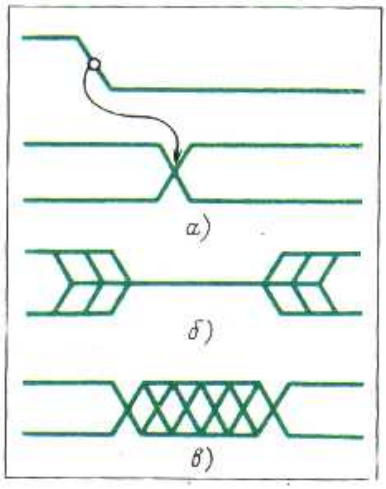


Рис. 5.18. Изображение временных интервалов

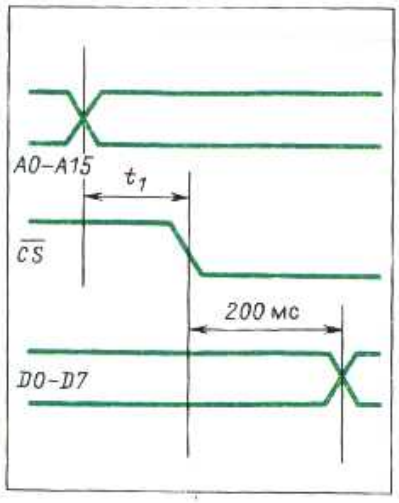


Рис. 5.17. Изображение нескольких сигналов, ведущих себя идентично с точки зрения переходов с уровня на уровень

5.4. МИКРОСХЕМЫ, РЕАЛИЗУЮЩИЕ ЛОГИЧЕСКИЕ ФУНКЦИИ

В этом и следующем параграфах будут описаны микросхемы, используемые в ПМ-ЭВМ. Как правило, инженеров, имеющих дело с микросхемами, не интересует их внутреннее устройство, их интересуют только выполняемые ими функции и характеристики входов и выходов. Функции микросхем будут описываться с помощью таблиц, в которых будут указываться входные и выходные комбинации сигналов низкого и высокого уровней. Сигнал низкого уровня будет обозначаться буквой L, высокого — буквой H, фронт — стрелкой t, срез — стрелкой 4. Если сигнал никак не влияет на работу микросхемы в каком-либо режиме, он будет обозначаться символом *X*. Для описания работы микросхем будут также применяться таблицы истинности и временные диаграммы. Для каждой микросхемы будет приводиться ее условное графическое изображение. Номера выводов, на которые подается напряжение питания, и номера общих выводов перечислены в приложении 3.

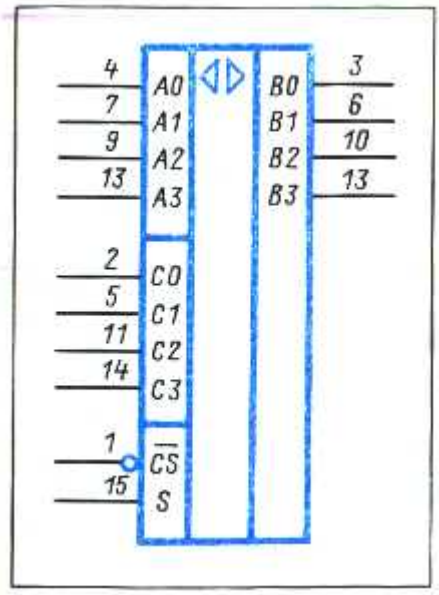


Рис. 5.19. Микросхема К5 89АП16

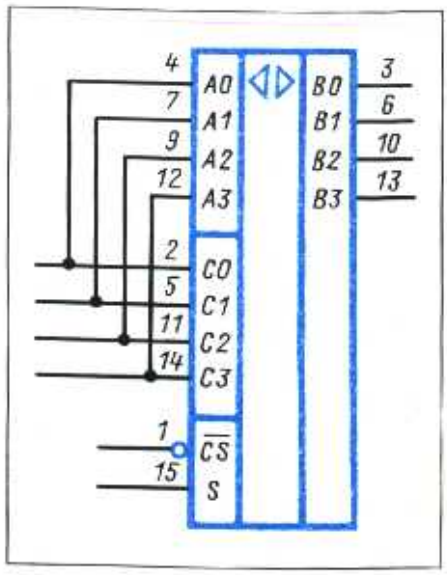


Рис. 5.20. МикросхемаК589АП16 как двунаправленный буфер

Микросхема К589АП16 (рис. 5.19). Эта микросхема содержит четыре элемента, каждый из которых предназначен для организации одной линии двунаправленной шины передачи данных и называется шинным формирователем. Все четыре элемента имеют общие управляющие входы CS и S. Каждый элемент имеет вход A, выход C с тремя состояниями и вывод B, который работает как вход или как выход

с тремя состояниями. Когда на управляющем входе CS высокий уровень, выходы С и В находятся в отключенном состоянии (табл5.1).

Если на входе CS низкий уровень, то сигналом на входе S можно управлять направлением передачи. При низком уровне на входе S сигнал передается от входа А на выход В, т. е. уровень сигнала на выходе такЬй же, как и на входе. Выход С при этом находится в отключенном состоянии. При высоком уровне на входе S сигнал передается от входа В к выходу С, а выход В находится в отключенном состоянии. Таким образом, вывод В может работать и как вход для выхода С, и как выход для входа А, и его можно использовать для организации одной линии двунаправленной шины для передачи данных. Выводы А и С можно соединять друг с другом (рис. 5.20), тогда эти выводы можно также использовать как одну двунаправленную линию, направлением передачи по которой, управляет вход S.

Таблица5.1

Уровень сигнала на управляющих входах сигнала		Направление передачи		Выход в отключенном состоянии
CS	**S**	**от входа**	**к выходу**	
H L L	**X** **L H**	**Нет А В**	**Нет В С**	**В, С С В**

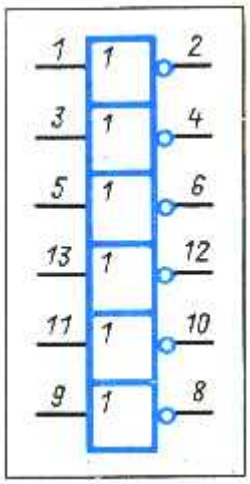


Рис. 5.21. Микросхема К155ЛН1

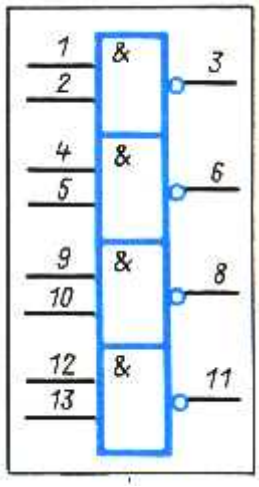


Рис. 5.22. Микросхема К155ЛАЗ

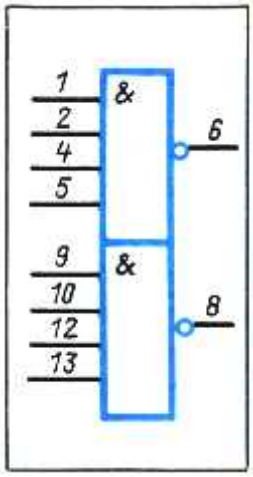


Рис. 5.23. Микросхема К155ЛА1

Микросхема К155ЛН1 (рис. 5.21). Эта микросхема содержит шесть элементов, каждый из которых имеет один вход и один выход и выполняет логическую функцию НЕ (табл. 5.2 и 5.3).

Микросхема К155ЛАЗ (рис. 5.22). Эта микросхема содержит четыре элемента, каждый из которых имеет два входа и один выход и выполняет функцию И — НЕ при позитивной и функцию ИЛИ — НЕ при негативной логике. На рис. 5.22 приводится условное графическое изображение этой микросхемы при позитивной логике. Таблица 5.4 описывает работу каждого двухвходового элемента; табл. 5.5 и табл. 5.6 — таблицы истинности для позитивной и негативной логик соответственно.

Таб л и ц а 5.2

Вход	Выход
L	H
H	L

Таблица 5.3

Вход	Выход
0	1
1	0

Таб лица 5.4

Входы		Выход
L	**L**	**H**
L	**H**	**H**
H	**L**	**H**
H	**H**	**L**

Та б л и ц а 5.5

Входы		Выход
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Т а б л и ц а 5.6

Входы		Выход
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

Микросхемы К155ЛА1 и К155ЛА2. Эти микросхемы выполняют функции, аналогичные функциям микросхемы К155ЛА3, т. е. функции И-НЕ при позитивной и функции ИЛИ-НЕ при негативной логиках. Микросхема К155ЛА1 (рис. 5.23) содержит два четырехвходовых элемента, и поэтому выполняемая ею функция называется и обозначается 4И — НЕ (4ИЛИ- НЕ). Так как полные таблицы, описывающие работу этой микросхемы, были бы велики, приведем их в сокращенном виде (табл. 5.7-5.9). Таблица 5.7 описывает работу одного элемента микросхемы К155ЛА1, табл. 5.8 и 5.9 - таблицы истинности при позитивной и негативной логиках соответственно.

Та б ли ца 5.7

Входы				Выход
L	X	X	X	H
X	L	X	X	H
X	X	L	X	H
X	X	X	L	H
H	H	H	H	L

Таблица 5.8

Входы				Выход
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1
1	1	1	1	0

Таблица 5.9

Входы				Выход
1	X	X	X	0
X	1	X	X	0
X	X	1	X	0
X	X	X	1	0
0	0	0	0	1

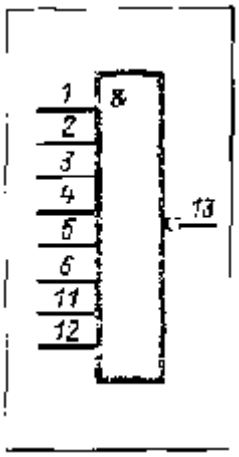


Рис. 5.24. Микросхема К155ЛА2

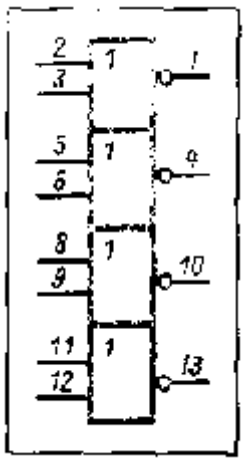


Рис. 5.25. Микросхема К155ЛЕ1

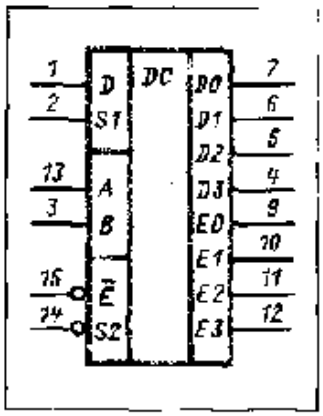


Рис. 5.26. Микросхема К155ИД4

Из табл. 5.7 видно, что если хотя бы на одном из четырех входов элемента присутствует низкий уровень, то на выходе будет высокий уровень. Только когда на всех четырех входах высокий уровень, то на выходе — низкий уровень.

Микросхема К155ЛА2 (рис. 5.24, табл. 5.10) содержит один восьмивходовый элемент, который выполняет функцию 8И — НЕ при позитивной логике (табл. 5.11) и функцию 8ИЛИ — НЕ (табл. 5.12) при негативной логике.

Микросхема К155ЛЕ1 (табл. 5.13, рис. 5.25) содержит четыре элемента, каждый из которых имеет два входа и один выход и выполняет функцию ИЛИ — НЕ при позитивной логике(табл. 5.14) или функцию И — НЕ (табл. 5.15) при негативной логике.

Таблица 5.10

Входы								Выход
L	X	X	X	X	X	X	X	H
X	L	X	X	X	X	X	X	H
X	X	L	X	X	X	X	X	H
X	X	X	L	X	X	X	X	H
X	X	X	X	L	X	X	X	H
X	X	X	X	X	L	X	X	H

X	X	X	X	X	X	L	X	н
X	X	X	X	X	X	X	L	н
H	н	н	н	н	н	H	H	L

Таблица 5.11

Входы								Выход
0	X	X	X	X	X	X	X	1
X	0	X	X	X	X	X	X	1
X	X	0	X	X	X	X	X	1
X	X	X	0	X	X	X	X	1
X	X	X	X	0	X	X	X	1
X	X	X	X	X	0	X	X	1
X	X	X	X	X	X	0	X	1
X	X	X	X	X	X	X	0	1
1	1	1	1	1	1	1	1	0

Таблица 5.12

Входы								Выход
1	X	X	X	X	X	X	X	0
X	1	X	X	X	X	X	X	0
X	X	1	X	X	X	X	X	0
X	X	X	1	X	X	X	X	0
X	X	X	X	1	X	X	X	0
X	X	X	X	X	1	X	X	0
X	X	X	X	X	X	1	X	0
X	X	X	X	X	X	X	1	0
0	0	0	0	0	0	0	0	1

Таблица 5.13

Входы		Выход
L	L	H
L	H	L
H	L	*L*
H	H	L

Таблица 5.14

Входы		Выход
0	0	1
0	1	0
1	0	0
1	1	0

Микросхема К155ИД4 (рис. 5.26, табл. 5.16, 5.17). Эта микросхема содержит два дешифратора-мультиплексора.

Прежде всего обратим внимание на то, что выводы В и А являются входами для обоих дешифраторов-мультиплексоров. Каждый дешифратор-мультиплексор имеет свои управляющие входы: выводы SI, D и S2, Е. Из табл. 5.16 видно, что если на управляющем входе S1 высокий уровень, а на входе D низкий, то на выходах DO — D3 высокий уровень независимо от того, какие уровни на входах В и А. Так же и для второго дешифратора из табл. 5.17 видно, что если на управляющих входах S2 и Е высокие уровни, то на выходах ЕО-Е3 тоже высокие уровни независимо от того, какие уровни на входах В и А. Поэтому, чтобы использовать эту микросхему как два дешифратора, необходимо подать на управляющие входы уровни, разрешающие работу выходов (низкий и высокий уровни на входы S1 и D соответственно, и низкие уровни на входы S2 и Е). Тогда в зависимости от уровней (т. е. от кодовой комбинации) на входах В и А низкий уровень появится на одном из выходов DO-D3 и ЕО-Е3.

Таблица 5.15

Входы		Выход
0	0	1
0	1	1
1	0	1
1	1	0

Таблица 5.16

Входы				Выходы			
S1	D	B	A	DO	DI	D2	D3
H	X	X	X	H	H	H	H
X	L	X	X	H	H	H	H
L	H	L	L	L	H	H	H
L	H	L	H	H	L	H	H
L	H	H	L	H	H	L	H
L	H	H	H	H	H	H	L

Таблица 5.17

Входы				Выходы			
S2	E	B	A	EO	E1	E2	E3
H	X	X	X	H	H	H	H
X	H	X	X	H	H	H	H
L	L	L	L	L	H	H	H
L	L	L	H	H	L	H	H
L	L	H	L	H	H	L	H
L	L	H	H	H	H	H	L

Мультиплексором называется схема, которая позволяет передавать сигнал, поступающий на ее вход, на разные выходы, причем то, на какой выход передается сигнал, определяется кодом на других управляющих входах. Микросхему К155ИД4 можно использовать как два мультиплексора. Каждый из них может передавать сигнал, поступающий на его вход, на четыре разных выхода. Входом для одного мультиплексора может служить вывод S1, для другого - вывод S2 или E. Тогда при соответствующем сигнале на втором управляющем входе и в зависимости от кода на входах B и A на одном из выходов (DO-D3 для одного мультиплексора и EO-E3 для другого) будет тот же сигнал, что и на входе.

5.5. МИКРОСХЕМЫ, СОДЕРЖАЩИЕ ЭЛЕМЕНТЫ ПАМЯТИ

Микросхема К155ТМ2 (рис. 5.27, табл. 5.18). Эта микросхема содержит два D-трштера. Поясним работу D-триггеров микросхемы К155ТМ2, пользуясь табл. 5.18. Каждый D-триггер этой микросхемы имеет вход предустановки S и сброса R. Из таблицы видно, что при подаче низкого уровня на вход S на прямом выходе триггера Q появится высокий уровень, а на инверсном выходе Q - низкий Уровень. Низкий уровень на входе R устанавливает выходы триггера в противоположные состояния. Если низкий уровень присутствует хотя бы на одном из входов S или R, то сигналы на входах C и D никак не влияют на состояние триггера. Если сигнал низкого уровня подавать одновременно на входы S и R, то на обоих выходах (прямом и инверсном) установится высокий уровень; если эти сигналы одновременно снять, то триггер может установиться в любое состояние, поэтому такой ситуации необходимо избегать.

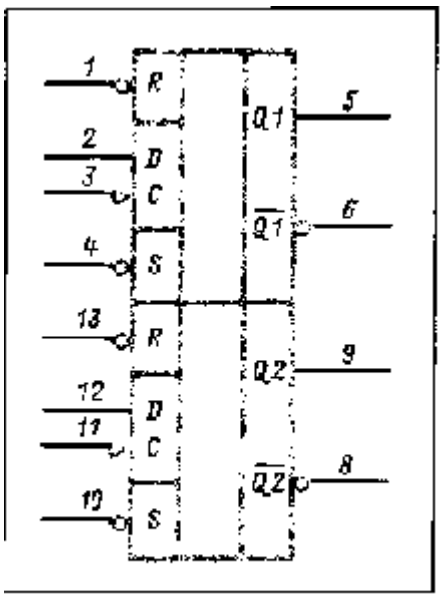


Рис. 5.27. **Микросхема К155ТМ2**
Рис. 5.27. Микросхема К155ТМ2

Таблица 5.18

Входы				Выходы	
S	**R**	**C**	**D**	**Q**	**Q**
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	L	X	Qs	Qs
H	H	H	X	Qs	Qs
H	H	1	X	Qs	*Qs*
H	H	t	H	H	L
H	H	t	L	L	H

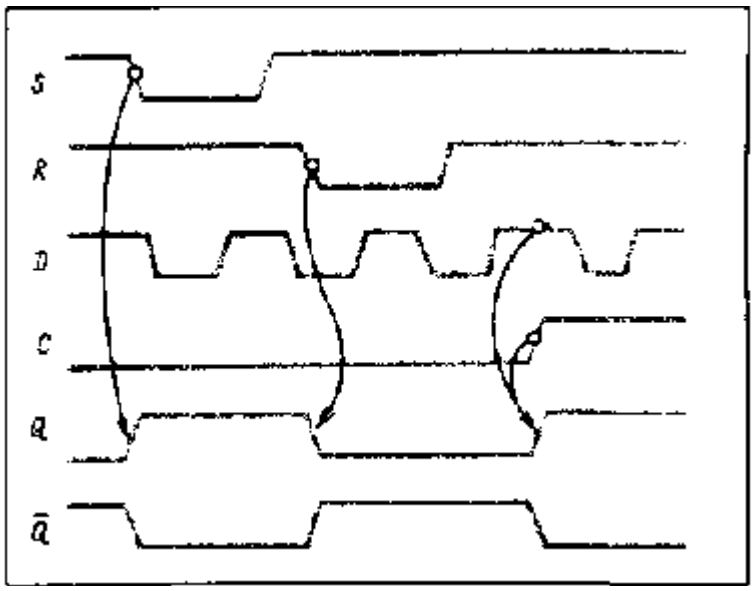


Рис. 5.28. Временная диаграмма работы микросхемы К155ТМ2

Буквой Q_s в табл. 5.18 обозначено состояние триггера, которое было до подачи указанных управляющих сигналов. Из табл. 5.18 видно, что низкий уровень, высокий уровень, а также переход от высокого к низкому уровню на входе С никак не влияют на состояние триггера. Только переход от низкого уровня к высокому устанавливает на выходе триггера тот уровень, который в данный момент присутствует на входе D, а на инверсном выходе устанавливает инверсный уровень. На рис. 5.28 приведена временная диаграмма, иллюстрирующая работу D-триггера микросхемы К155ТМ2.

Таблица 5.19

Входы			Выходы
C	**D**	**Q**	**0**
L	X	Qs	*Qs*
H	H	H	*L*
H	L	L	H

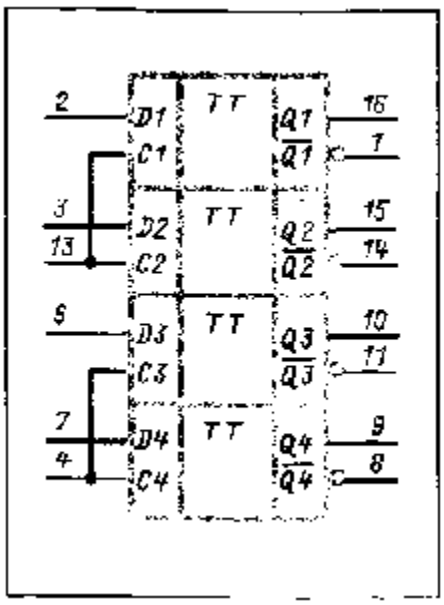


Рис. 5.29. Микросхема К155ТМ7

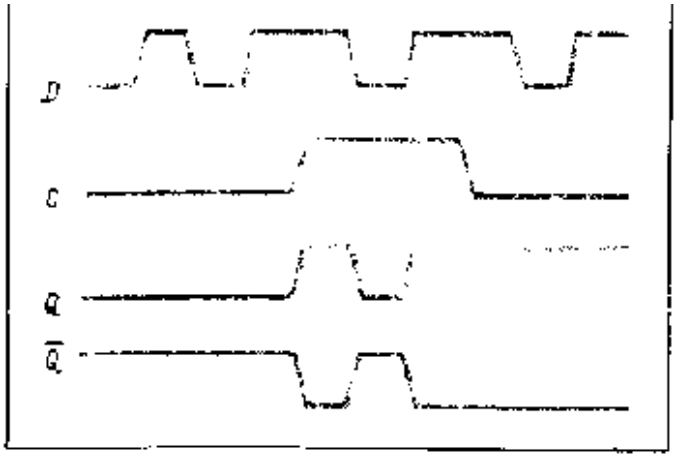


***Рис.** 5.30.* Временная диаграмма работы микросхемы К155ТМ7

Микросхема К155ТМ7 (рис. 5.29, табл. 5.19). Эта микросхема содержит четыре D-триггера.

Каждый триггер имеет вход D, выходы Q и Q и управляющий вход С, но управляющие входы всех триггеров не подключены к отдельным выводам микросхемы, а соединены попарно и подключены к выводам *4* и *13*. Поэтому триггерами этой микросхемы нельзя управлять по отдельности. Из табл. 5.19 видно, что если на управляющем входе С низкий уровень, то уровень сигнала на выходе триггера Q остается неизменным, а если высокий уровень, то уровень сигнала на выходе триггера повторяет уровень на входе D. Временная диаграмма, поясняющая работу триггера, приведена на рис. 5.30. Триггер такого типа часто называют "защелкой", так как с его помощью можно в любой момент зафиксировать уровень какого-либо сигнала.

Микросхема КР541РУ2 (рис. 5.31). Эта микросхема является оперативным запоминающим устройством емкостью 4 Кбит с организацией 1024х4 бита. Это значит, что внутри микросхемы содержится 1024 запоминающие ячейки, каждая из которых состоит из четырех двоичных разрядов. Для адресации ячеек памяти внутри микросхемы имеется 10 адресных входов АО — А9. Каждой комбинации сигналов на этих входах соответствует одна ячейка. Для передачи информации в ячейки памяти при записи и из ячеек памяти при считывании имеются выводы QO — Q3. При записи информации эти выводы работают как входы, а при считывании — как выходы с открытым коллектором. Вход WE управляет режимом записи и считывания: при высоком уровне на этом входе микросхема работает в режиме считывания, при низком — в режиме записи. Вход CS управляет работой выводов QO — Q3. При высоком уровне на этом входе информация не может быть считана или записана. В режиме считывания низкий уровень на входе CS разрешает работу выводов QO — Q3 как выходов, а в режиме записи — как входов. Временные диаграммы работы микросхемы в режиме записи считывания приведены на рис. 5.32.

Микросхема КР556РТ4 (рис. 5.33). Эта микросхема является постоянным запоминающим устройством емкостью 1 Кбит с организацией 256х4 бита, т. е. внутри микросхемы содержится 256 ячеек памяти, каждая из которых состоит из четырех разрядов. Информация, записанная в ПЗУ, не стирается при выключении напряжения питания. Для записи информации в ПЗУ используется специальное устройство, называемое *программатором*. Схема программатора для микросхемы К556РЕ4 (старое название микросхемы КР556РТ4) и режим записи информации не приводятся в этой книге.

Микросхема КР556РТ4 имеет восемь входов АО-А7 для адресации ячеек памяти, четыре выхода с открытым коллектором QO — Q3 для считывания содержимого ячейки и два управляющих входа CSO и CS1. Считывание информации может происходить только тогда, когда на обоих входах CSO и CS1 низкий уровень (рис. 5.34).

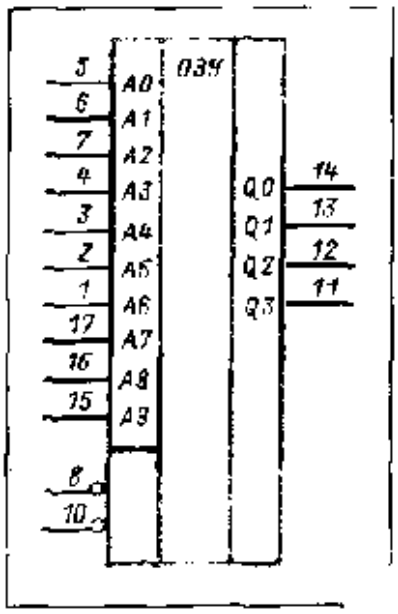


Рис. 5.31. Микросхема КР541РУ2

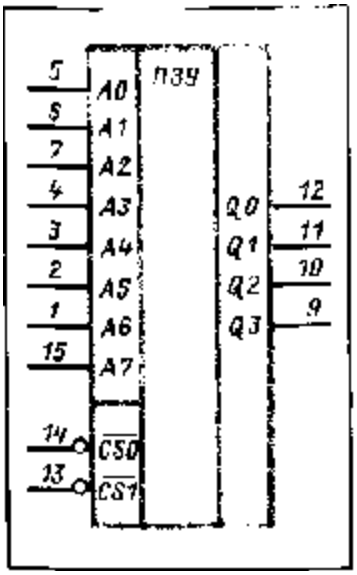


Рис. 5.33. Микросхема КР556РТ4

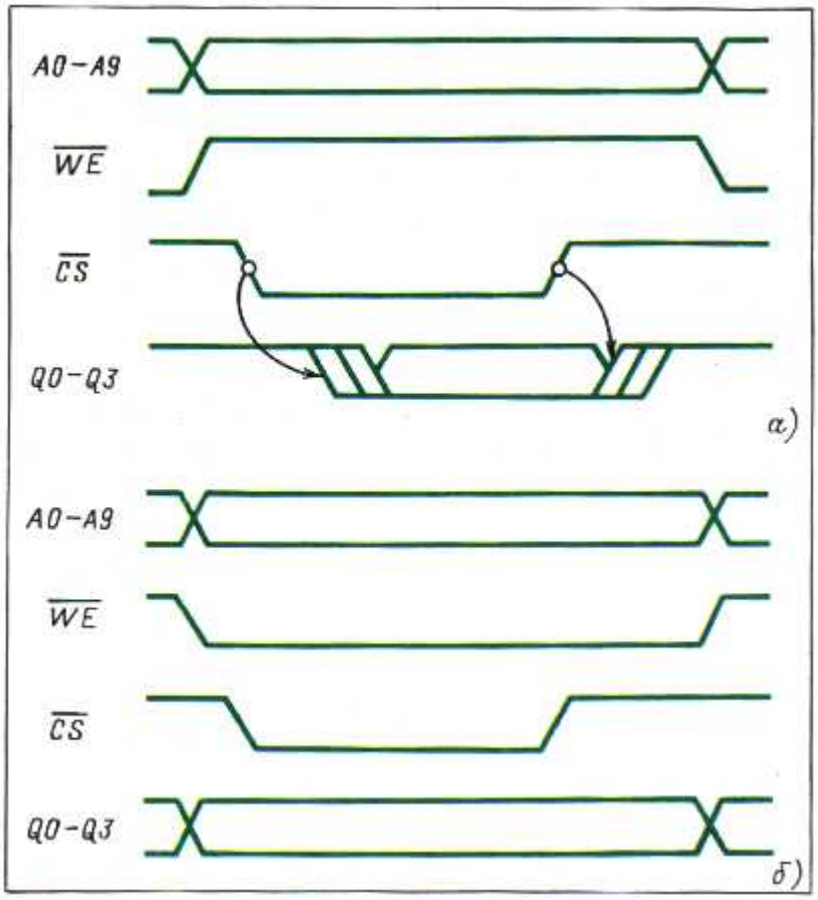


Рис. 5.32. Временная диаграмма работы микросхемы КР541РУ2

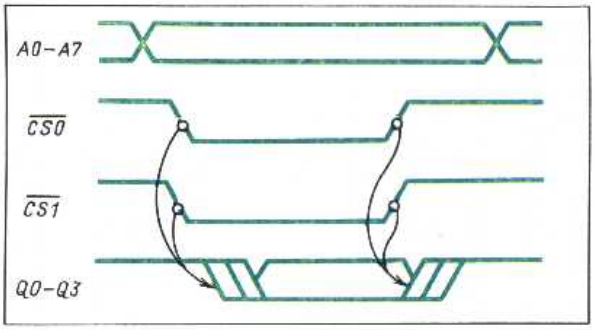


Рис. 5.34. Временная диаграмма работы микросхемы КР556РТ4 при считывании

6

СТРУКТУРА И ФУНКЦИОНИРОВАНИЕ МИКРОПРОЦЕССОРНОГО БЛОКА

6.1. МИКРОПРОЦЕССОР КР580ИК80А

В предыдущих главах уже приводились некоторые данные о микропроцессоре КР580ИК80А. В этой главе будет продолжено описание этого микропроцессора в основном с точки зрения электронных или, как говорят, аппаратурных особенностей его устройства и работы. Микропроцессор КР580ИК80А представляет собой центральный процессорный элемент, выполненный по и-МОП технологии в виде одной микросхемы. Микросхема упакована в прямоугольный пластмассовый корпус с двухрядным расположением выводов. Серия КР580 является развитием серии К580 и содержит несколько больших интегральных схем, на базе которых можно эффективно реализовывать различные микропроцессорные системы. Микросхема КР580ИК80А содержит 5000 транзисторов и имеет 40 выводов.

Приведем некоторые типовые характеристики микропроцессора.

Диапазон рабочих температур, °С..... От -10 до+70
Максимальная тактовая частота, МГц..... 2,5
Напряжение источников питания, В:
U_1.....-..... +12±0,6
U_2..... +5 ±0,25
Uз..... -5 ±0,25
Потребляемая мощность, мВт..... 1500
Быстродействие (количество операций типа регистр-регистр в секунду)..... 625 000

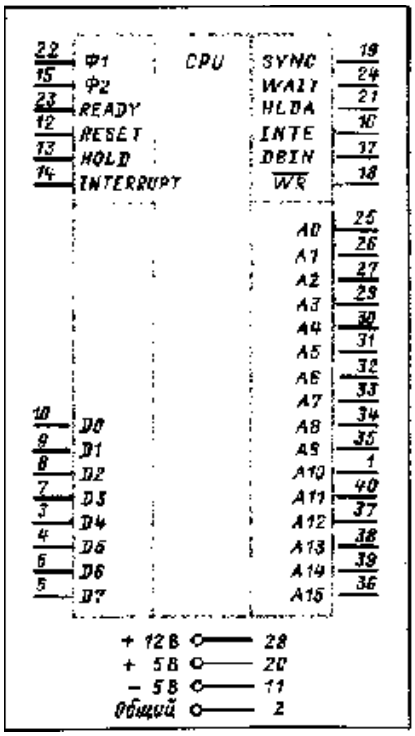


Рис. 6.1. Микропроцессор КР580ИК80А

Каждый выход микропроцессора обеспечивает $I^0_{вых}$ не более 1,8 мА и $I^1_{вых}$ не более 0,1 мА, т. е. может быть нагружен одним стандартным ТТЛ-входом серии К155. Напряжения питания необходимо подавать или одновременно, или в последовательности U_1, U_2, Uз и снимать в обратной последовательности. Микропроцессор КР580ИК80А относится к универсальным микропроцессорам, он имеет возможность работать в самых разнообразных режимах. Тот или иной режим может не использоваться в каждой конкретной конструкции микро-ЭВМ. Условное графическое изображение микропроцессора приводится на рис. 6.1, а функции выводов — в табл. 6.1.

Таблица 6.1

Обозначение вывода	Функциональное назначение вывода
AO-A15 DO-D7	Выходы, линии шины адреса Двунаправленные линии шины данных
	Сигналы управления шинами
DBIN	Выход, признак того, что шина данных находится в состоянии приема информации
WR HOLD	Выход, признак того, что шина данных находится в состоянии передачи информации Вход, переводит шины данных и адреса в состояние высокого сопротивления
HLDA	Выход, признак того, что шины данных и адреса находятся в состоянии высокого сопротивления
READY	Вход, переводит микропроцессор в состояние ожидания
WAIT	Выход, признак того, что микропроцессор находится в состоянии ожидания
Обозначение вывода	**Функциональное назначение вывода**
SYNC	Выход, признак того, что по шине данных передается управляющее слово микропроцессор
	Сигналы управления прерываниями
INTERRUPT INTE	Вход, запрос прерывания работы микропроцессора Выход, соответствует состоянию внутреннего триггера, управляющего прерыванием микропроцессора
	Сигналы синхронизации
Ф1,Ф2	Входы для тактовых импульсов
	Сигнал начального запуска
RESET	Вход, вызывает запись в программный счетчик адреса нулевой ячейки памяти

6.2. СИНХРОНИЗАЦИЯ

Как уже известно из § 4.3, каждая команда выполняется микропроцессором не мгновенно, а как последовательность машинных циклов. Микропроцессор КР580ИК80А имеет 10 типов машинных циклов (табл. 6.2), и все его команды состоят из комбинаций только этих циклов.

В состав команды может входить от одного до пяти циклов. Каждый машинный цикл также не является неделимой операцией, а состоит из машинных тактов. В состав машинного цикла микропроцессора КР580ИК80А может входить от трех до пяти машинных тактов. Машинный такт не является таким же законченным процессом, как машинный цикл, большинство сигналов микропроцессора вырабатываются в одном такте и снимаются в другом. Поэтому для машинных тактов нельзя привести такую же классификацию, как для машинных циклов. Как и в гл. 4, в тексте и на рисунках будем обозначать такты одного цикла Tl, T2, ... и т. д., циклы одной команды C1,C2,... и т. д.

Для того чтобы сформировать интервал времени, соответствующий одному машинному такту, на входы микропроцессора Ф1 и Ф2 подаются тактовые импульсы (синхроимпульсы). Длительность машинного такта равняется одному периоду синхроимпульсов. Все внутренние операции микропроцессора и формирование внешних сигналов происходят в моменты времени, определяемые синхроимпульсами.

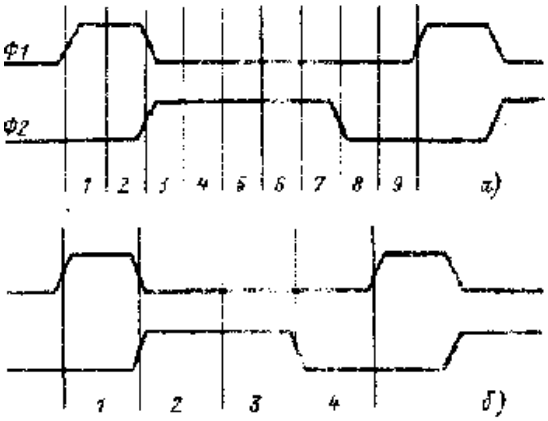


Рис. 6.2. **Тактовые импульсы Таблица 6.2**

Тип цикла	Управляющее слово, состоящее из разрядов шины данных DO — D7							
	DO	DI	D2	D3	D4	D5	D6	D7
ВЫБОРКА КОМАНДЫ	L	H	L	L	L	H	L	H
ЧТЕНИЕ ИЗ ПАМЯТИ	L	H	L	L	L	L	L	H
ЗАПИСЬ В ПАМЯТЬ	L	L	L	L	L	L	L	L
ЧТЕНИЕ ИЗ СТЕКА	L	H	H	L	L	L	L	H
ЗАПИСЬ В СТЕК	L	L	H	L	L	L	L	L
ВВОД С ВНЕШНЕГО УСТРОЙСТВА	L	H	L	L	L	L	H	L
ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО	L	L	L	L	H	L	L	L
ПРЕРЫВАНИЕ	H	H	L	L	L	H	L	L
ОСТАНОВ	L	H	L	H	L	L	L	H
ПРЕРЫВАНИЕ ВО ВРЕМЯ ОСТАНОВА	H	H	L	H	L	H	L	L

К уровням напряжения, частоте, фронтам и фазе синхроимпульсов предъявляются особые требования (рис. 6.2). Входы Ф1 и Ф2 не являются ТТЛ-входами. Это единственное исключение для микропроцессора КР580ИК80А. Для Ф1 и Ф2 напряжение сигнала низкого уровня должно быть в пределах от -0,3 до 0,8 В, а напряжение сигнала высокого уровня - в пределах от 10 до 12 В. Частота синхроимпульсов должна быть не более 2,5 МГц, переход с одного уровня напряжения на другой должен иметь длительность 20-30 не. На рис. 6.2,я приводится временная диаграмма сигналов Ф1 и Ф2, которая может быть получена с помощью микросхемы тактового генератора КР580ГФ24, который выпускается специально для микропроцессора КР580ИК80А (см. § 6.4). Из этой диаграммы видно, что если условно разбить период синхроимпульсов на девять интервалов, то сигнал Ф1 будет иметь высокий уровень в первых двух интервалах и низкий уровень во всех остальных, а сигнал Ф2 будет иметь высокий уровень в интервалах с третьего по седьмой и низкий уровень во всех остальных.

Сигналы Ф1 и Ф2 могут иметь и более простую временную диаграмму (рис. 6.2,6). Если не пользоваться для их генерации микросхемой КР580ГФ24, то получить такую диаграмму проще, чем стандартную. Требования к уровням, фронтам и частоте остаются такими же, как и для стандартной временной диаграммы.

6.3. ШИНЫ АДРЕСА, ДАННЫХ И УПРАВЛЕНИЯ

Шина адреса. Микропроцессор КР580ИК80А имеет 16-разрядную шину адреса А0-А15. Эта шина выполняет две функции. Первая функция состоит в передаче адреса ячейки памяти (стека) при обращении к памяти (стеку). С помощью 16 разрядов шины адреса можно адресовать 2^{16} ячеек памяти (т. е. 64 Кбайта). Вторая функция шины адреса — это передача адреса внешнего устройства при выполнении команд IN и OUT. В этом случае адрес внешнего устройства появляется на линиях АО-А7 и дублируется на линиях А8-А15. Так как для передачи адреса внешнего устройства используется фактически только восемь разрядов, то можно адресовать 256 (2^8) различных внешних устройств ввода/вывода.

Временная диаграмма работы шины адреса приведена на рис. 6.3. Адрес появляется на шине по фронту сигнала Ф2 в такте Т1 и поддерживается три такта до фронта сигнала Ф2 в такте ТХ. Такт ТХ может быть четвертым тактом (Т4) машинного цикла, если цикл имеет более трех тактов, и может быть тактом Т1 следующего машинного цикла, если данный цикл имеет три такта.

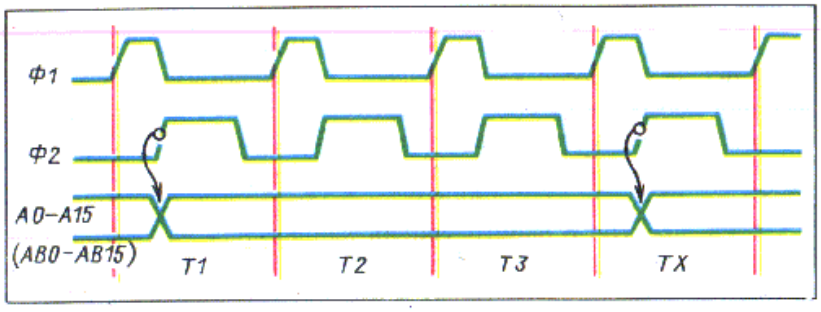


Рис. 6.3. Временная анаграмма работы шины адреса

Каждая линия шины адреса является ТТЛ-выходом с тремя состояниями и нагрузочной способностью, равной единице. Так как шину адреса необходимо подавать на многие блоки ПМ-ЭВМ, то необходимо буферизовать ее линии, т. е. увеличить их нагрузочную способность. Для этого каждая линия подключается к входу схемы, состоящей из двух последовательно включенных инверторов (рис. 6.4). В результате на выходе второго инвертора имеется сигнал такого же уровня, как и на входе первого, но выход инвертора микросхемы К155ЛН1 имеет нагрузочную способность, равную 10.

Шина данных. Микропроцессор КР580ИК80А имеет 8-разрядную двунаправленную шину данных DO — D7. Шина данных выполняет две функции. Первая функция — передача управляющего слова, вторая — обмен информацией между регистрами микропроцессора и блоками микро-ЭВМ. Шина данных — единственная двунаправленная шина микропроцессора. Ее выходы также являются выходами с тремя состояниями.

Рассмотрим функцию передачи управляющего слова. Известно (см. § 6.2), что микропроцессор КР580ИК80А имеет 10 типов машинных циклов. Информация о том, какого типа цикл будет выполняться, передается в начале каждого цикла по линиям шины данных и называется *управляющим словом.* Все управляющие слова перечислены в табл. 6.2. Управляющее слово или только его часть можно запомнить с помощью дополнительных триггеров и использовать затем для формирования сигналов шины управления. Как это делается, будет описано ниже.

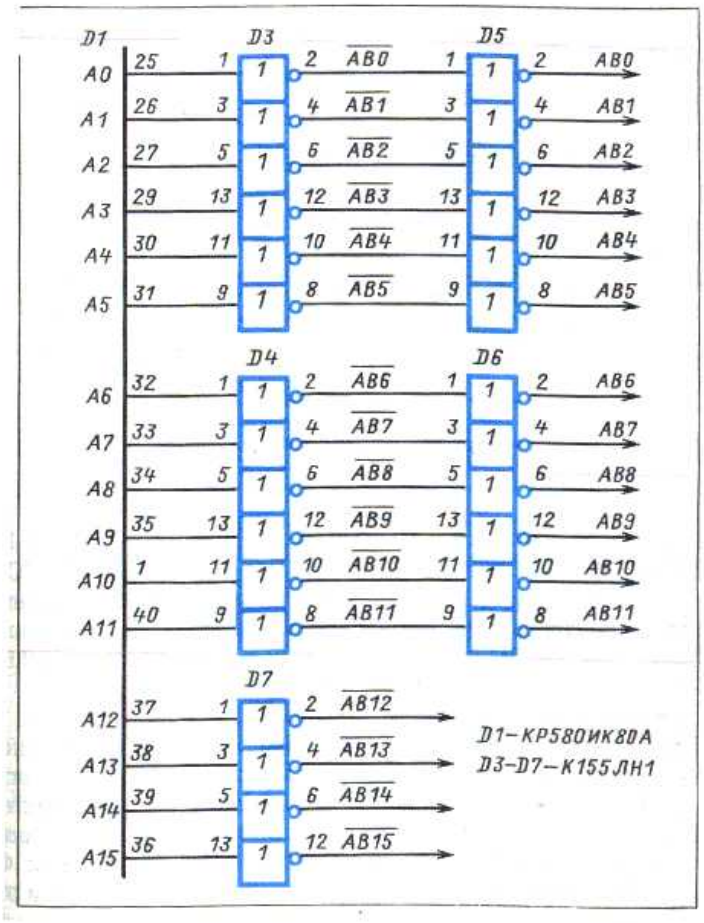


Рис. 6.4. Шина адреса

Для того чтобы показать, что идет процесс передачи управляющего слова, используется выход микропроцессора SYNC. Временная диаграмма приводится на рис. 6.5. Из нее видно, что передача управляющего слова по шине данных начинается по фронту сигнала Ф2 в такте Т1 и заканчивается по тому же фронту сигнала Ф2 в такте Т2.

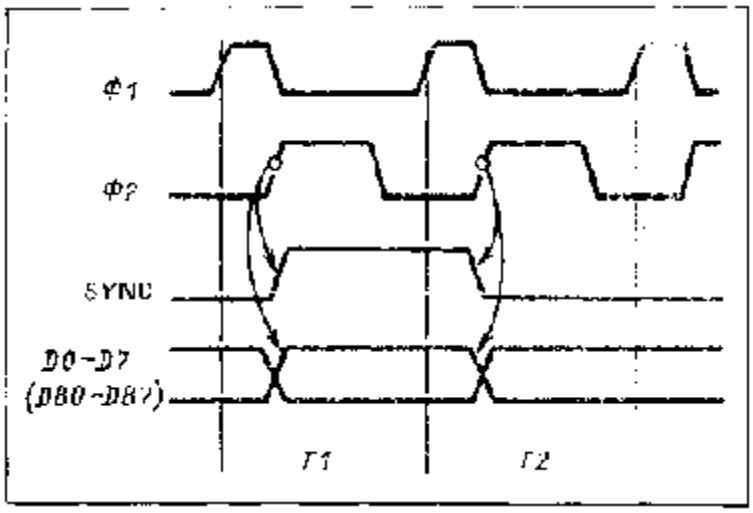


Рис. 6.5. Временная диаграмма процесса передачи управляющего слова

Рассмотрим вторую функцию шины данных — обмен информацией между регистрами микропроцессора и блоками микро-ЭВМ. Во время циклов ВЫБОРКА КОМАНДЫ, ЧТЕНИЕ ИЗ ПАМЯТИ, ЧТЕНИЕ ИЗ СТЕКА, ВВОД С ВНЕШНЕГО УСТРОЙСТВА, ПРЕРЫВАНИЕ информация передается по шине данных в один из внутренних регистров микропроцессора из какого-либо блока микро-ЭВМ, а во время циклов ЗАПИСЬ В ПАМЯТЬ, ЗАПИСЬ В СТЕК, ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО - из внутреннего регистра в какой-либо блок.

Для указания направления передачи информации по шине данных микропроцессор имеет два выхода: DBIN и WR. Когда информация передается в микропроцессор, то в этом машинном цикле на выходе DBIN появляется высокий уровень (рис. 6.6,а). Сигнал DBIN появляется по фронту сигнала Ф2 в такте Т2 и снимается по фронту Ф2 в такте Т3. Именно в это время внешние схемы должны поместить на шину данных код, который должен быть записан в один из внутренних регистров. Когда информация передается из микропроцессора, то в этом машинном цикле вырабатывается сигнал WR низкого уровня (рис. *6.6,б)*. Этот сигнал вырабатывается по фронту сигнала Ф1 в такте Т2 и снимается по фронту Ф1 в такте Т4 этого же машинного цикла (если цикл состоит более чем из трех тактов) или в такте Т1 следующего машинного цикла (если данный цикл состоит из трех тактов). На шине данных ЕЮ — D7 немного ранее появления сигнала WR микропроцессор помещает данные и снимает их позднее снятия сигнала WR, поэтому внешние схемы могут использовать этот сигнал для управления записью информации.

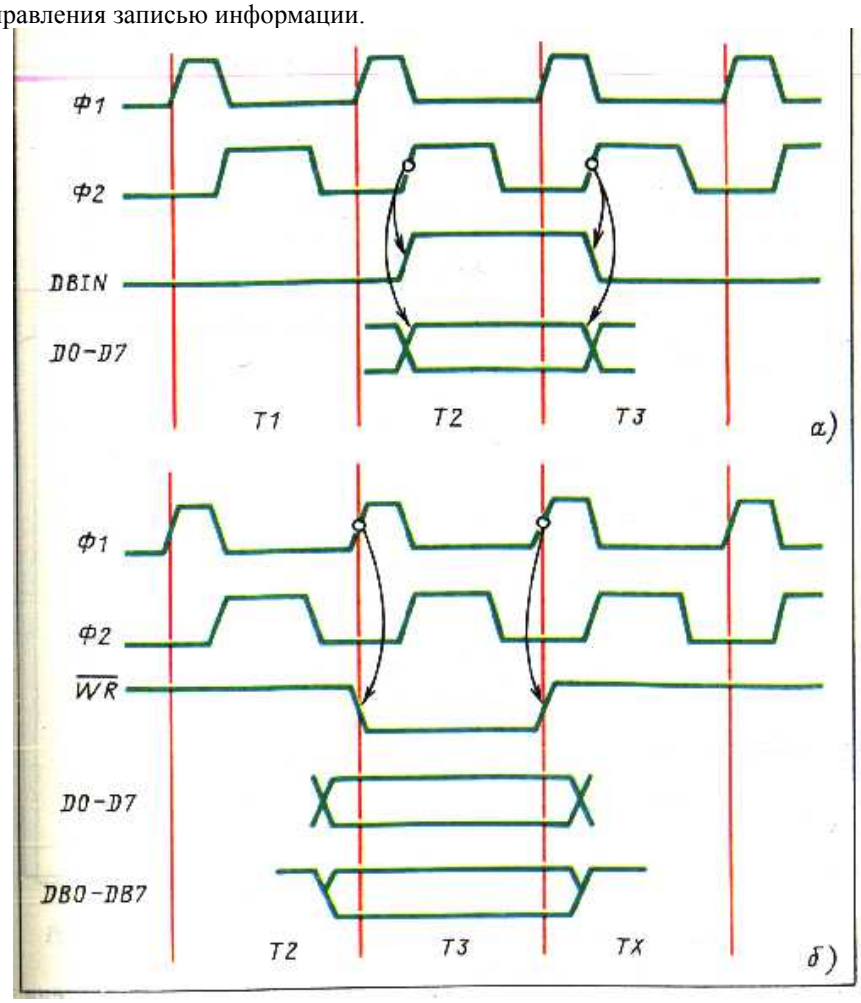


Рис. 6.6. Временная диаграмма работы шины данных

Каждая линия шины данных имеет единичную нагрузочную способность и поэтому нуждается в буферизации. Поскольку линии шины данных двунаправленны, необходим и двунаправленный буфер. Схема такого буфера, состоящая из двух микросхем К589АП16 (D9, D10), приводится на рис. 6.7. На входы *1* микросхем D9, D10 подан низкий уровень, и поэтому они работают все время в режиме передачи информации. Направление передачи определяется уровнем буферизованного сигнала DBIN, который подключен к входам *15* микросхем D9, D10. Заметим, что направление передачи не управляется сигналами WR и SYNC .

Если уровень сигнала DBIN высокий, то шинные формирователи передают информацию в направлении *к* микропроцессору (от DBO — DB7 к DO — D7), если нет — то в обратном направлении (от ЕЮ — D7 к DBO — DB7), т. е. направление передачи шинных формирователей при низком уровне сигнала DBIN соответствует тому, которое должно быть при выработке сигналов WR и SYNC.

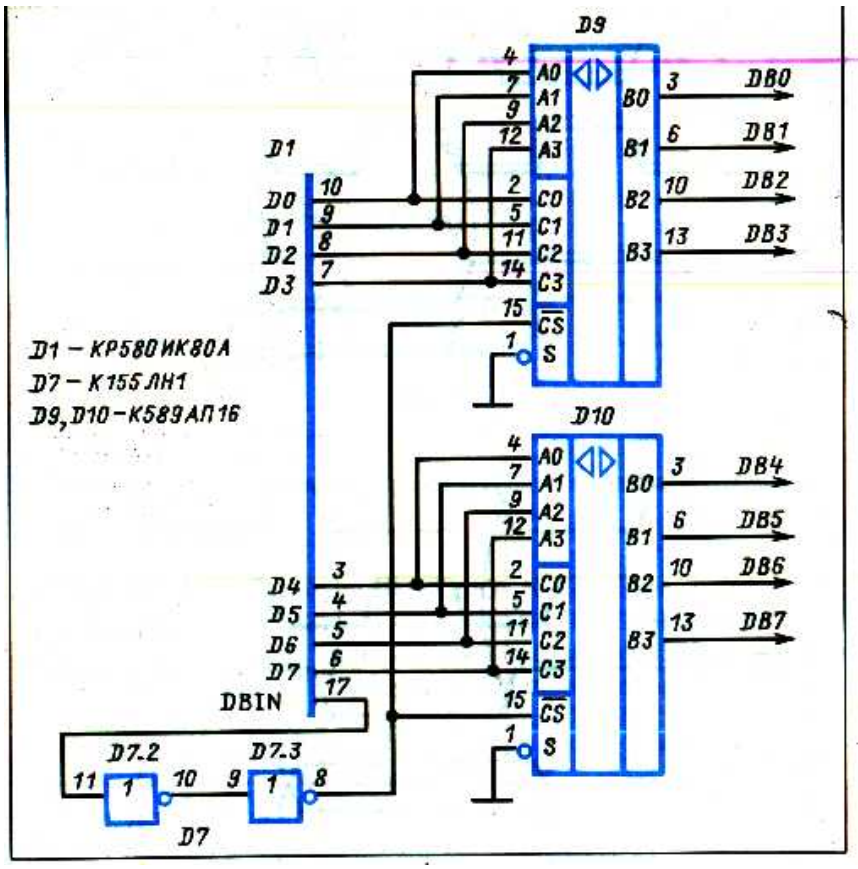


Рис. 6.7. Шина данных

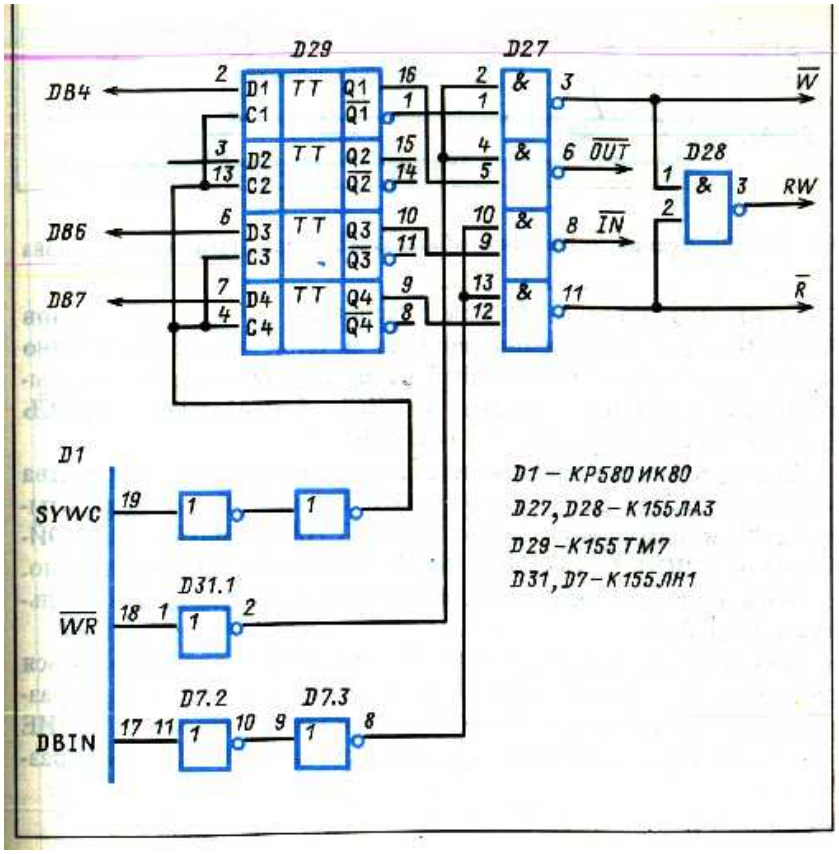


Рис. 6.8. Шина управления

Шина управления. Микропроцессор КР580ИК80А не обладает готовой шиной управления, как шиной адреса и шиной данных. Поэтому эта шина организуется с помощью дополнительной внешней схемы

(микросхемы D27, D29), называемой иногда системным контроллером (СК), которая использует управляющее слово и управляющие сигналы микропроцессора (рис. 6.8). Часть разрядов управляющего слова (D4, D6, D7) фиксируется в триггерах микросхемы К155ТМ7 (D29) по срезу сигнала SYNC (рис. 6.9). Сигнал SYNC пропущен через два инвертора для увеличения нагрузочной способности (эта схема, хотя и работоспособна, не будет применяться в ПМ-ЭВМ). Позднее будет приведена схема для фиксации управляющего слова, где сигнал на управляющие входы триггеров будет вырабатываться с помощью микросхемы КР580ГФ24. Сигналы шины управления вырабатываются с помощью сигналов с выходов триггеров микросхем D29 и сигналов DBIN и WR.

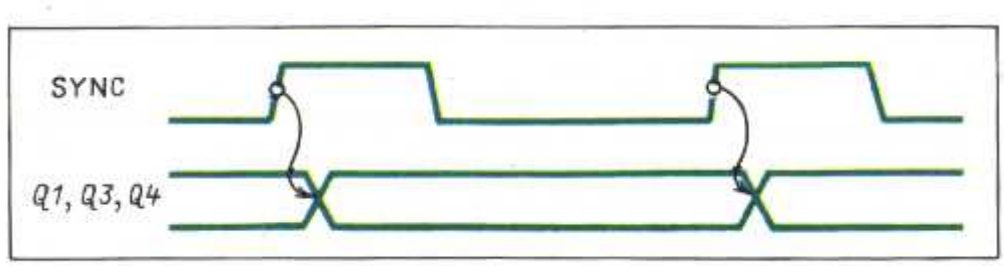


Рис. 6.9. Временная диаграмма работы фиксаторов управляющего слова

В ПМ-ЭВМ команды, данные и содержимое стека хранятся в едином блоке памяти, состоящем из ОЗУ и ПЗУ. Поэтому Для управления чтением из блока достаточно иметь один сигнал R, который будет вырабатываться при выполнении машинных циклов ВЫБОРКА КОМАНДЫ, ЧТЕНИЕ ИЗ ПАМЯТИ, ЧТЕНИЕ ИЗ СТЕКА. Назовем условно эту группу циклов ЧТЕНИЕ. Для управления записью в блок также достаточно иметь один сигнал W, который будет вырабатываться при выполнении машинных циклов ЗАПИСЬ В ПАМЯТЬ, ЗАПИСЬ В СТЕК. Эту группу назовем ЗАПИСЬ.

Для управления вводом и выводом на внешние устройства необходимы два сигнала OUT и IN, которые будут вырабатываться при выполнении циклов ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО и ВВОД С ВНЕШНЕГО УСТРОЙСТВА соответственно.

Формирование сигналов шины управления поясним, пользуясь табл. 6.3.

Таблица 6.3

Тип цикла или группы циклов	Вырабатываемый сигнал DBIN или WR	Используемый разряд	Уровень напряжения на выходах триггеров D29 микросхемы				Вырабатываемый сигнал шины управления
			9	1	16	10	
ЧТЕНИЕ	DBIN	D7	H	H	L	L	R, RW
ЗАПИСЬ	WR	D4	L	H	L	L	W, RW
ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО	WR	D4	L	L	H	L	OUT
вводе ВНЕШНЕГО УСТРОЙ-СТВА	DBIN	D6	L H		L	H	IN

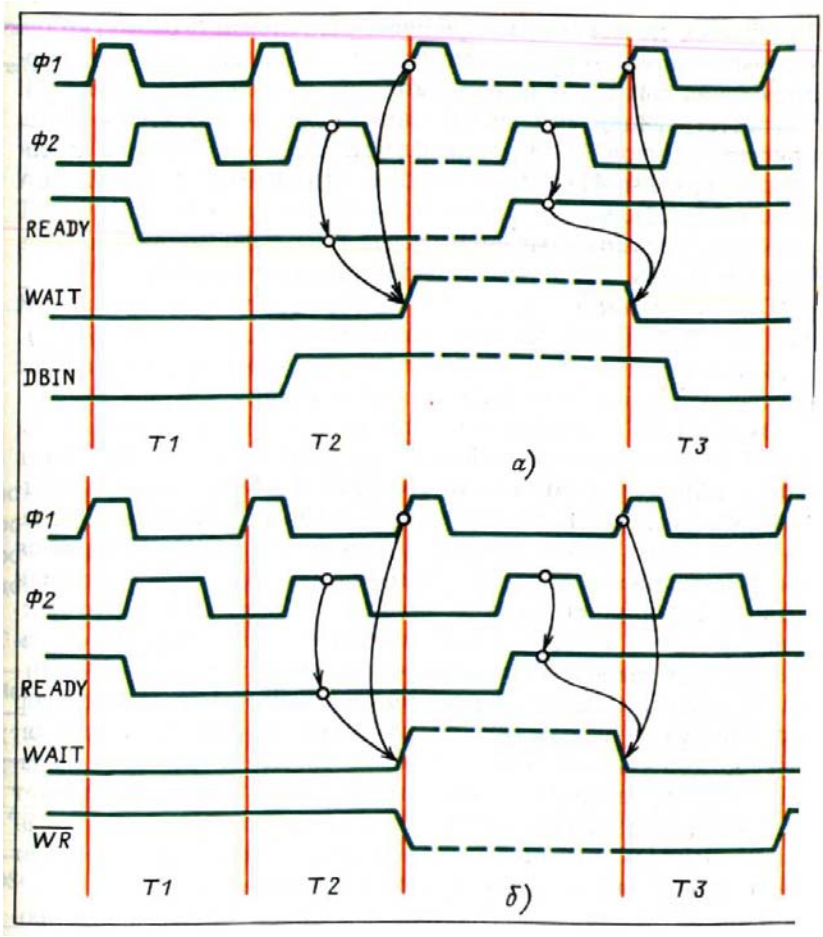


Рис. 6.10. Временная диаграмма циклов чтения и записи с переходом в режим ожидания:
а - цикл чтения; *б* - цикл записи

Для формирования сигнала R естественно воспользоваться разрядом D7 управляющего слова, так как именно в этом разряде имеется сигнал высокого уровня в циклах группы ЧТЕНИЕ (см. табл. 6.2), а для формирования сигналов OUT и IN — разрядами D4 и D6, потому что именно в этих разрядах сигнал имеет высокий уровень при выполнении соответствующих циклов и низкий уровень во всех остальных циклах. Сложнее дело обстоит с формированием сигнала W для циклов группы

ЗАПИСЬ. Казалось, было бы естественно воспользоваться разрядом D1 управляющего слова, в котором передается сигнал низкого уровня для циклов группы ЗАПИСЬ. Но в этом разряде сигнал низкого уровня бывает также и в цикле ВЫВОД НА ВНЕШНЕЕ УСТРОЙСТВО, и так как в этом цикле также вырабатывается сигнал WR, то пришлось бы усложнять схему, чтобы сформировать два разных сигнала W и OUT. Поэтому для формирования сигнала W используется инвертированный разряд D4. Соответствующий ему сигнал снимается с инверсного выхода триггера (вывод 7). Этот сигнал имеет высокий уровень и для циклов группы ЧТЕНИЕ и ВВОД С ВНЕШНЕГО УСТРОЙСТВА (см. табл. 6.2), но при этом не формируются сигналы W и OUT (так как не в ырабатывается сигнал WR), а формируются лишь сигналы R и IN (так как вырабатывается сигнал DBIN).

Сигнал высокого уровня RW формируется из сигнала R или W и управляет работой дешифратора адреса (см. § 7.2). В заключение опишем еще некоторые управляющие сигналы микропроцессора КР580ИК80А и их функции.

Вход READY используется для того, чтобы удлинить циклы ввода и вывода информации в микропроцессор и обеспечить таким образом возможность его работы с менее быстродействующими устройствами. Если во время высокого уровня сигнала Ф2 в такте Т2 какого-либо цикла на вход READY подается высокий уровень, то происходит нормальный цикл ввода или вывода информации. Если же подается низкий уровень, то микропроцессор с этого момента прекращает свою работу, сохраняя на всех выходах имеющиеся уровни сигналов (рис. 6.10). Такой режим называется *режимом ожидания.* Микропроцессор сигнализирует о том, что он находится в режиме ожидания, выработкой сигнала WAIT высокого уровня. В режиме ожидания микропроцессор не выполняет никаких операций и может находиться как угодно долго — до тех пор, пока не будет подан сигнал READY высокого уровня. После этого он завершает начатую операцию и продолжает далее свою работу.

Вход RESET используется для начального запуска микропроцессора и для повторных перезапусков. При подаче высокого уровня на этот вход микропроцессор прекращает свою работу, помещает во все разряды счетчика команд нули и бездействует до тех пор, пока на входе RESET остается высокий уровень. Как только высокий уровень снимается, он начинает выполнять команду, расположенную в ячейке памяти с нулевым адресом.

6.4. ТАКТОВЫЙ ГЕНЕРАТОР И СХЕМА ПОШАГОВОГО ИСПОЛНЕНИЯ ПРОГРАММ

Для того чтобы получить тактовые импульсы с необходимыми для микропроцессора КР580ИК80А параметрами (см. § 6.2, рис. 6.2), можно использовать микросхему КР580ГФ24 (рис. 6.11, табл. 6.4). Эта микросхема является тактовым генератором, специально сконструированным для данного микропроцессора, и выполняет еще ряд дополнительных функций.

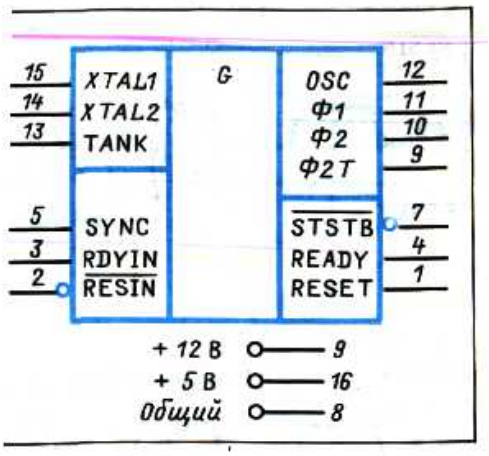


Рис. 6.11. Тактовый генератор КР580ГФ24

Таблица 6.4

Название	Назначение вывода
XTAL1 XTAL2	Выводы для подключения кварцевого резонатора
TANK	Вывод, используемый при применении кварцевого резонатора с обертоном
OSC	Выход для синусоидального сигнала с частотой кварцевого резонатора
Ф1.Ф2	Выходы для тактовых импульсов
Ф2Т	Выход для тактовых импульсов Ф2, но ТТЛ-уровня
SYNC	Вход для сигнала SYNC микропроцессора
STSTB	Выход для синхронизированного сигнала SYNC
-	
RESIN	Вход для асинхронного сигнала сброса
RESET	Выход для синхронизированного сигнала RESET микропроцессора
RDYIN	Вход для асинхронного сигнала готовности
READY	Выход для синхронизированного сигнала READY микропроцессора

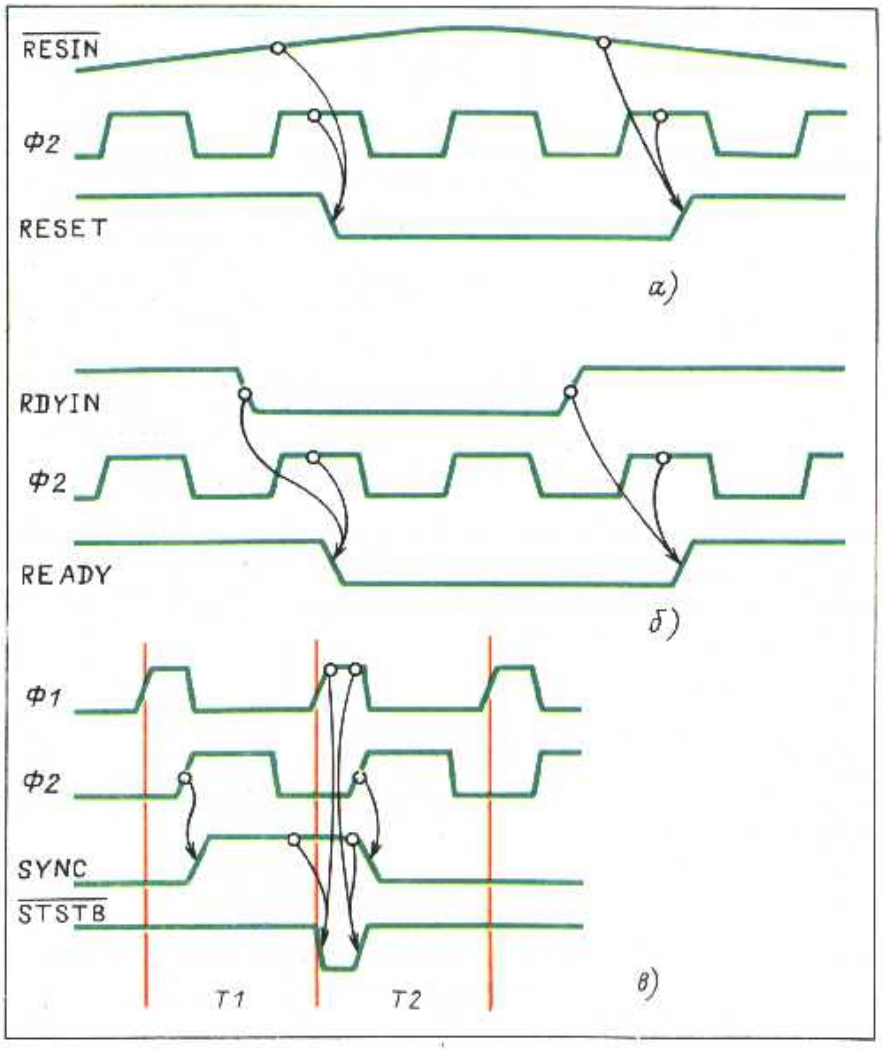


Рис. 6.12. Временные диаграммы синхронизации сигналов при помощи КР580ГФ24:
а - сигнал RESET; *б* - сигнал READY; *в* - сигнал SYNC

Опишем более подробно функции выводов тактового генератора. Выводы XTAL1 и XTAL2 предназначены для подключения кварцевого резонатора для стабилизации частоты генератора. При этом частота тактовых импульсов Ф1 иФ2 равняется 1/9 частоты кварцевого резонатора (так, в ПМ-ЭВМ при резонаторе на частоту 9 МГц частота тактовых импульсов равна 1 МГц). На выходах Ф1 и Ф2 вырабатываются тактовые импульсы, удовлетворяющие всем предъявляемым для микропроцессора КП580ИК80А требованиям. На выходе OSC вырабатывается синусоидальный сигнал с частотой, которую имеет кварцевый резонатор. Все оставшиеся выводы (т. е. кроме Ф1, Ф2 и OSC) работают с сигналами, имеющими ТТЛ-уровни. На выходе Ф2Т вырабатывается сигнал, идентичный Ф2, но имеющий ТТЛ-уровень, который может быть использован для синхронизации каких-либо устройств в микро-ЭВМ.

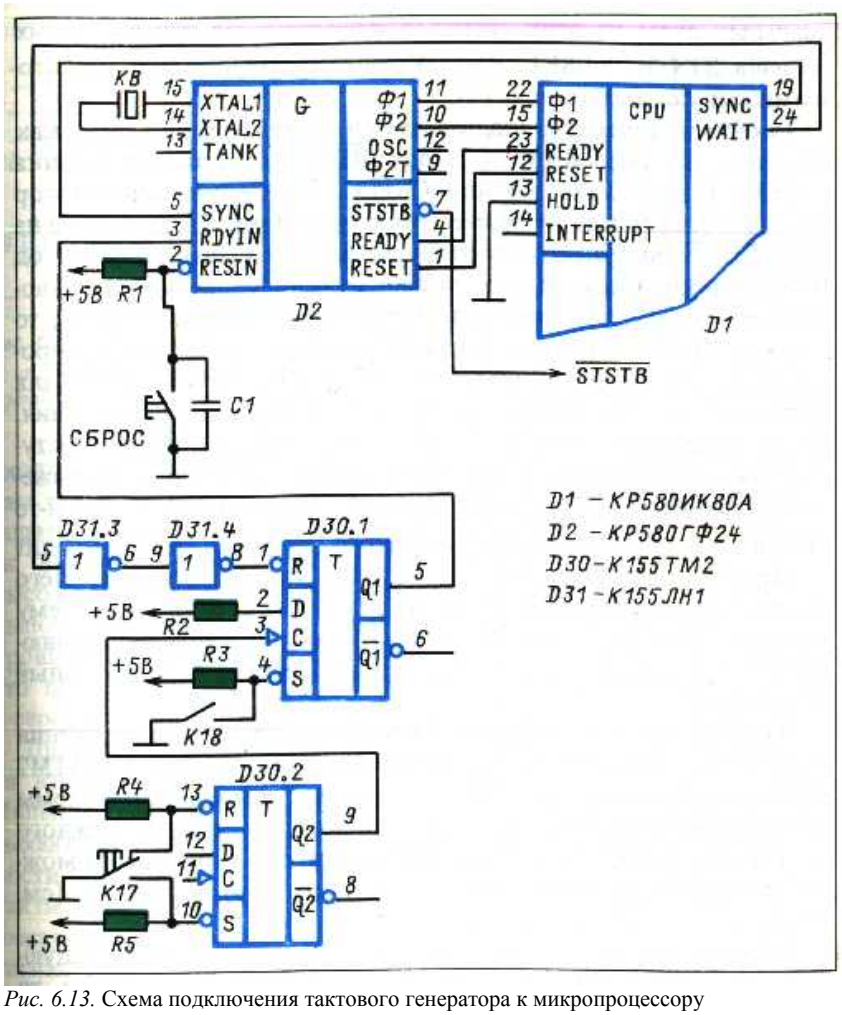


Рис. 6.13. Схема подключения тактового генератора к микропроцессору

Три входа SYNC, RESIN, RDYIN являются входами для асинхронных сигналов. Эти сигналы могут менять свой уровень в любой момент времени. Внутри КР580ГФ24 с помощью специальных схем из этих сигналов формируются три выходных сигнала STSTB, RESET, READY, синхронизированных с тактовыми импульсами (рис. 6.12).

Тактовый генератор подключается к микропроцессору, как показано на рис. 6.13. К входу RESIN подключена кнопка СБРОС. Параллельно кнопке СБРОС подключен конденсатор *С1,* Если эта кнопка не нажата, то конденсатор *С1* заряжен и на вход HESIN поступает высокий уровень. Следовательно, на вход RESET микропроцессора поступает низкий уровень, что позволяет ему нормально работать. Если нажать кнопку СБРОС, то на вход RESET поступит высокий уровень и работа микропроцессора будет прервана (см. § 6.3). Конденсатор *С1* служит для начального запуска микропроцессора после включения питания. После включения питания *С1* разряжен и на вход RESIN поступает низкий уровень. Затем конденсатор заряжается, напряжение на входе RESIN экспоненциально возрастает и в какой-то момент времени достигает высокого уровня. Это приводит к выработке сигнала RESET низкого уровня, разрешающего работу микропроцессора. Вход RESIN имеет специальную схему (триггер Шмидта), которая перерабатывает медленно меняющийся при заряде конденсатора входной сигнал в нормальный ТТЛ-фронт.

ПМ-ЭВМ можно снабдить схемой пошагового выполнения программы (рис. 6.13), построенной на микросхеме К155ТМ2 (D30) и двух инверторах (D31.3, D31.4). С помощью этой схемы выполнение программы приостанавливается в середине каждого машинного цикла. Состояния шины данных в этот момент можно наблюдать на индикаторах одного из портов вывода (см. §7.3).

Схема пошагового выполнения программы работает следующим образом. Если контакты выключателя *К18* замкнуты, то на вход *4* триггера D30.1 подается низкий уровень. На вход *1* также подается низкий уровень с выхода WAIT через буфер, состоящий из двух инверторов. В результате этого на выходе *5* триггера имеется высокий уровень, который, поступая на вход RDYIN, обеспечивает сигнал READY высокого уровня, что позволяет микропроцессору работать в нормальном режиме. Для перехода в режим пошагового выполнения программы необходимо разомкнуть контакты выключателя *К18.* Тогда на входе *4* триггера D30.1 будет высокий уровень, а на входе *1* — низкий уровень. Это приведет к тому, что на выходе *5* появится низкий уровень и микропроцессор перейдет в режим ожидания. После перехода в

режим ожидания на выходе WAIT и на входе *1* триггера установится высокий уровень. В этом состоянии схема может находиться неограниченно долго. Если теперь подать на вход *3* триггера D30.1 фронт, то на выходе *5* появится высокий уровень. Микропроцессор снимет сигнал WAIT высокого уровня, закончит начатый машинный цикл и начнет выполнять следующий машинный цикл. Сигнал WAIT низкого уровня вновь установит на выходе *5* триггера низкий уровень, что приведет к переходу микропроцессора в режим ожидания в середине следующего машинного цикла.

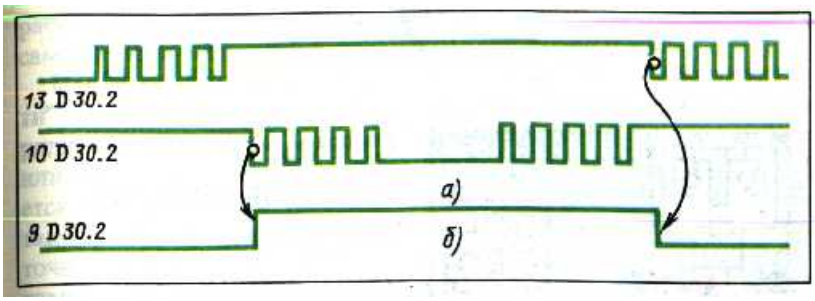


Рис. 6.14. Явление дребезга контактов:
а - реальная временная диаграмма, дребезг контактов; *б* - идеальная временная диаграмма

Для подачи импульсов на вход *3* триггера D30.1 используются кнопка *К17* и схема для подавления дребезга контактов. Явление дребезга контактов происходит при замыкании любых механических контактов вследствие неидеальности их поверхности. Вместо однократного замыкания или размыкания контакты замыкаются и размыкаются несколько раз. Так, при нажатии на кнопку *К17* на выводах *13, 10* микросхемы D30.2 сигнал имеет форму, изображенную на рис. 6.i4. Чтобы сформировать сигнал, имеющий один срез, используется кнопка с двумя контактами: замыкающимся и размыкающимся при нажатии на нее. Когда один из контактов кнопки, соединенный с входами *10* и *13* микросхемы D30.2, замыкается, то первый из возникающих импульсов перебрасывает триггер, а остальные импульсы уже не влияют на его состояние. Таким образом, на выходе 9 формируются фронт при нажатии на кнопку *К17* и срез при отпускании этой кнопки.

7
СХЕМЫ И ОСОБЕННОСТИ РАБОТЫ ОСНОВНЫХ БЛОКОВ ПМ-ЭВМ

7.1. ОБЩИЕ ПОЛОЖЕНИЯ

Выше была рассмотрена шинная архитектура, на основе которой строятся все современные микро-ЭВМ. Составные части или блоки микро-ЭВМ подключаются к шинам адреса, данных и управления. Эти шины служат для передачи информации от одного блока к другому, причем в определенный момент времени передатчиком может быть лишь один из блоков. В гл. 6 было рассмотрено устройство микропроцессорного блока, который хотя и является "сердцем" микро-ЭВМ, но не может самостоятельно работать без блока памяти и устройств ввода/вывода. Поэтому в данной главе рассматриваются функционирование и подключение к микропроцессорному блоку блока памяти (§ 7.2), устройств вывода — светодиодов и устройств ввода — клавиатуры (§ 7.3). Далее, в § 7.4, приводится полный текст программы-монитора, которая управляет работой микро-ЭВМ, в том случае, когда она не решает какую-либо задачу и позволяет пользователю с помощью кнопок клавиатуры и индикаторов вводить, отлаживать и запускать свои программы.

7.2. СТРУКТУРА ПАМЯТИ

Блок памяти состоит, как правило, из ОЗУ и ПЗУ. В ОЗУ и ПЗУ хранятся данные и программы пользователя. В ПЗУ помешаются те данные и программы, которые должны сохраняться при выключении питания. Во-первых, в ячейки ОЗУ информация может попадать, в результате работы какой-либо программы, во-вторых, пользователь может сам изменить их содержание при помощи клавиатуры.

Максимальное количество ячеек памяти (или "объем памяти"), к которым может обращаться при помощи своей 16-разрядной шины адреса микропроцессор КР580ИК80А, равно 2^{16}, или 65 536. Обычно говорят: не 65 536 ячеек, а 64 К (1 К равняется 1024). Для решения некоторых задач оказывается мало и такого объема памяти, но для большинства применений достаточно использовать лишь его часть. С другой стороны, объем памяти, содержащейся в одной микросхеме, может быть меньше, чем максимальный объем, который может адресовать

микропроцессор. Эти два обстоятельства, а также и то, что в составе памяти микро-ЭВМ необходимо иметь память с разными свойствами (ОЗУ и ПЗУ), приводят к специальным техническим решениям при конструировании памяти микро-ЭВМ.

Рассмотрим эти технические решения на примере блока памяти ПМ-ЭВМ. Блок памяти, состоящий из двух микросхем ОЗУ и двух ПЗУ, подключается к шинам данных, адреса и управления (рис. 7.1). В качестве ОЗУ в ПМ-ЭВМ используются микросхемы статической памяти КР541РУ2 (см. гл. 4) с организацией 1024х4 бита. Поэтому для получения объема памяти 1 Кбайт (1024 байта) необходимо подключить две микросхемы, присоединив выводы данных одной микросхемы (D12) к линиям DBO-DB3 шины данных, а выводы другой (D13) — к линиям DB4 — DB7. Остальные выводы микросхем D12 и D13 подключаются к одним и тем же линиям. Таким образом составляется блок ОЗУ емкостью 1 Кбайт.

В качестве ПЗУ в ПМ-ЭВМ используются две микросхемы КР556РТ4 (D14, D15) с организацией 256х4 бит. Они включены аналогично микросхемам ОЗУ, т. е. организованы в блок 25 6 байтов.

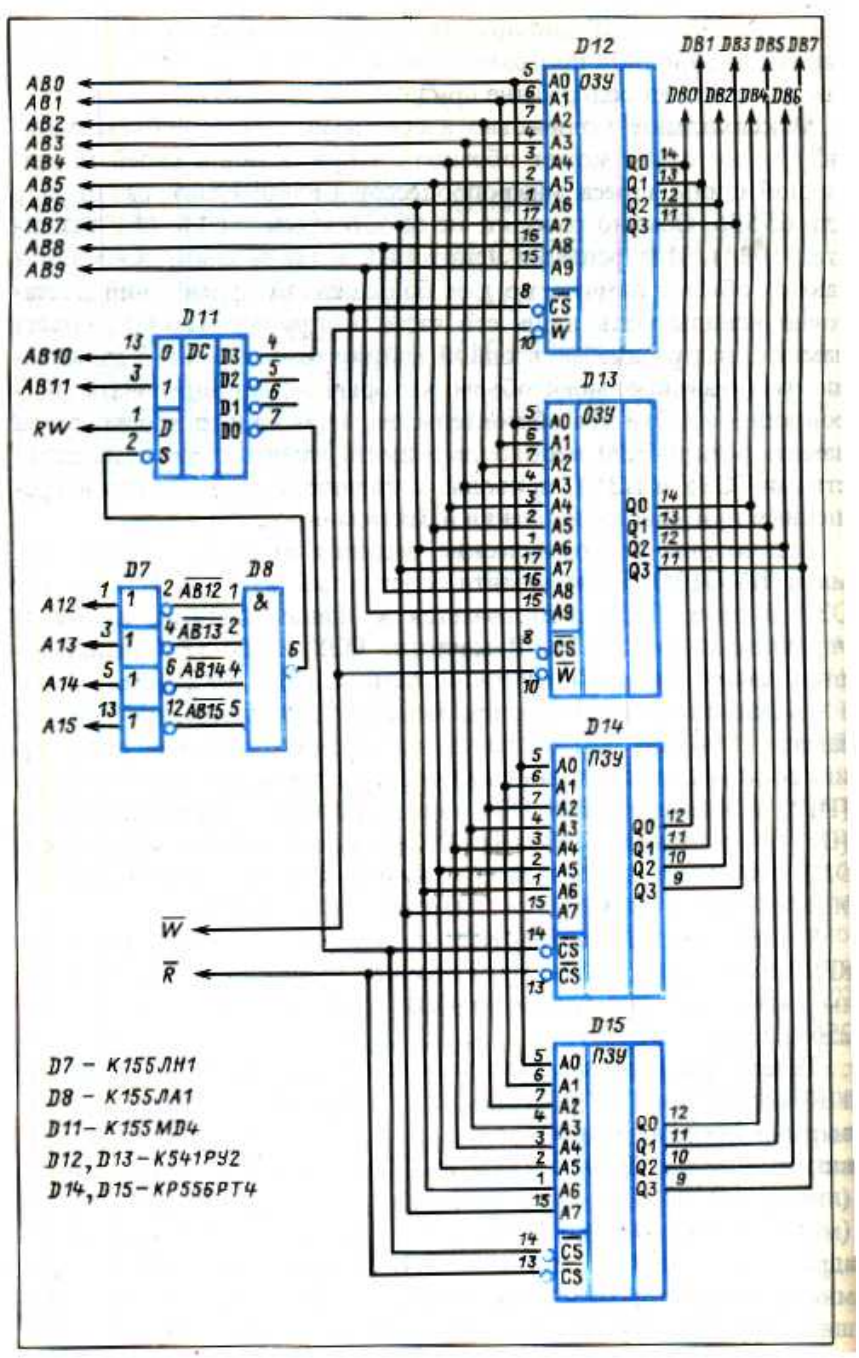


Рис. 7.1. Схема блока памяти

Теперь рассмотрим, как размещается блок ОЗУ объемом 1 Кбайт и блок ПЗУ объемом 256 байт в пространстве адресов, вырабатываемых микропроцессором. Для того чтобы осуществить привязку блоков ОЗУ и ПЗУ, старшие разряды адреса (линии AB10-AB15) подаются на схему дешифрации адреса (микросхемы D8,

D11, рис. 7.1). Шесть старших разрядов адреса АВ10-АВ15 определяют, к какому килобайту памяти микропроцессора будет произведено обращение. Десять младших разрядов адреса АВО-АВ9 адресуют байты памяти в пределах каждого килобайта. Каждому коду на АВ10-АВ15 соответствует один килобайт памяти, но вследствие того, что инвертированные линии АВ12-АВ15 шины адреса поступают на микросхему D8, сигнал с выхода которой разрешает или запрещает работу дешифратора D11, дешифратор D11 будет работать только при нулевых кодах на этих линиях. В зависимости от кода на линиях АВ10 — АВ15 низкий уровень будет появляться на одном из выходов 7, *6, 5* или *4* дешифратора D11 при условии, что на втором управляющем входе дешифратора (вывод *1)* будет высокий уровень. В табл. 7.1 дается соответствие кода на линиях АВ 10 — АВ 15 номеру вывода D11.

В табл. 7.1 приводятся адрес первого байта, адрес последнего байта и условный номер килобайта памяти, к которому происходит обращение при определенном коде на линиях АВ10 — АВ15. Из пятой строки табл. 7.1 также видно, что если адрес, вырабатываемый микропроцессором, лежит в диапазоне от 010000Q до 177 OOOQ, то ни на одном из выходов дешифратора не будет низкого уровня, т. е. с помощью выходов этого дешифратора нельзя адресоваться к килобайтам с 4-го по 63-й.

Выходы дешифратора используются для привязки отдельных блоков памяти емкостью 1 Кбайт к определенным областям в пространстве адресов микропроцессора; для этого они соединены со входами микросхемы памяти, разрешающими их работу, а на адресные входы микросхемы памяти подаются младшие разряды адреса, которые осуществляют адресацию ячеек памяти внутри микросхем.

Таблица 7.1

Код на линиях АВ10— АВ15						Номер вывода D11.на котором низкий уровень напряжения	Диапазон адресов области памяти в восьмеричном коде		Номер адресуемого ки-
A15	A14	A13	A12	All	A10		от адреса	до адреса	лобайта
0	0	0	0	0	0	7	000 000	001 777	0
0	0	0	0	0	1	6	002 000	003 777	1
0	0	0	0	1	0	5	004000	005 777	2
0	0	0	0	1	1	4	006 000	007 777	3
Любой другой код						-	010 000	177777	4-63

Рис. 7.1. Схема блока памяти

Блок из микросхем ПЗУ подключен так, что он занимает самые младшие адреса, начиная с 000 000Q. Это сделано для размещения программы-монитора, которая должна начинать работать сразу после включения микро-ЭВМ, или нажатия кнопки СБРОС, в результате чего вырабатывается сигнал RESET, по которому микропроцессор начинает выполнять команду, расположенную по адресу 000 OOOQ (т. е. в ПЗУ). Заметим,что для адресации 256 байтов ПЗУ достаточно восьми адресных линий АВО — АВ7; линии АВ8, АВ9 излишни и поэтому к ПЗУ не подсоединены. В результате этого независимо от кода на линиях АВ8 и АВ9 адресуется байт ПЗУ, определяемый адресом на линиях АВО — АВ7. Это приводит к тому, что один и тот же байт ПЗУ можно считывать по адресам, в разрядах АВ8, АВ9 которых находятся коды 00, 01, 10 или 11. Область памяти ПЗУ, состоящая из 256 байт, повторяется 4 раза в нулевом килобайте памяти. Ясно, что это повторение не мешает работе микропроцессора.

Такая схема адресации с неиспользованием части разрядов адреса называется схемой с неполной дешифрацией адреса. При необходимости использовать весь нулевой килобайт и поместить в пространство его адресов еще три блока памяти по 256 байт в каждом нужно использовать для разрядов АВ8 и АВ9 полный дешифратор, устроенный так же, как для разрядов АВ10 и АВ11. Схема с неполной дешифрацией адреса приводит к тому, что используется не все адресное пространство, но зато требуется меньше микросхем для ее реализации, и поэтому такая схема часто применяется в простых микро-ЭВМ.

Блок из микросхем ОЗУ подключен к выходу *4* дешифратора D11, и поэтому килобайт ОЗУ адресуется как 3-й килобайт в адресном пространстве микропроцессора. На рис. 7.2 приведена карта распределения памяти в ПМ-ЭВМ.

Этого объема памяти достаточно для нормальной работы, но его можно при необходимости увеличить. Объем используемого ОЗУ можно увеличить до 3 Кбайт, ничего не переделывая в схеме дешифрации адреса, а просто подключением дополнительных блоков ОЗУ емкостью по 1 Кбайт к выходам *6* и *5* дешифратора D11 (рис. 7.1). Можно также увеличить объем используемого ПЗУ, если построить его из микросхем большей емкости, например из микросхем К573РФ1 с организацией 1024х8 бит (1 Кбайт) или микросхем КР556РТ5 емкостью 4 Кбита и с организацией 512х8 бит (0,5 Кбайта). В первом случае будет использоваться весь 0-й килобайт, а во 2-м ячейки ПЗУ будут повторяться во второй половине 0-го килобайта.

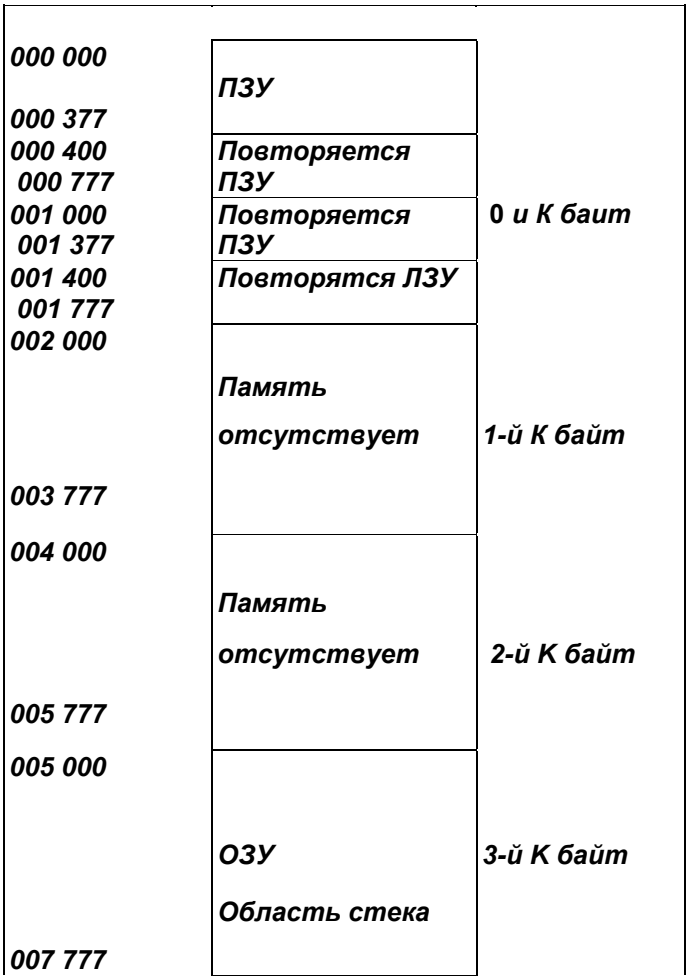


Рис. 7.2. Карта распределения памяти

Осталось рассмотреть подключение блока памяти к линиям шины управления. Линия R (рис. 7.1) подключена к входам *13* микросхем ПЗУ и разрешает их работу во время цикла считывания байта из ПЗУ. Линия W подключена к входам *10* микросхем ОЗУ и управляет записью байта в ОЗУ. Если на ней во время цикла обращения к ОЗУ поддерживается высокий уровень, то происходит считывание, а если низкий - то запись байта. Сигнал RW, вырабатывающийся и сигналом R, и сигналом W, служит Для того, чтобы разрешать работу дешифратора D11 только во время низкого уровня сигналов R и W. Это необходимо, чтобы считывание и запись в блок памяти происходили во время выра ботки сигналов R и W, которые формируются сигналами DBIN и WR микропроцессора, определяющими момент передачи информации по шине данных.

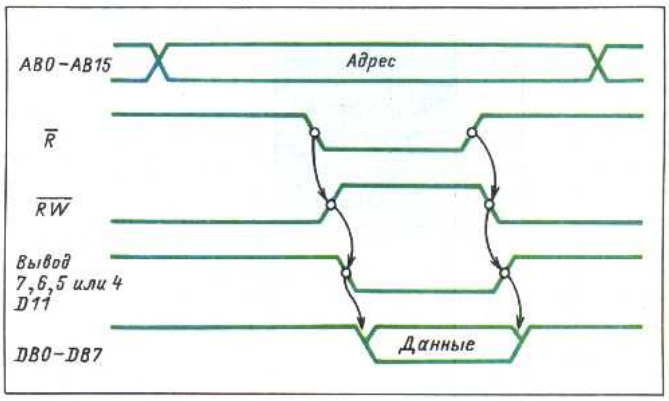


Рис. 7.3. Временная диаграмма работы блока памяти при чтении

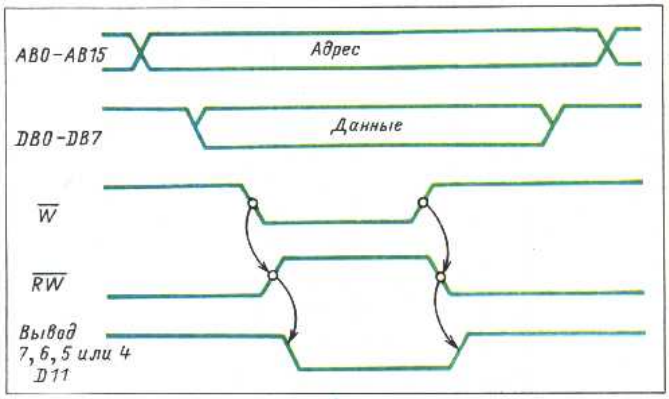


Рис. 7.4. Временная диаграмма работы блока памяти при записи

Считывание из ОЗУ микропроцессор может выполнить в циклах ВЫБОРКА КОМАНДУ, ЧТЕНИЕ ИЗ ПАМЯТИ, ЧТЕНИЕ ИЗ СТЕКА а считывание из ПЗУ - только в циклах ВЫБОРКА КОМАНДЫ И ЧТЕНИЕ ИЗ ПАМЯТИ. Временные диаграммы сигналов шины адреса, данных и управления приведены на рис. 7.3. Запись информации в ОЗУ микропроцессор может выполнить в циклах ЗАПИСЬ В ПАМЯТЬ, ЗАПИСЬ В СТЕК. Временные диаграммы записи приведены на рис. 7.4.

7.3. КЛАВИАТУРА И ИНДИКАЦИЯ

Устройства ввода/вывода необходимы для связи микро-ЭВМ с внешним миром. Без такой связи ее работа была бы бессмысленной. Устройства ввода/вывода бывают самые различные по конструкции и особенностям работы. Отличительной особенностью всех этих устройств является их способность перекодировать информацию из той формы, в которой она представляется в микро-ЭВМ, в форму, необходимую для связи с какими-либо приборами (для устройства вывода), и осуществлять обратный процесс для устройства ввода. Схема индикации и клавиатура микро-ЭВМ подключаются к линиям DBO-DJB7 шины данных, линиям ABO — AB7 шины адреса и линиям IN и OUT шины управления (рис. 7.5, 7.6). Схема индикации выполнена на базе трех байтовых (8-разрядных) портов вывода (микросхемы D16-D21), а схема клавиатуры — на базе одного 4-разрядного порта ввода (микросхемы D23).

Каждый из портов вывода схемы индикации выполнен из двух микросхем К155ТМ7. Эта микросхема представляет собой четыре D-триггера с прямыми и инверсными выходами (см. гл. 4). Входы триггеров подключены к линиям шины данных. Каждый выход D-триггера микросхемы К155ТМ7 имеет нагрузочную способность, равную 10, т. е. к нему можно подсоединить 10 стандартных ТТЛ-входов, а это значит, что при низком уровне на выходе (напряжение меньше или равно 0,4 В) в него может втекать ток до 16 мА. Это позволяет подключать свето-Диоды для индикации состояний триггеров непосредственно к их выходам (рис. 7.7). Сопротивление включается последовательно с светодиодом для того, чтобы ограничить ток до 5-10 мА. Для индикации можно использовать любые диоды с рабочим током 5 — 15 мА, например АЛ102А, АЛ102Г, АЛ112А-М. Светодиоды подключены к инверсным выходам триггеров, поэтому они зажигаются, если в соответствующий триггер записывается 1, и гаснут, если записывается 0.

В схему индикации и клавиатуры входит микросхема К155ИД4 (D24). Эта микросхема представляет собой два дешифратора-мультиплексора (см. гл. 5), объединенные в один дешифратор трехразрядного двоичного кода в код "один из восьми". Это достигается соединением входов *1* и 75, а также входов *2 и 14* микросхемы. На вход *13* подается первый разряд кода, на вход *3* - второй, а на объединенные входы 1и *15* -третий разряд двоичного кода, предназначенного для преобразования в код "один из восьми". Объединенные входы 2 и *14* служат для запрещения работы дешифратора. При этом на них должен быть подан высокий уровень. Такой режим работы микросхемы К155ТД4 представлен в табл. 7.2.

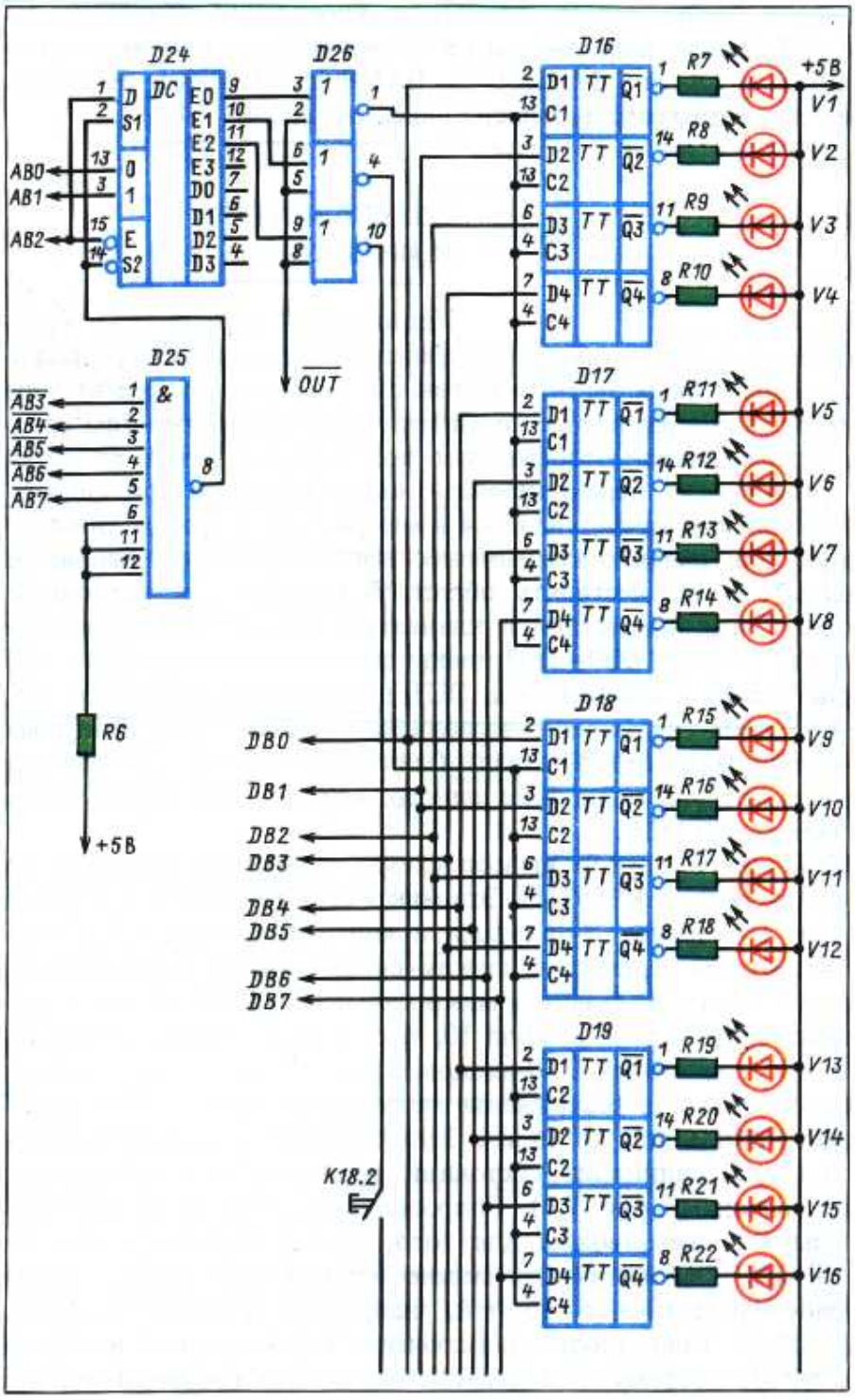


Рис. 7.5. Схема индикации

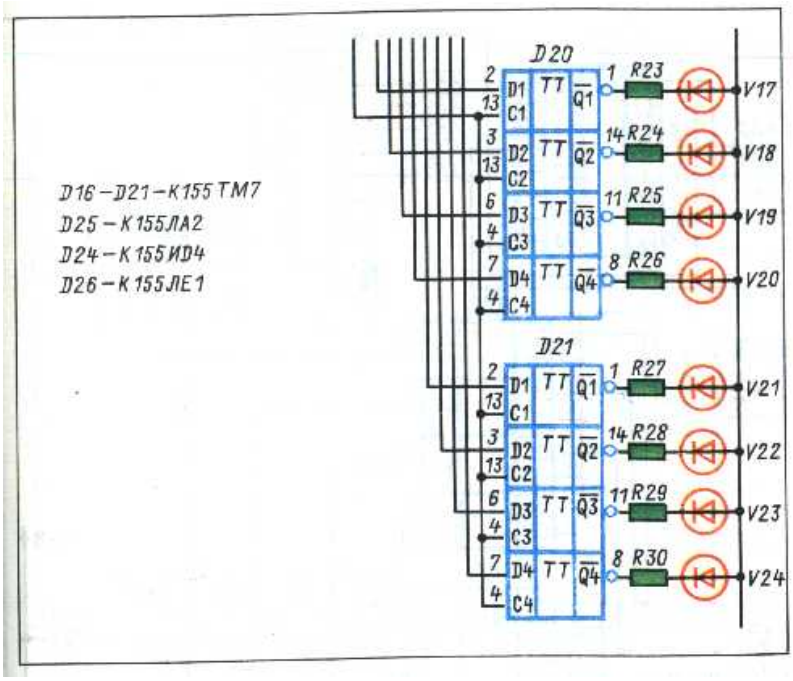


Рис. 7.5. Схема индикации (продолжение)

Микросхема К155ИД4 (D24) вместе с микросхемой К155ЛА2 (D25) составляет схему дешифрации адреса для выбора внешних устройств ввода и вывода. Эта схема функционирует следующим образом. Входы 13, 3 и 1/15 микросхемы D24 подсоединены к линиям ABO, AB1, AB2 шины адреса, а входы микросхемы D25 — подсоединены к инвертированным линиям шины адреса AB3, AB4, AB5, AB6, AB7. Поэтому когда во время выполнения команд IN или OUT на линиях шины адреса появляется адрес внешнего устройства (см. гл. 6), на одном из выходов микросхемы U24 появляется низкий уровень. Соответствие выполняемой команды IN или OUT тем выходам D24, на которых появляется низкий уровень, приведено в табл. 7.3.

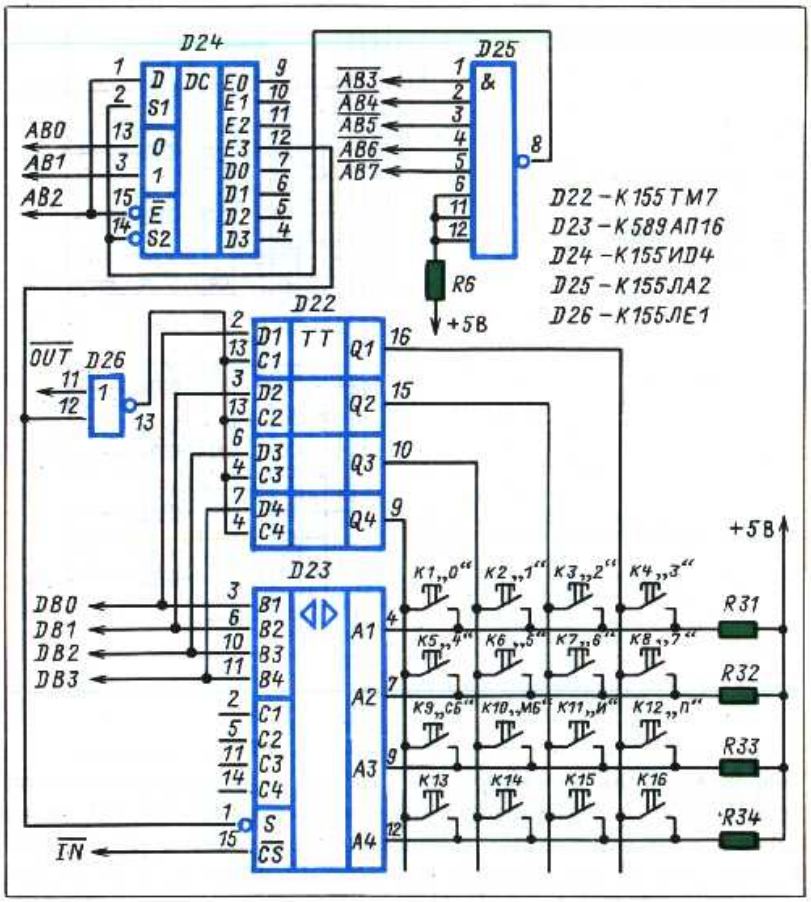


Рис. 7.6. Схема клавиатуры

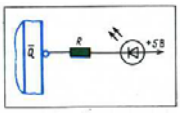


Рис. 7.7. Схема подключения све-тодиода к выходу порта вывода

Таблица 7.2

Входы							Выходы				
2/14	1/15	3	13	9	10	11	12	7	6	5	4
H	X	X	X	H	H	H	H	H	H	H	H
T	L	L	L	L	H	H	H	H	H	H	H
T	L	L	H	H	L	H	H	H	H	H	H
т	L	H	L	H	H	L	H	H	№	H	H
I,	L	H	H	H	H	H	L	H	H	H	H
т,	H	L	L	H	H	H	H	L	H	H	H
т,	H	L	H	H	H	H	H	H	L	H	H
т,	H	H	L	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	L

Таблица 7.3

Команда	Код на линиях шины адреса во время выполнения команды								Вывод микро-схемы D24 с низким уровнем
	AB7	AB6	AB5	AB4	AB3	AB2	AB1	ABO	
IN 000Q или OUT 000 Q	1	1	1	1	0	0	0	9	
IN 001 Q или OUT 001Q	1	1	1	1	0	0	1	10	
IN 002 Qили OUT 002 Q	1	1	1	1	0	1	0	11	
IN 003 Q или OUT 003 Q	1	1	1	1	0	1	1	12	
IN 004 Q или OUT 004 Q	1	1	1	1	1	0	0	7	
IN 005 Q или OUT 005 Q	1	1	1	1	1	0	1	6	
IN 00бQили OUT 006 Q	1	1	Г	1	1	1	0	5	
IN 007Qили OUT 00 7 Q	1	1	1	1	1	1	1	4	

Таблица 7.4

Команда	Микросхемы, составляющие порт ввода или вывода	Функция порта ввода или вывода
OUT 000 Q	D16, D17	Индикация одного байта
OUT 001 Q	D18, D19	То же
OUT 002 Q	D20, D21	То же

OUT003Q	**D22**	**Вывод четырех битов для обслуживания клавиатуры**
IN 003 Q	**D23**	**Ввод четырех битов для обслуживания клавиатуры**

Заметим, что если выполнить команду IN или OUT с отличным от перечисленных в табл. 7.3 адресом внешнего устройства, то ни на одном из выходов микросхемы D24 не появится низкого уровня. Из восьми возможных адресов для портов ввода и вывода в схеме индикации и клавиатуры используются три адреса 000 Q, 001Q и 002 Q, которые присвоены портам вывода, служащим для индикации, и один адрес 003 Q, который присвоен одновременно 4-битовым портам ввода и вывода, обслуживающим клавиатуру (табл. 7.4).

Остальные выходы дешифратора D24 можно использовать для подключения дополнительных устройств ввода/вывода. Всего при такой схеме дешифрации адреса внешнего устройства можно подключить восемь устройств ввода и восемь вывода. Для того чтобы еще увеличить число устройств ввода/вывода, схему дешифрации необходимо изменить и подвергать дешифрации большее число младших разрядов адреса.

При выпрлнении одной из команд OUT, перечисленных в табл. 7.4, во время ее третьего цикла на шинах данных, адреса и управления вырабатываются сигналы, изображенные на рис. 7.8.

При этом вырабатывается сигнал на соответствующем выходе дешифратора D24 (см. табл. 7.3). Сигнал с выхода дешифратора подается вместе с сигналом OUT на входы микросхемы К155ЛЕ1 (D26) и формирует на ее выходе положительный импульс. Этот импульс, поступая на управляющие входы *4, 13* D-триггеров одного из портов, своим фронтом (переход из 0 в 1) переписывает код с шины данных в триггеры, а своим срезом фиксирует его там. Таким образом, 8 бит кода, содержащегося в аккумуляторе перед выполнением команды OUT, высвечиваются на индикаторах порта 000Q, 001Q или 002Q. Что касается порта с адресом 003 Q, то в него командой OUT 003 Q можно переписать только четыре младших бита аккумулятора, так как порт состоит только из четырех триггеров.

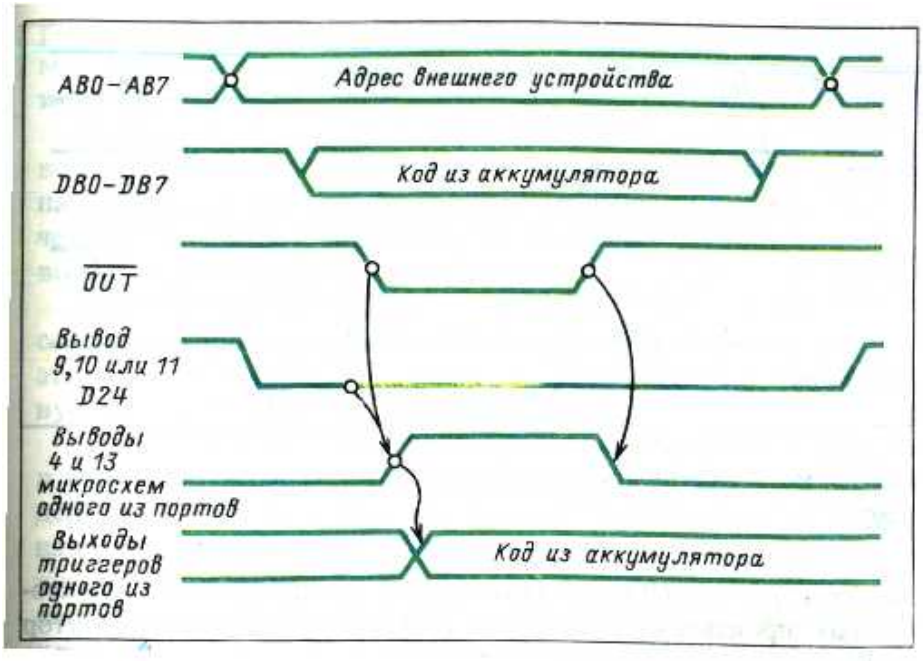


Рис. 7.8. Временная диаграмма работы порта вывода

Этот порт совместно с портом ввода (с тем же адресом 003Q), реализованным на микросхеме К589АШ6 (D23), предназначен для подключения к микро-ЭВМ клавиатуры (см. рис. 7.6). Кнопки клавиатуры включены между выходами микросхемы D22 и входами D23. Входы D23 через сопротивление 5 кОм соединены с напряжением питания. Это сделано для того, чтобы поддерживать на каждом входе D23 высокий уровень, если ни одна кнопка, связанная с этим входом, не нажата. Значения сопротивлений (5 кОм) выбраны с тем расчетом, чтобы не перегружать выходы D22, если какая-нибудь кнопка нажата.

Схема клавиатуры работает следующим образом. Специальная программа SKL (см. § 7.4) записывает в порт с адресом 003 Q такие коды, у которых в одном из четырех разрядов О, а в остальных разрядах 1,

причем 0 появляется последовательно в разрядах О, 1, 2 и 3 и соответственно на выходах *16,15, 10* и 9 микросхемы D22. После того как код записан и на одном из выходов *16, 15, 10* или 9 установился низкий уровень, эта программа производит командой IN 003 Q ввод в аккумулятор кода, который в этот момент присутствует на входах *4,* 7, 9 и *12* микросхемы D23. Ввод в аккумулятор происходит во время третьего цикла команды IN (рис. 7.9). Адрес внешнего устройства 003 Q вызывает появление низкого уровня на выходе *12* микросхемы D24. Этот низкий уровень, попадая на вход *1* микросхемы D23, разрешает ее работу, но ее выходы *3, 6, 10* и *13* остаются в состоянии высокого сопротивления. По сигналу IN эти выходы открываются и в четыре младших разряда аккумулятора микропроцессора переписываются состояния входов *4, 7, 9* и *12*. Если ни одна из кнопок, соединенных с тем выходом D22, на котором присутствует низкий уровень, не нажата, то все четыре младших разряда аккумулятора равны 1, если одна из кнопок нажата, то соответствующий разряд будет равен 0. По тому коду, который был записан в порт с адресом 003 Q, и тому коду, который был считан из порта 003 Q, можно определить, какая из кнопок нажата. Как это делается, описано в следующем параграфе.

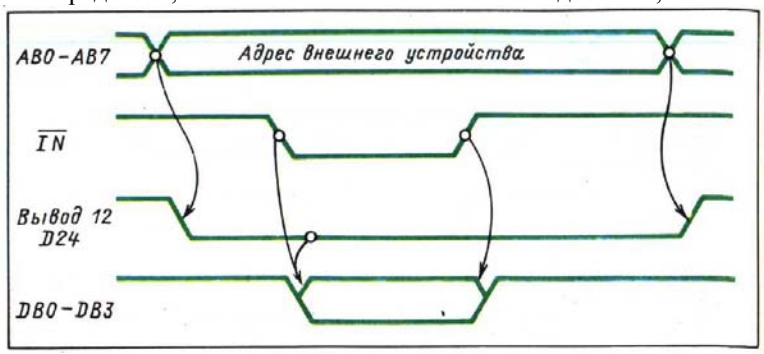


Рис. 7.9. Временная диаграмма работы порта ввода

7.4. ПРОГРАММА-МОНИТОР

Программа-монитор (или просто монитор) — это программа, которая управляет работой микро-ЭВМ. Монитор выполняет следующие функции: подготовку к работе устройств микро-ЭВМ после включения питания и повторных перезапусков, обслуживание индикаторов и клавиатуры, выполнение указаний пользователя по реализации программы. Микро-ЭВМ выполняет программу-монитор все то время, когда она не выполняет какую-либо программу пользователя.

Ниже приведен полный текст программы-монитор. При первом прочтении можно пропустить дальнейший текст данного параграфа, но при отладке ПМ-ЭВМ и работе с нею необходимо четкое понимание функционирования монитора, для чего и приводится ее детальное описание.

Текст программы состоит из трех столбцов. Первый столбец содержит адреса ячеек памяти, второй столбец - содержимое этих ячеек в восьмеричном коде, а третий столбец — ассемблерную запись программы.

После подачи всех необходимых напряжений на микросхемы и микропроцессор микро-ЭВМ начинает работать. При этом содержимое всех регистров микропроцессора и ячеек ОЗУ устанавливается случайным образом. Поэтому в микро-ЭВМ происходит в этот момент неуправляемый и непредсказуемый процесс. Для того чтобы остановить его и пустить по вполне определенному руслу, в микропроцессоре КР580ИК80А имеется вход RESET (вывод 72). На этот вход подается высокий уровень, который вырабатывается при помощи специальной схемы сразу после включения питания (см. гл. 6) или при нажатии на кнопку СБРОС. По этому сигналу микропроцессор заносит в счетчик команд (PC) во все разряды нули. Это означает, что следующая команда будет считываться из ячеек с нулевым адресом. По этому адресу размещается первая команда монитора. Для того чтобы монитор уже находился в памяти микро-ЭВМ после ее включения, он помещается в ПЗУ. Кроме того что монитор занимает память ПЗУ, он, как и всякая программа, при своей работе использует ячейки ОЗУ и регистры микропроцессора. Какие ячейки ОЗУ и как он использует, будет ясно после разбора самой программы.

Итак, первая команда монитора, расположенная по адресу 000000Q, - команда безусловного перехода JMP Ml, которая передает управление команде, расположенной по адресу 000070Q. Команда JMP Ml занимает три байта. Байты с 000 003 Q по 000 067 Q не используются. Эту область ПЗУ следует оставить незапрограммированной, так как она может понадобиться для расширения возможностей микро-ЭВМ по работе с прерываниями. Под меткой Ml (по адресу 000070Q) находится команда LXI SP, 010000Q. Эта команда загружает константу 010000Q в указатель стека (SP), чтобы он указывал на первую несуществующую ячейку ОЗУ. Тогда при первом занесении данных в стек (ячейки которого располагаются в ОЗУ) SP будет увеличен на единицу и будет указывать последнюю ячейку физически существующего ОЗУ, куда и поместятся данные. При дальнейших обращениях стек будет "расти" от старших адресов памяти к младшим при записи в него и "уменьшаться" при считывании. Если пользователь далее будет правильно использовать команды записи и считывания (их число должно быть одинаковым), никогда не произойдет ошибки обращения к стеку. Обратим здесь внимание на то, что восьмеричная константа, записанная в ассемблерной строке под меткой Ml, приобрела другую кодировку при записи в память в ячейки 000071Q

и 000 072 Q. Чтобы понять, что произошло при переходе от записи на ассемблере к реальному расположению константы в памяти, запишем сначала эту константу в двоичном коде: 010000Q равняется 0001000000000000B. Если теперь разбить это 16-разрядное двоичное число на два байта, то получится 00010000B и 00000000B, или в восьмеричной системе 020Q и OOOQ. Эти числа и записаны в ячейки с адресами 000 072 Q и 000071Q соответственно. Этот перевод константы от той формы, в которой она записана на ассемблере, к реальному расположению в памяти будет встречаться еще во многих командах монитора.

Следующая команда LXI H, 006 000 Q загружает пару регистров H-L константой 006 000Q. Эта константа является адресом первой ячейки ОЗУ. Загрузка SP и H-L начальными значениями выполняется только один раз при входе в монитор (после сигнала RESET). В дальнейшем эта пара регистров будет использоваться для хранения адреса ячейки памяти, с которой в данный момент работает программист.

Итак, SP установлен таким образом, что поле стека будет находиться в старших адресах ОЗУ, а в H-L находится адрес первой ячейки ОЗУ. Следующие семь команд, занимающие ячейки с 000076Q по 000110Q, служат для того, чтобы высветить на индикаторах адрес ячейки памяти, с которой в настоящее время работает программист, и ее содержимое. Команда M2: MOV C, M (адрес 000 076 Q) переписывает данные из ячейки, адрес которой хранится в H-L (при первом проходе данного места программы это адрес первой ячейки ОЗУ), в регистр С. Далее команда MOV A, H переписывает данные из регистра H в аккумулятор, чтобы с помощью следующей команды OUT 001Q вывести ее на индикаторы с адресом 001Q. Данные переписываются в аккумулятор перед их выводом этой командой (в двух последующих аналогичных случаях применяется команда OUT OOOQ и OUT 002Q), потому что команда OUT может переслать в устройство вывода только содержимое аккумулятора. После выполнения команды OUT 001 Q на соответствующих индикаторах высветится содержимое регистра H. Каждый светодиод, которому соответствует единица в каком-либо разряде регистра H, загорится, а каждый светодиод, которому соответствует нуль в каком-либо разря-пе H погаснет. Аналогично на светодиодах порта OOOQ высветится содержимое регистра L (команды MOV A, L и OUT OOOQ) и порта 002 Q - регистра С (команды M3: MOV A, C и OU 1 00?Q) После этого выполняется команда M4: CALL SKL, которая вызывает подпрограмму SKL, обслуживающую клавиатуру. Прежде чем переходить к описанию этой подпрограммы, опишем еще одну подпрограмму DL, которая вызывается из подпро-граммы SKL.

Подпрограмма DL служит для задержки выполнения программы на 10 мс. Такие задержки часто необходимы при работе с внешними устройствами, быстродействие которых значительно меньше, чем у микро-ЭВМ. Поэтому эта подпрограмма, находящаяся и используемая в мониторе, написана таким образом что сохраняет значение всех регистров и ячеек памяти во время своей работы. Она может быть вызвана из других программ пользователя. Подпрограмма DL начинается с ячейки 000 277 Q Команды PUSH PSW и PUSH D переписывают в стек содержимое аккумулятора, регистра состояний и пары регистров D-E для того, чтобы освободить эти регистры и использовать далее для своей работы, а перед выходом из подпрограммы восстановить их старое содержание. Затем пара регистров D-E загружается с помощью команды LXI D, 001 016Q константой 001016Q. Следующие пять команд образуют цикл. Команда N:DCX D уменьшает содержимое пары регистров D-E, вычитая из хранящегося в них двоичного числа единицу. Затем старший байт пары D-E командой MOV A, D переписывается в аккумулятор Далее команда ORA E выполняет поразрядно логическую операцию ИЛИ над содержимым аккумулятора и регистра E и результат этой операции записывается в аккумулятор, при этом если содержимое A и E (или D и E) не равно нулю, сбрасывается флаг признака нуля результата Z. Это приводит к тому что следующая команда JNZ N (адрес 000 307 Q) осуществляет переход к метке N (адрес 000304Q) и фрагмент программы от метки N до команды JNZ N (или, что то же самое, от адреса 000304Q до 000311Q) будет повторяться до тех пор пока в обоих регистрах D и E не образуются нули. Следовательно этот фрагмент программы выполняется 001016Q раз. Обозначим буквой *N c* соответствующим индексом число машинных тактов определенной команды. Тогда число тактов при однократном выполнении этого фрагмента N_0 $^{-}N_{DCXD}$ + + $N_{MOVA,D}+N_{ORAE}+N_{JNZN} = 5 + 5 + 4+10= 19$.

Так как длительность одного такта при использовании кварцевого резонатора на частоту 9 МГц равняется 1 мкс (см. гл. 6), время выполнения этого фрагмента 19 мкс, а чтобы выполнить его 001 016Q раз, нужно 9994 мкс. Следовательно, выполнение этой программы задерживает выполнение программы, вызвавшей ее, примерно на 10 мс. После того как выполнено 001 016 Q циклов и содержимое D и E равно нулю, флаг Z не устанавливается командой OKA E и командой JNZ N не передает управление к метке N. Тогда выполняются команды POP D и POP PSW, которые восстанавливают содержимое пары регистров D-E, аккумулятора и слова состояний. Затем команда RET передает управление команде, следующей за той командой CALL DL, которая вызвала эту подпрограмму.

Рассмотрим теперь подпрограмму SKL. Эта подпрограмма обеспечивает работу клавиатуры микро-ЭВМ. Она располагается в области ПЗУ с адресами 000 177Q - 000 276Q. При ее вызове командой CALL SKL управление передается команде, располагаемой по адресу 000177Q. Команда SKL: MVI A, 000Q загружает в аккумулятор OOOQ, а следующая команда OUT 003Q записывает содержимое младших четырех битов аккумулятора, т. е. 0000B, в четыре триггера микросхем К155ТМ7 (см. микросхему D22 на рис. 7.7).

Следовательно, на выходах триггеров этой микросхемы (выводы *16, 15, 10,* 9) устанавливаются нули. При этом если нажать хотя бы одну из кнопок К1 — К16, то на соответствующем входе микросхемы К589АП16 (D23) установится также нуль. Далее следующая команда М5: IN 003Q переписывает в четыре младших бита аккумулятора те логические состояния, которые в момент выполнения этой команды присутствуют на входах микросхемы D23. Затем команда ANI 017Q путем выполнения логической операции И записывает в старшие четыре разряда аккумулятора нули, а младшие четыре разряда оставляет такими, какие они были. После этого команда CPI 017Q сравнивает содержимое аккумулятора с константой 017Q (00001111B). В результате этого сравнения в регистре состояний процессора будет установлен флаг нулевого результата, если при выполнении команды М5: IN 003Q на входах микросхемы D23 (выводы *4, 7, 9, 12)* были единицы, т. е. ни одна кнопка *К1-К16* не была нажата. Если хотя бы одна из кнопок была нажата, то флаг нулевого результата после сравнения с 017 Q не будет установлен и следующая команда условного перехода JNZ М5 вызовет переход к метке М5. Таким образом, фрагмент программы, начиная с метки М5 и кончая командой JNZ М5, будет выполняться до тех пор, пока все кнопки не будут отпущены. Этот фрагмент программы нужен потому, что после нажатия кнопки монитор выполняет все необходимые операции столь быстро, что пользователь еще не успевает отпустить кнопку, а программа уже готова обработать следующее нажатие кнопки. Поэтому если бы не было этого фрагмента, одно нажатие кнопки воспринималось бы как несколько нажатий, что приводило бы к неправильной работе монитора.

Прежде чем рассматривать дальше работу подпрограммы, обратим внимание на тот факт, что при нажатии кнопок *К1-К16* происходит дребезг контактов, рассмотренный в гл. 6. Там же приводилась специальная схема, позволяющая устранить это явление. Дребезг контактов (см. рис. 6.14) приводит к тому, что на входе D23 вместо идеального перехода из нуля в единицу при размыкании контактов кнопки и из единицы в нуль при замыкании контактов кнопки имеется серия переходов. Каждый такой переход может быть воспринят как новое нажатие на кнопку, поэтому необходимо принять специальные меры, чтобы этого не произошло. Можно было бы снабдить каждую кнопку электронной схемой для подавления дребезга контактов (см. рис. 6.13), но можно воспользоваться программным методом, описываемым ниже. В подпрограмме для этого после команды JNZ М5 идет команда CALL DL, которая вызывает подпрограмму временной задержки. Выполнение программы задерживается на 10 мс. За это время дребезг контактов заканчивается.

Следующий фрагмент программы "ожидает" нажатия кнопки и вводит код нажатой кнопки в аккумулятор. В качестве рабочих регистров будут использоваться пара D-E и аккумулятор, поэтому командой PUSH D содержимое пары регистров D-E сохраняется в стеке. Следующие две команды М8: MVI D, 003Q и MVI E, 376Q загружают в регистры D и E коды 003 Q и 376Q соответственно. Затем команда М7: MOV A, E переписывает содержимое регистра E в аккумулятор, а команда OUT 003Q переписывает четыре младших разряда аккумулятора в триггеры микросхемы D22. После этого команда RLC сдвигает содержимое аккумулятора на один бит влево, а команда MOV E, A переписывает его в регистр E. Содержимое регистров и состояние выходов триггеров микросхемы D22 в двоичном коде в этот момент представлено в первой строке табл. 7.5.

Та б л и ц а 7.5

Номер строки, номер проход.	**Содержимое регистров**				**Состояние выходов триггеров D22**			
	D	**E**			**9**	**10**	**15**	**16**
1	**00 000 01 1**	**11**	**111**	**101**	**1**	**1**	**1**	**0**
2	**00 000 010**	**11**	**111**	**011**	**1**	**1**	**0**	**1**
3	**00 000 001**	**11**	**110**	**111**	**1**	**0**	**1**	**1**
4	**00 000 000**	**11**	**101**	**111**	**0**	**1**	**1**	**1**

Далее команда IN 003Q вводит в аккумулятор состояния входов микросхемы D23, команда ANI 017Q устанавливает в старших четырех разрядах аккумулятора нули, а команда CPI 017Q сравнивает его содержимое с константой 017Q и устанавливает флаг нуля результата, если содержимое равно 017Q. Это произойдет в том случае, если ни одна из кнопок *К4, К8, К12, К16* не нажата. Рассмотрим этот вариант. Тогда следующая команда JNZ Мб не осуществляет условный переход (так как установлен флаг нуля) и выполняется команда OCR D, которая уменьшает содержимое регистра D на единицу. Затем содержимое регистра D пересылается в аккумулятор командой MOV A, D и сравнивается командой CPI 377Q с константой 377 Q, и поскольку равенства нет (так как в регистре D код 002Q), команда JNZ М7 осуществляет переход к команде М7: MOV A, E и фрагмент программы от М7: MOV A, E до JNZ М7 повторяется. Этот фрагмент будет выполняться 4 раза (при условии, что ни одна кнопка не нажата). Состояния выходов триггеров микросхемы D22 и содержимое регистров D и E после выполнения команды MOV E, A (адрес 000230Q) указаны в табл. 7.5 для каждого прохода.

Во время четвертого прохода содержимое регистра станет равным 377Q и команда JNZ M7 (адрес 000 246Q) не осуществит переход к M7, а будет выполняться следующая за ней -команда безусловного перехода JMP M8, в результате чего команды M8: MVI D, 003Q и MVI E, 376Q загрузят регистры D и E и фрагмент от M7: MOV A, E до JNZ M7 повторится опять 4 раза. Этот процесс будет повторяться до тех пор, пока не нажата ни одна кнопка и на входах *4, 7, 9, 12* микросхемы D23 находятся высокие уровни, а следовательно, в аккумуляторе после выполнения команды ANI 017Q (адрес 000 233 Q) находится код 017Q. Если какая-либо кнопка нажата, то код в аккумуляторе не равен 017Q и команда JNZ Мб осуществляет переход к команде Мб: CALL DL, которая вызывает подпрограмму временной задержки для того, чтобы переждать дребезг контактов после нажатия кнопки. Значения кода в аккумуляторе после выполнения команды ANI 017Q в зависимости от кода на выходах триггеров микросхемы D22 и от того, какая нажата кнопка, приведены в табл. 7.6.

Таблица 7.6

Номер строки	Выход триггера микро-Код в ре-схемы гистре D В22,на котором низкий уровень	Нажатая кнопка	Вход мик-росхемы D23, на котором низкий уровень	Код в аккуму-ляторе
1	9 00 000 000	К1	4	00001 ПО
		К5	7	00001 101
		К9	9	00001 011
		К13	12	00000 111
2	10 00000001	К2	4	00001 ПО
		Кб	7	00 001 101
		К10	9	00001 011
		К14	12	00 000 1 1 1
3	15 00000010	КЗ	4	00001 110
		К7	7	00 001 101
		К11	9	00001 011
		К15	12	00 000 1 1 1
4	16 00000011	К4	4	00001 110
		К8	7	00 001 101
		К12	9	00001 011
		К16	12	00000 111

После выполнения подпрограммы DL команда M10: RRC сдвигает содержимое аккумулятора на один бит вправо, а самый младший бит при этом попадает в флаг переноса. Следующая команда JNC M9 осуществляет условный переход, если флаг переноса не установлен (перенос С раве« нулю). Рассмотрим вариант, когда перенос не равен нулю (строки 2 — 4 табл. 7.5) и перехода к M9 не происходит. Тогда следующая команда PUSH PSW сохраняет содержимое аккумулятора и слово состояний микропроцессора в стеке для того, чтобы можно было дальше использовать аккумулятор. Команда MOV A, D переписывает содержимое регистра D в аккумулятор, а команда ADI 004Q прибавляет к содержимому аккумулятора константу 004Q. После этого команда MOV D, A переписывает результат сложения в регистр D, а команда POP PSW восстанавливает содержимое аккумулятора и слово состояний микропроцессора, считывая их из стека. Далее команда безусловного перехода JMP M1C осуществляет переход к команде M10: RRC, т. е. повторяет фрагмент программы, начиная с M10. Теперь ясно, что в зависимости от кода в аккумуляторе (табл. 7.6) фрагмент программы от команды JNC M9 (адрес 000 260Q) до командь: JMP M10 (адрес 000 271Q) будет выполняться нуль раз для первой строки таблицы, 1 раз — для второй. 2 раза — для третьей и 3 раза — для четвертой. Это значит, что к коду в регистре (см, табл. 7.6} соответствующее число раз прибавится константа 004Q. Следовательно, в регистре D после выполнения команды JNC M9 будет содержаться код, зависящий от нажатой кнопки. Все возможные коды перечислены в табл. 7.7

Таблица 7.7

Нажатая кнопка	Код в регистре D в двоичной системе	Код в регистре D в восьмеричной системе	Мнемоничес кое название кнопки
К1	00 000 000	000	0
К2	00 000 001	001	*i*

К3	**00000010**	**002**	***2***
К4	**00000011**	**003**	**3**
К5	**00000 100**	**004**	**4**
Кб	**00 000 101**	**005**	**5**
К7	**00 000 110**	**006**	**6**
К8	**00 000 111**	**007**	**7**
К9	**00 001 000**	**010**	**СБ**
К10	**00 001 001**	**011**	**МБ**
К11	**00001 010**	**012**	**К**
К 12	**00001 011**	**013**	**П**
К13	**00 001 100**	**014**	**-**
К14	**00001 101**	**015**	**=**
К15	**00001 110**	**016**	**=**
К16	**00 001 111**	**ОП**	**=**

После перехода в результате выполнения команды JNC M^c-выполняется команда М9: MOV A, D, которая переписывав-: содержимое регистра D в аккумулятор. Затем командой РОР Г восстанавливается то содержание регистра D, которое было до начала работы подпрограммы SKL. На этом работа подпрограммы заканчивается и команда RET загружает в счетчик команд адрес команды, следующей за той, которая вызвала переход к подпрограмме. Заметим, что подпрограммы SKL и DL не портят при своей работе содержимое каких-либо регистров. Результат своей работы — код, соответствующий нажатой кнопке, — подпрограмма SKL возвращает в аккумулятор. Итак, была нажата кнопка, подпрограмма SKL выработала код, соответствующий этой кнопке, и поместила его в аккумулятор. Работа монитора продолжается с команды CPI 010Q (адрес 000113Q). Эта команда сравнивает содержимое аккумулятора с константой 010Q. Это сравнение происходит путем вычитания из кода, хранящегося в аккумуляторе, кода 010Q (по правилам двоичного вычитания), хотя содержимое аккумулятора не портится. При этом если значение двоичного числа, которое выражено кодом, содержащимся в аккумуляторе, меньше 010Q (или 8D), то по правилам двоичного вычитания происходит заем и устанавливается флаг переноса С, если больше или равно 010Q - то флаг не устанавливается. Затем команда JNCM11 осуществляет переход к метке МП (адрес 000134Q), если код нажатой кнопки больше или равен 010Q, если нет, то выполняются команды, следующие за JNC МП. Коды, меньшие 010Q, соответствуют кнопкам *К1-К8* и обрабатываются МОНИТОРОМ ПО-ОСОбому, потому что эти кнопки кодируют восьмеричные цифры от 0 до 7 (см. табл. 7.7) для ввода в микро-ЭВМ. Остальным кнопкам присвоены специальные функции, которые выполняются монитором (см. ниже).

Рассмотрим, как происходит ввод восьмеричного числа в микро-ЭВМ. Напомним, что регистр С используется для временного хранения введенного числа до того момента, как онс будет переписано в память по адресу, хранящемуся в паре регистров H-L. После того как была нажата одна из кнопок *К1-К8,* программа переходит к команде MOV B, A (адрес 000 120Q), которая переписывает код нажатой кнопки из аккумулятора в регистр В. Затем код, содержащийся в регистре С (этот код был высвечен на индикаторах порта OOOQ), переписывается в аккумулятор командой MOV A, C и сдвигается тремя командами RAL на три бита влево. После этого с помощью команды ANI 3 70Q очищаются три младших бита, а с помощью команды ORA B на эти места записываются новые трк разряда кода нажатой кнопки. В результате в аккумуляторе готов новый код для индикации на индикаторах порта OOOQ, Далее содержимое аккумулятора переписывается в регистр С командой MOV C_5 A, а затем команда JMP М3 осуществляет переход к фрагменту программы, реализующему индикацию.

На индикаторе порта OOOQ пользователь видит следующее. Если до нажатия кнопки состояния индикаторов были XO — X7 (рис. 7.10,д), то после нажатия эти состояния сдвигаются на три индикатора влево (причем X5 — XV пропадают), а на месте XO, XI, X2 высвечивается код вновь нажатой кнопки (рис. 7.10,6, табл. 7.8).

Рис. 7.10. Состояния индикаторов порта вывода с адресом OOOQ:
а - до нажатия какой-либо из кнопок 0-7; *б* - после нажатия какой-либо из кнопок 0 — 7

Следовательно, нажав три необходимые кнопки из кнопок *К1-К8,* пользователь может набрать на восьми индикаторах порта OOOQ любое нужное ему двоичное число длиной 1 байт.

Рассмотрим теперь, какие специальные функции и как выполняются монитором.

Таблица 7.8

Кнопка	Код в разрядах		
	Y2	Y1	YO
К1	0	0	0
К2	0	0	1
К3	0	1	0
К4	0	1	1
К5	1	0	0
Кб	1	0	1
К7	1	1	0
К8	1	1	1

Коды всех кнопок, вызывающих выполнение специальных функций, больше или равны 010Q. Поэтому команда JNC M11 (адрес 000115Q) будет осуществлять переход к команде МИ: CPI 010Q, которая сравнивает код нажатой кнопки с константой 010Q. Если код не равен 010Q, то команда JNZ M12 вызовет переход к Ml2, если равен, то это значит, что нажата кнопка *К9 (СБ)* и произойдет следующее: команда MOV H, C перепишет содержимое регистра C в регистр H, а затем команда JMP M2 вызовет безусловный переход к M2. Смысл этих действий в том, что байт, хранящийся в регистре C, станет старшим байтом адреса той ячейки, с которой работает пользователь. Этот новый старший байт адреса высветится на индикаторах порта 002Q, а содержимое ячейки памяти с новым адресом, составленным из нового старшего байта и старого младшего байта адреса, высветится на индикаторах порта OOOQ. Таким образом, функция, выполняемая монитором при нажатии на кнопку *К9 (СБ* — старший байт), — это формирование старшего байта нового адреса. Его значение может быть предварительно введено с помощью кнопок *К1 - К8* (цифры 0 — 7) в регистр C и высвечено на индикаторах порта OOOQ. На индикаторах пользователь видит следующее (рис. 7.11). Показания индикаторов порта OOOQ после нажатия *СБ* перемещаются на индикаторы порта 002Q (рис. 7.11,6), на индикаторах порта OOOQ после нажатия *СБ* высвечивается содержимое ячейки памяти с адресом, старший байт которого высвечен на индикаторах порта 002Q, а младший — порта 001Q.

Аналогичную функцию, только по формированию младшего байта адреса, выполняют кнопка *К10 {МБ* — младший байт) и связанный с ней фрагмент программы от команды M12: CPI 011Q до JMP M2. При нажатии на эту кнопку содержимое регистра C, высвеченное на индикаторах порта OOOQ, переписывается в регистр L, становясь младшим байтом адреса той ячейки, с которой работает пользователь, и высвечивается на индикаторах порта 001Q (рис. 7.12). На индикаторах порта OOOQ при этом высвечивается содержимое ячейки памяти с адресом, составленным из нового младшего байта и старого старшего байта.

Следующая кнопка *К11 (И* — индикация содержимого ячейки памяти и его изменение) выполняет две функции: просмотр содержимого ячеек памяти и изменение содержимого на новое в случае необходимости. При нажатии на эту кнопку подпрограмма SKL вырабатывает код 012Q, поэтому срабатывают команды условного перехода JNZ M12 и JNZ M13. После этого команда M13: CPI 012Q устанавливает флаг равенства нулю и команда JNZ M14 не осуществляет переход к M14. Затем команда MOV M, C переписывает содержимое регистра C (это содержимое пользователь видит на индикаторах порта OOOQ) в ячейку памяти, адрес которой содержится в паре регистров H-L (этот адрес пользователь видит на индикаторах портов 002Q и 001Q). Следующая команда INX H увеличивает содержимое H-Ьна единицу, а затем происходит безусловный переход (команда JMP M2) на фрагмент программы, который высвечивает на индикаторах адрес и содержимое следующей по порядку возрастания двоичных адресов ячейки памяти. Теперь ясно, что если нажимать только кнопку *К11,* то можно последовательно просматривать содержимое ячеек памяти в порядке возрастания их адресов (адрес высвечивается на индикаторах порта 002Q и 001Q, а содержимое — на OOOQ) ; но если между нажатием на кнопку *К11* с помощью кнопок *К1-К8* ввести в регистр C и на индикаторы порта OOOQ какое-либо двоичное число, то последующим нажатием на *К11* можно переписать это число в текущую ячейку памяти и перейти к индикации адреса и содержимого следующей ячейки памяти.

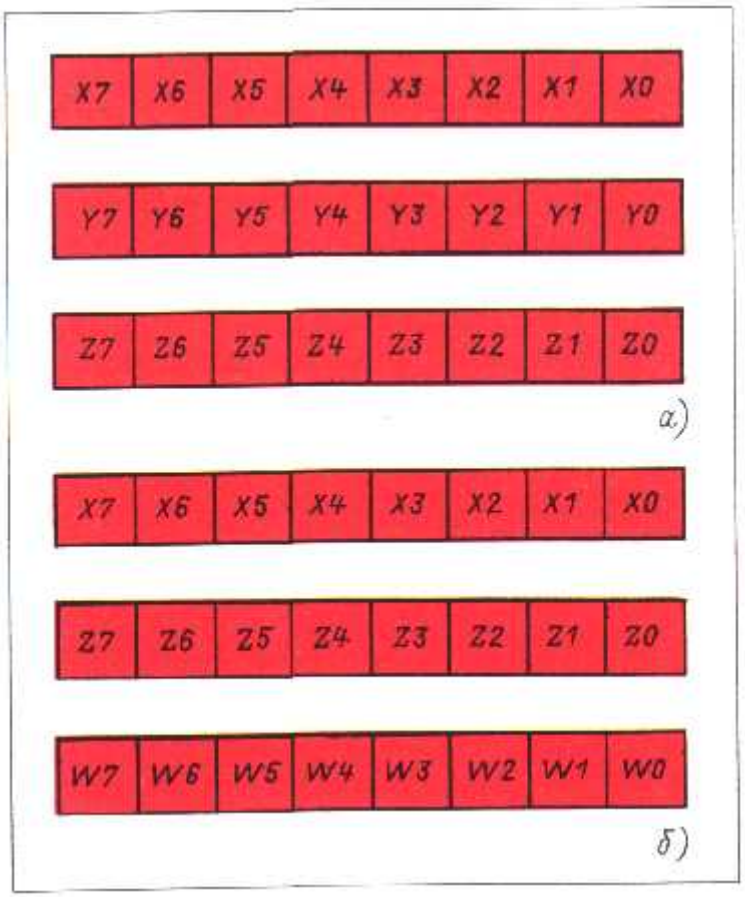


Рис. 7.11. Состояния индикаторов портов вывода:
а - до нажатия кнопки *СБ; б* - после нажатия кнопки *СБ*

Кнопка *К12 (П* — пуск программы) выполняет функцию запуска программы, записанной пользователем в ОЗУ с адреса, который содержится в паре регистров H-L и высвечен на индикаторах портов 002Q и 001Q. При нажатии этой кнопки программа вырабатывает код 013Q и команды JNZ Ml2, JNZ M13, JNZ M14 осуществляют переход к M14: CPI 013Q. Эта команда устанавливает флаг равенства нулю, команда JNZ M4 не осуществляет переход к M4, и следующая команда PCHL загружает в счетчик команд содержимое пары регистров H-L, что вызывает переход к выполнению команды, расположенной по этому адресу.

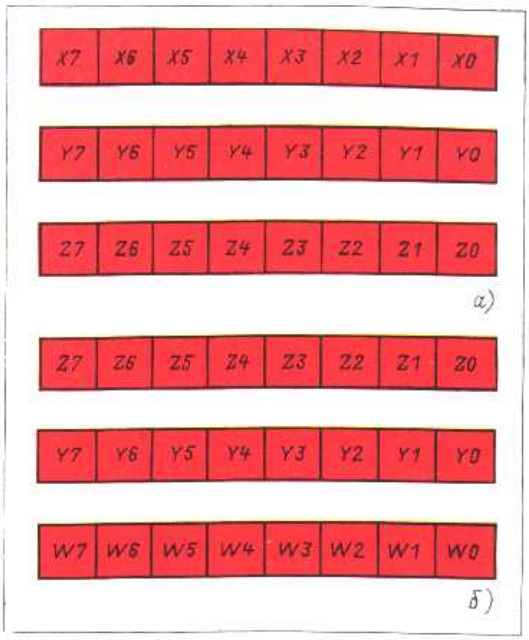


Рис. 7.12. Состояния индикаторов портов вывода:
а - до нажатия кнопки *МБ; б* - после нажатия кнопки *МБ*

Кнопки *К13-К16* не задействованы, и монитор не выполняет никаких функций при нажатии на них. Эти кнопки оставлены для дальнейшего расширения функций монитора.

Программа-монитор:

```
000000303                       JMPM1
000 001 070
000 002 000
000070061               Ml:     LXISP, 010 OOOQ
000071 000
000 072 020
000073041                       LXIH.0060000
000 074 000
000 075 004
000076 116              M2:      MOV C M
000077174                       MOV A, M
000100323                       OUT 0010
000 101 001
000102175                       MOVA.L
000103323                       OUT 0000
000 104 000
000105171               M3:     MOV A, C
000106323                       OUT 002Q
000 107 002
000110315               M4:     CALL SKL
000 111 175
000112000
000113376                       CP1 010Q
000 114 010
000115322                      JNCM11
000 116 134
000117 000
000120107                      MOVE, A
000121 171                      MOV A, C
000122027                      RAL
000 123 027                     RAL
000124027                      RAL
000125346                      ANI 370Q
000 126 370
000127260                      OR A B
000130117                      MOV C, A
000 131 303                     JMPM3
000 132 105
000 133 000
000134376               Mil:   CPI 010Q
000 135010
000136302                       JNZM12
000137 145
000 140 000
000 141 141                     MOVH, C
000142303                      JMPM2
000 143 076
000 144 000
000145376               M12:    CPI011Q
000 146 Oil
000147302                      JNZM13
000 150 156
000 151 000
000152151                       MOVL, C
000153303                       JMPM2
000 154 076
000 155 000
000156376               M13:    CPI 012Q
000 157 012
000160302                       JNZM14
000 161 170
000 162 000
000163161                       MOVM, C
000 164 043                     INX H
000 165 303                     JMPM2
000 166 076
000 167 000
000170376               M14:    CPI 013Q
000 171 013
```

```
000172302                       JNZ M4
000 173 110
000 174 000
000175351                       PCHL
000176000
000177076               SKL:    MVI A, OOOQ
000 200 000
000201323                       OUT003Q
000 202 003
000203333               M5:     IN 003Q
000 204 003
000205346                       ANI 017Q
000206 017
000207376                       CPI017Q
000210 017
000 211 302                      JNZM5
000 212 203
000 213 000
000214315                       CALLDL
000 215 277
000 216 000
000217325                       PUSH D
000 220 026              M8:     MVI D, 003Q
000 221 003
000222036                       MVIE, 376Q
000 223 376
000224173               M7:     MOV A, E
000225323                       OUT003Q
000 226 003
000 227 007                      RLC
000230137                       MOVE, A
000231333                       IN003Q
000 232 003
000233346                       ANI017Q
000 234 017
000235376                       CPI017Q
000236 017
000 237 302                      JNZ M6
000 240 254
000 241 000
000 242 025                      DCR D
000 243 172                      MOV A, D
000 244 376                     CPI 377Q
000 245 377
000 246 302                     JNZ M7
000 247 224
000 250 000
000251303                       JMPM8
000 252 220
000 253 000
000254315               Мб:     CALL DL
000 255 277
000 256 000
000257017               M10:     RRC
000 260 322                      JNC M9
000261 274
000 262 000
000 263 365                       PUSH PSW
000264172                       MOV A, D
000 265 306                       ADI 004Q
000 266 004
000 267 127                      MOV D, A
000270361                       POP PSW
000271303                       JMPM10
000 272 257 000 273 000
000274172               M9:     MOV A, D
000275321                       POP D
000276311                       RET
000 277 365              DL:     PUSH PSW
000 300 325                      PUSH D
000301021                       LXID, 0010160
000302016 000303 002
000 304 033                       OCX D
000305 172                      MOV A, D
```

```
000 306 263                    ORA E
000 307 302                    JNZ N
000310304 000311 000
000312321                      POPD
000313361                      POP PSW
000314311                      RET
```

Теперь можно сформулировать инструкцию по работе на микро-ЭВМ.

7.5. ИНСТРУКЦИЯ ПО РАБОТЕ НА МИКРО-ЭВМ

1. После включения микро-ЭВМ на индикаторах портов 002Q и 001Q высветится адрес младшей ячейки ОЗУ 00001100В и 00000000 В соответственно, а на индикаторах порта ОООQ высветится содержимое этой ячейки.

2. Для того чтобы посмотреть содержимое произвольной ячейки ОЗУ, необходимо:

а) набрать с помощью кнопок 0 — 7 на индикаторах порта ОООQ старший байт адреса этой ячейки и нажатием на кнопку *СБ* переслать его на индикаторы порта 002Q;

б) набрать с помощью кнопок *0 — 1* на индикаторах порта ОООQ младший байт адреса этой ячейки и нажатием на кнопку *МБ* переслать его на индикаторы порта 001Q. После этого на индикаторах порта ОООQ высветится содержимое необходимой ячейки.

3. Для того чтобы записать в произвольную ячейку необходимый код команды или данные, нужно:

а) выполнить п. 2 Инструкции;

б) набрать с помощью кнопок 0 — 7 необходимый код на индикаторах порта ОООQ. Этот код также попадет в регистр С;

в) нажать кнопку *И,* что вызовет запись кода в ячейку памяти и одновременно индикацию адреса и содержимого следующей ячейки.

4. Для того чтобы запустить программу, нужно:

а) записать ее в память (см. пп. 1, 2, 3 Инструкции) ;

б) набрать с помощью кнопок 0 — 7 старший байт адреса команды, с которой должно начаться выполнение программы на индикаторах порта ОООQ, и нажатием на кнопку *СБ* переслать его на индикаторы порта 002Q;

в) набрать с помощью кнопок 0 — 7 младший байт адреса команды, с которой должно начаться выполнение программы на индикаторах порта ОООQ, и нажатием на кнопку *МБ* переслать его на индикаторы порта 001Q;

г) запустить программу нажатием на кнопку 77.

5. Для того чтобы прервать выполнение программы, необходимо нажать на кнопку СБРОС.

8

СБОРКА И ОТЛАДКА ПМ-ЭВМ

8.1. ЭТАПЫ СБОРКИ И ПРОВЕРКИ УЗЛОВ

Читатель ознакомился с принципами работы микропроцессорной системы на базе МП типа КР580ИК80А, системой команд микропроцессора и схемой ПМ-ЭВМ, построенной на базе данного микропроцессора. Теперь ему предстоит собрать микро-ЭВМ и, используя простейшие средства отладки, довести ее до рабочего состояния.

Для сборки необходима плата, на которой будут располагаться микросхемы и другие элементы ПМ-ЭВМ. Плата изготовляется из стеклотекстолита или гетинакса толщиной 1,5 — 2,0 мм размером примерно 210х240 мм. В зависимости от имеющихся возможностей монтаж платы может быть выполнен по-разному.

Если плата не из фольгированного материала, то предварительно делается разметка на миллиметровке размещения деталей, затем миллиметровка наклеивается на плату и сверлятся отверстия под все выводы деталей. При небольших размерах платы микросхемы можно расположить перпендикулярно длинной стороне платы в два ряда, сделав пропилы тонкой ножовкой для выводов. В этом случае отпадает необходимость сверлить отверстия под каждый вывод. После отмывания миллиметровки детали устанавливаются на плате с одной стороны так, что их выводы попадают в просверленные отверстия и слегка отгибаются, препятствуя выпадению деталей. Микросхемы, пока они не подпаяны, требуют осторожного обращения ввиду опасности повреждения разрядом статического электричества. Берите их не за выводы, а за корпус!

Выводы деталей в соответствии с принципиальной схемой соединяются проводами путем пайки припоем ПОС-60 с жидким канифольным флюсом маломощным низковольтным паяльником, корпус которого подсоединен к заземлению через резистор сопротивлением несколько сотен килоом. Для монтажа можно использовать, например, провод МГТФ-0,05, зачищенный с концов на 5 мм с помощью кусачек, в которых сделаны специальные выемки глубиной около 0,5 мм.

Если для монтажа используется макетная плата с металлизированными отверстиями для выводов деталей и контактными площадками для подпайки соединительных проводов, то после размещения микросхем и других деталей на плате они подпаиваются с обратной стороны к плате, а затем соединяются проводами аналогично предыдущему.

От качества пайки существенным образом зависит надежность работы будущей ПМ-ЭВМ. Использование специально разработанной печатной платы облегчает последующие монтаж, отладку и повышает надежность работы устройства. Для ПМ-ЭВМ может быть использована как односторонняя, так и двухсторонняя плата. Если плата односторонняя, придется часть монтажа выполнить навесными проводниками. В двухсторонней плате, если сквозные отверстия, предназначенные для соединения печатных проводников с разных сторон платы, не металлизированы, в них необходимо вставить кусочки провода и пропаять с обеих сторон. Если нужно произвести изменения в плате с печатным монтажом, лишние соединения ликвидируются, для чего дважды разрезается печатный проводник острым ножом на расстоянии 1-2 мм и середина удаляется. Новые соединения можно сделать навесными проводниками. Монтаж каждого из блоков: микропроцессорного, памяти и устройств ввода/вывода - можно выполнить на отдельных платах, что в какой-то мере облегчает отладку. В конструкции, выполненной авторами, ПМ-ЭВМ размещена на плате с двухсторонним печатным монтажом. На торцевой стороне платы имеются два разъема. Через один разъем подключаются клавиатура и источники питания, а через другой выведены внутренние шины МП системы, что позволяет подключать различные расширяющие блоки.

Отладку ПМ-ЭВМ можно производить поэтапно по мере сборки В первую очередь необходимо собрать блок процессора (микросхемы D1-D7, D9, D10, D27, D28, D30, D31), причем для микросхемы D1 (микропроцессор) крайне желательно поставить панельку. Проверьте правильность монтажа, выньте микропроцессор из панельки и подайте питание. Проверьте тестером наличие питающих напряжений на выводах микросхем. При наличии осциллографа проверьте работу тактового генератора - на выводах *10 к 11* микросхемы D2 должны наблюдаться тактовые импульсы Ф2 и Ф1. Выключите питание и вставьте микропроцессор в панельку, подключите резисторы по 300 510 Ом между общим проводом и линиями шины данных DB7 — DBO. Тем самым имитируется считывание команды NOP (код OOOQ). Поставьте переключатель К18 в положение АВТОМАТ, подайте питание и проверьте с помощью осцилло-графа наличие импульсов на линиях шины адреса АВ11 — АВО и АВ15-АВ12. Длительность импульсов (длительность уровня логического нуля или логической единицы) на линии АВО должна составить четыре тактовых интервала (машинный цикл команды NOP), т. е. 4 мкс. Длительность импульсов на следующих линиях шины адреса последовательно удваивается. Если на какой-либо линии нет импульсов, проверьте последовательнс прохождение сигнала от вывода МП до соответствующей линии. Если сигнала нет и на выводе МП, то можно предположить либо замыкание данной сигнальной линии на какой-либо постоянный уровень, либо неисправность МП по данному выводу. Если импульсов нет ни на одной из линий шины адреса и на соответствующих выводах МП, то, скорее всего, неисправен МП. Но к тогда прежде чем забраковать МП, сначала проверьте напряжения и сигналы на всех его выводах. Желательно проверить МП. вставив его в панельку заведомо исправной МП-системы. Если сигнал исчезает по пути от МП к адресной шине, нетрудно локализовать неисправность, которая может быть вызвана следующими причинами: ошибками в логической схеме, ошибками в монтаже, случайными замыканиями или обрывами проводящих линий, неисправностью соответствующих микросхем. Прежде чем выпаивать неисправнук микросхему, убедитесь еще раз, что других причин неисправности нет, так как замена микросхемы, особенно в печатной плате, — процесс весьмг трудоемкий. Если микросхема заведомо неисправна, проще всего ее извлечь, перекусив кусачками ножки. Затем уже можно извлечь по отдельности остатки каждой ножки с помощью пинцета и паяльника и прочистить отверстия с помощью заостренной спички или высверлить тонким сверлом. Извлечь микросхему без повреждения значительно сложнее. Для этого необходимо иметь либо паяльник с многими жалами или широким жалом, либо паяльник с отсосом.

После проверки шины адреса проверьте с помощью осциллографа выдачу сигнала R (чтение) на выводе *11* микросхемы D27 и сигнала RW (обращение к памяти) на выводе *3* микросхемы D28. Если сигналов нет, проверьте наличие сигналов DBIN на выводах *10 к 13* микросхемы D27, MR на выводе *12* микросхемы D27, STSTB на ножках *4 и 13* микросхемы D29 и соответствующего машинному циклу чтения команды управляющего слова (его код 242Q) на шине данных DBO-DB7, т. е. на выводах *3, 6, 10, 13* микросхемы D9 и *2, 3, 6, 7* микросхемы D29 соответственно. Если какого-либо сигнала нет, проверьте по цепочке логических схем последовательно, придя к выводам шины данных, DBIN и SYNC микропроцессора на выводах *10, 9, 8, 7, 3, 4, 5,* 6, *17, 19* соответственно. Проверьте функционирование кнопки СБРОС. При ее нажатии МП не выдает никаких сигналов, при отпускании вновь появляются описанные выше сигналы.

Поставьте переключатель *К18* в положение ШАГ, нажмите и отпустите кнопку СБРОС. Проверьте тестером или логическим пробником состояние шины адреса, на которой должен быть адрес 000Q 000Q, наличие сигналов R, DBIN и RW, а также наличие кода 000Q на шине данных МП. Нажимайте кнопку *К17.* При каждом

нажатии кнопки на шине адреса должен появляться адрес, на единицу больший предыдущего, а состояние остальных контролируемых линий не должно изменяться. Если этого не происходит, проверьте исправность схемы шагового режима (микросхемы D3I.3, D31.4, D30).

После проверки микропроцессорного блока снимите питающие напряжения, отключите резисторы, временно подпаянные к шине данных, и соберите блок памяти, состоящий из микросхем D8, D11-D15, пока не подпаивая микросхемы ОЗУ (D12, D13). Подключите питание, поставьте переключатель К18 в положение ШАГ, нажмите и отпустите кнопку СБРОС. Начинается исполнение программы монитора в шаговом режиме по машинным циклам. Проследите с помощью тестера, проверяя коды на шине данных, исполнение команд монитора, начиная с команды JMP по адресу 000Q 000Q и кончая командой IN по адресу 000Q 203Q. Если команды исполняются в соответствии с программой и в соответствующих местах программы вырабатываются сигналы R, W. IN и OUT, то можно приступить к следующему этапу изготовления и отладки ПМ-ЭВМ; если же нет, тс необходимо проверять: цепь прохождения кода считываемой команды до выводов шины данных МП, правильность дешифрации адреса по сигналам выборки памяти на выводах *8* микросхем D12, D13 и выводах *14* микросхем D14, D15, правильность выдачи управляющего слова по сочетанию на выходах микросхемы D29, наличие сигнала WR на выходе МП при исполнении машинных циклов записи в память и вывода.

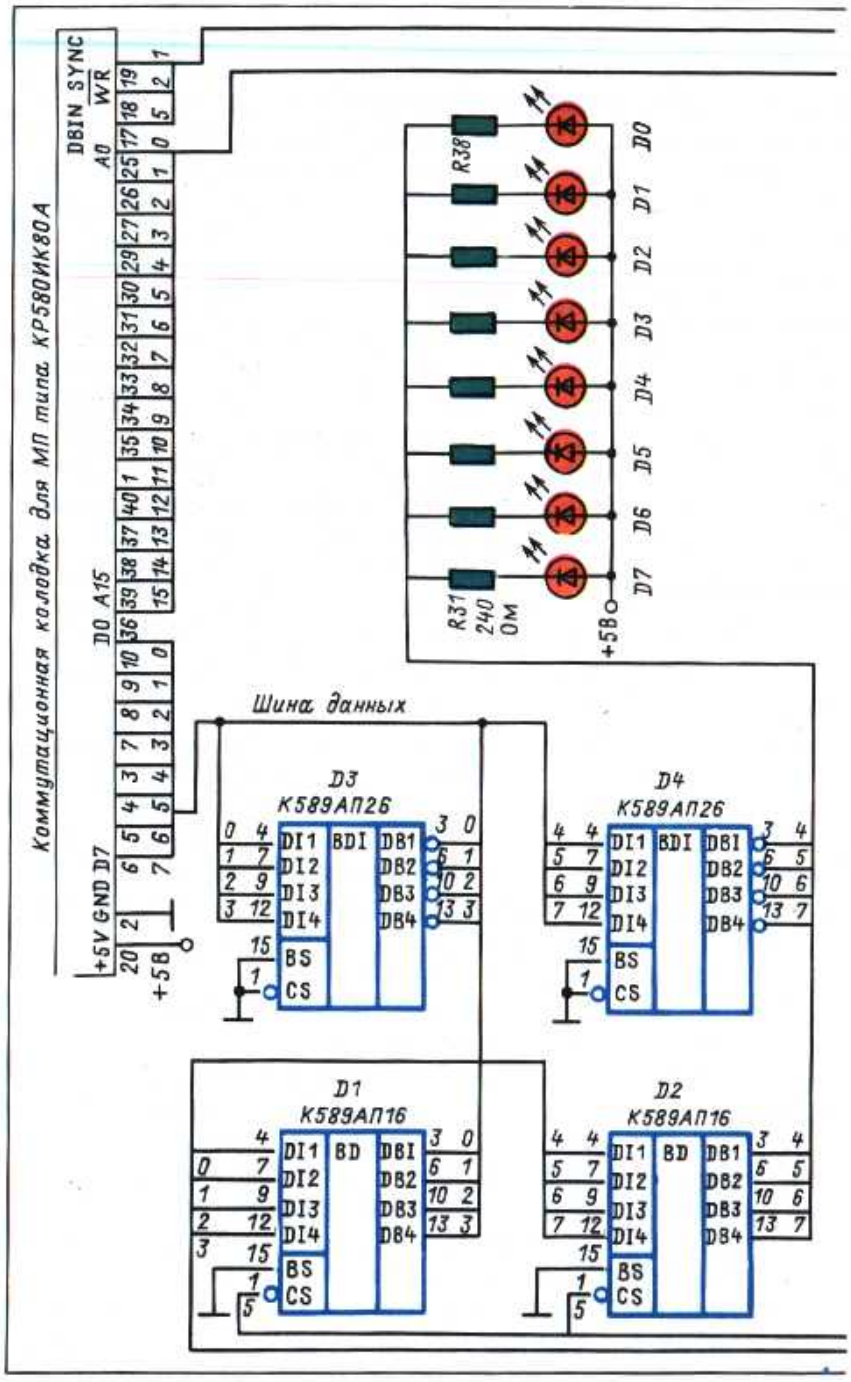
Проверив работу микропроцессорного блока совместно с ПЗУ в шаговом режиме, отключите питание и соберите остальную часть схемы ПМ-ЭВМ, впаяв в том числе и микросхемы ОЗУ. Подайте питание, поставьте переключатель *К18* в положение АВТОМАТ, нажмите и отпустите кнопку СБРОС. Если схема собрана правильно и все вновь включенные элементы исправны, то на индикаторах портов 001 и 000 высветится адрес начальной ячейки оперативной памяти 014Q OOOQ, а индикаторы порта 002 высветят содержимое этой ячейки. Проверьте работу клавиатуры, ее цифровой и функциональной частей. Если работа соответствует описанию, приведенному в гл. 7, то дальнейшие проверки будут чисто программными, например проверка ОЗУ. Если клавиатура функционирует неправильно, в целях проверки содержимого ПЗУ выясните, не работает ли клавиша просмотра памяти И. Переведите ПМ-ЭВМ в шаговый режим, нажмите и отпустите кнопку СБРОС и при последовательном исполнении команд монитора проследите прохождение всех сигналов. На каждом шаге порт 002 будет отображать информацию на шине данных, в то время как порты 001 и 000 будут отображать информацию, выдаваемую по командам OUT в соответствующие порты. В шаговом режиме затруднительно проверить правильность функционирования программы-монитора в той части, где анализируются коды нажатых клавиш, так как имеющееся в ней обращение к подпрограмме временной задержки занимает сотни машинных циклов и мало у кого хватит терпения дойти до конца. Если неисправность не удалось обнаружить и устранить, то необходимо собрать несложное устройство — статический аппаратный эмулятор, описываемый в § 8.2. Можно и начать с его сборки, прежде чем приступать к сборке и отладке ПМ-ЭВМ, и тем сберечь немало времени. В заключение еще раз напомним, что в процессе отладки микросхемы следует вставлять и вынимать только при выключенном питании и после каждой переделки, включив питание, следует прежде всего проверить, подается ли оно на все микросхемы.

8.2. СТАТИЧЕСКИЙ АППАРАТНЫЙ ЭМУЛЯТОР

Ограничив поставленную задачу проверкой правильности функционирования логических схем, обрамляющих МП, используем тот факт, что в МП-системах на базе микросхемы КР580ИК80А передача информации осуществляется уровнями. Следовательно, если заменить МП устройством, вырабатывающим в статике такие же выходные сигналы, можно проверить работу всех зависящих от него узлов, т. е. правильность адресации микросхем, правильность ввода/вывода, работу микросхем в статике, правильность коммутации. Такой метод проверки называется *тестированием статическими сигналами.* Устройство тестирования представляет статический аппаратный эмулятор микропроцессора (САЭ). На рис. 8.1 приведена схема САЭ. В устройстве использовано пять микросхем трех типов. САЭ осуществляет управление МП-системой точно так же, как и микропроцессор. В его составе имеются переключатели *A15-AO.* Каждый из них одним концом соединяется с общим проводом, а другим — с согласующим резистором 4,7 кОм, подключенным к источнику питания + 5 В. В зависимости от положения переключателей через кабель на ножки коммутационной колодки, соответствующие адресным выходам микропроцессора *A15 — AO,* подаются логические уровни 0 или 1, имитируя сигналы адресной шины МП. В составе САЭ имеются также переключатели *D7-DO. С* их помощью логические уровни 0 или 1 подаются на входы микросхем *D1 nD2* типа К589АП16, представляющих двунаправленные буферные усилители с высокой нагрузочной способностью. Логические уровни 0 или 1 через *D1* и *D2* подаются на ножки коммутационной колодки, соответствующие выводам *D7-DO* микропроцессора, имитируя выходные сигналы шины данных МП. К этим же ножкам подключены входы микросхем *D3* и *D4* типа К589АП26, представляющих инвертирующие двунаправленные буферные усилители. Выходы *D3* и *D4* нагружены на светодиодные индикаторы, постоянно отображающие состояние шины данных микропроцессора.

Особенностью микропроцессора КР580ИК80А является использование шины данных в начале каждого машинного цикла для выдачи управляющего слова, которое определяет один из десяти типов машинного цикла. По этой причине прежде чем с помощью САЭ осуществлять передачу информации по шине данных между САЭ, имитирующим МП, и памятью или между САЭ и УВВ, необходимо задать соответствующий тип машинного цикла путем выдачи кода управляющего слова по шине Данных и стробирования его сигналом SYNC. По сигналу SYNC тактовый генератор тестируемой системы вырабатывает импульс STSTB, стробирующий прием

и фиксацию управляющего слова системным контроллером тестируемой системы. После снятия сигнала SYNC можно передавать информацию по шине данных в соответствии с заданным управляющим словом, так как но шине управления будут выдаваться соответствующие управляющие сигналы. Так, если задан машинный цикл выдачи информации из процессора в память, УВВ или стек, необходимо переключателями *D7-DO* набрать код управляющего слова, перевести переключатель SYNC из положения О в положение 1 и обратно, затем переключателями *D7-DO* набрать код передаваемой информации, переключателями *A15-AO* набрать адрес ячейки памяти или порта вывода и перевести переключатель WR из положения 0 в положение 1 и обратно. При этом по сигналу WR и на основании зафиксированного управляющего слова системный контроллер тестируемой системы вырабатывает управляющий сигнал W или OUT соответственно, по которому информация, выданная САЭ по шине данных, запишется в заданную ячейку памяти или будет принята заданным портом вывода. Если задан машинный цикл приема информации из памяти, УВВ или стека, необходимо переключателями *D7-DO* набрать код управляющего слова, перевести переключатель SYNC из положения О в положение 1 и обратно, затем набрать переключателями *Л15 — АО* адрес ячейки памяти или порта ввода и перевести переключатель DBIN из положения 0 в положение 1 и обратно. При этом в положении 1 на светодиодных индикаторах САЭ будет отображаться принимаемая информация. Аналогично предыдущему системный контроллер тестируемой системы вырабатывает управляющий сигнал R или IN, по которому информация из заданной ячейки памяти или порта ввода передается по шине данных в САЭ.



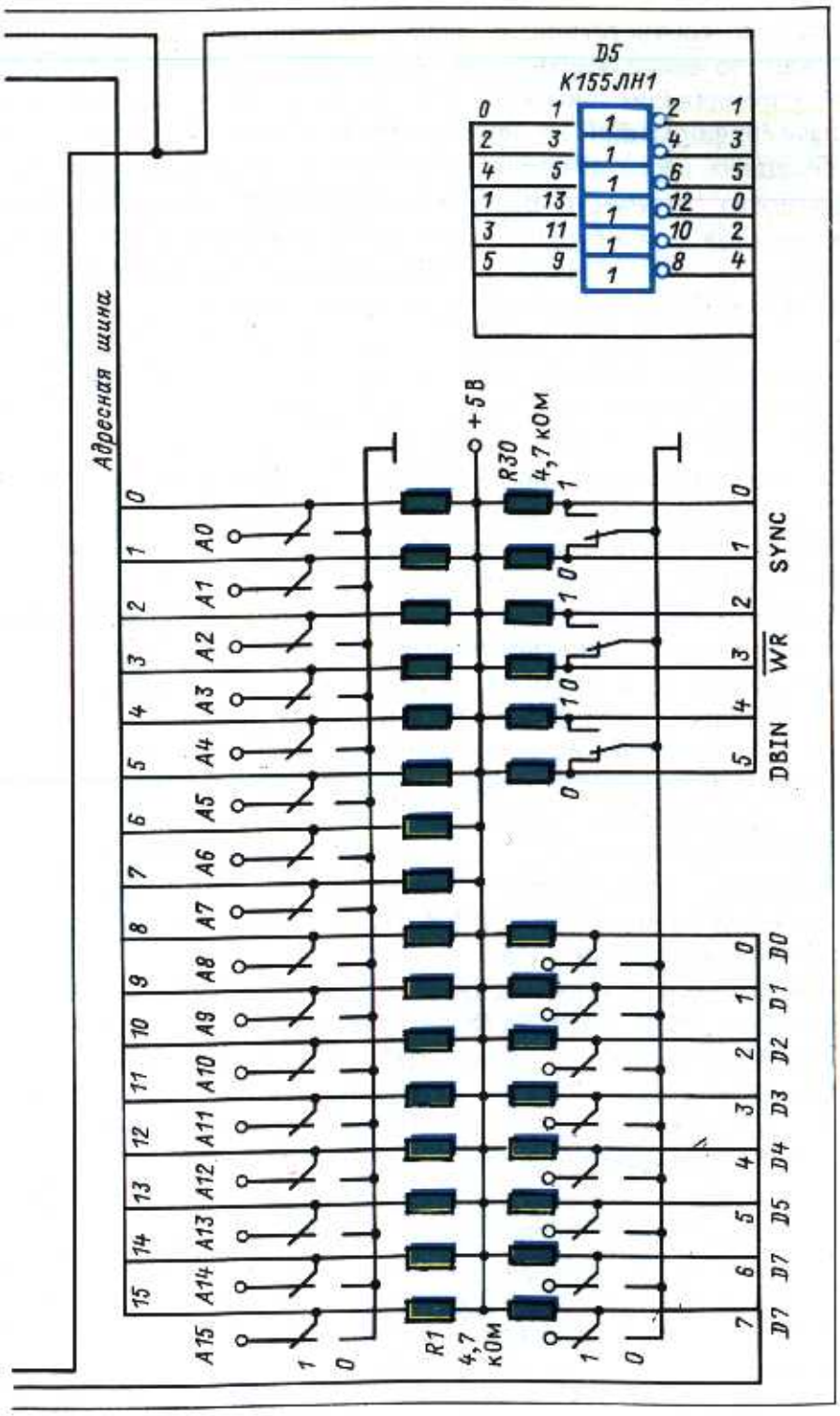


Рис. 8.1. Схема САЭ

Заметим, что с целью гашения дребезга контактов к переключателям DBIN, WR и SYNC подключены триггеры, построенные на инверторах микросхемы *D5* типа К155ЛЫ1. Поскольку остальные переключатели не оснащены схемами гашения дребезга, они должны переключаться только тогда, когда DBIN, WR и SYNC находятся в положении 0.

Во избежание конфликтной ситуации на шине данных при непредусмотренном положении переключателей при положении переключателя DBIN в 1 блокируется выдача информации с переключателей *D7-DO* на шину данных путем подачи логической 1 на входы выборки кристалла микросхем *D1* и *D2,* что переводит их выходы в третье состояние.

Отладка шины адреса ПМ-ЭВМ. Определим, возможна ли передача логических 0 и 1 по адресным линиям AB15 — AB12 и AB11-ABO. С этой целью введем САЭ в проверяемую ПМ-ЭВМ вместо МП путем установки коммутационной колодки САЭ в панельку МП и подадим питающие напряжения на ПМ-ЭВМ.

Проверяем, что для всех адресных линий производится переключение сигнала с уровня 0 на уровень 1 под управлением МП. Если обнаружено, что для некоторых линий такое переключение отсутствует, потребуется дальнейшее исследование. С этой целью осуществляется проверка каждой цепи путем прослеживания уровня напряжения в выбранных точках на протяжении всего информационного канала.

Отладка шины управления ПМ-ЭВМ. При отладке шины управления ПМ-ЭВМ будем использовать управляющие переключатели САЭ — DBIN, WR и SYNC. Отладка состоит из двух этапов: на первом проверяется правильность записи кода управляющего слова на микросхеме *D29,* на втором — правильность формирования системным контроллером ПМ-ЭВМ управляющих сигналов. При этом переключателями *A15-A0* набран код 200Q 200Q.

Устанавливаем, что выходные управляющие сигналы ПМ-ЭВМ правильно отражают входные воздействия микропроцессора. Если на каком-то шаге процедуры будет обнаружено нарушение этого соответствия, можно легко осуществить статическую проверку логических сигналов и локализовать неисправность.

Отладка шины данных ПМ-ЭВМ. При отладке шины данных будем проверять возможность двунаправленной передачи данных от МП к ОЗУ и УВВ и от ПЗУ, ОЗУ и УВВ к МП. Как и в предыдущем разделе, во избежание нежелательной выборки устройств и возникновения конфликта на и тане данных переключателями *A15-A0* набран код 200Q 200Q.

Если выходные сигналы на шине данных на некотором шаге процедуры не являются корректными, необходимо определить причины неисправности. Это достигается путем проверки статических сигналов на протяжении информационного канала.

Успешное завершение процедуры означает, что шина данных ПМ-ЭВМ при выводе данных функционирует правильно, т. е. шина данных должна выводить информацию, соответствующую сигналам от МП.

Отладка схем дешифрации адреса и выработки сигналов выборки устройств. При отладке будем проверять наличие сигналов выборки устройств в четырех режимах: чтения из памяти, записи в память, ввода из УВВ, вывода в УВВ.

Ускоренная проверка ПМ-ЭВМ. Приведенная методика проверки позволяет полностью и последовательно проверить в статике работу всех устройств ПМ-ЭВМ. Однако часто можно ограничиться проверкой по разделу "Отладка схем дешифрации адреса и выработки сигналов выборки устройств", так как данная проверка является итоговой. Если обнаружены какие-либо неисправности, то необходимо провести полную проверку. Хотя вероятность нормальной работы при выбранных параметрах схемы весьма высока, проверка в статике не гарантирует работоспособность ПМ-ЭВМ в динамических условиях, так как не проверяются временные соотношения между сигналами в системе, не известны действительные задержки логических схем и не исключено, что к какой-нибудь сигнальной линии по ошибке подпаян конденсатор. Кроме того, нет гарантии, что все ячейки ОЗУ функционируют нормально. Поэтому после отладки ПМ-ЭВМ с помощью САЭ, когда нормально функционирует клавиатура, вводится и считывается информация, следует произвести программные проверки, описываемые в § 8.3.

8.3. ОТЛАДКА В РАБОЧЕМ РЕЖИМЕ

Итак, ПМ-ЭВМ собрана, реагирует на нажатия клавиш, индикация загорается и гаснет, по-видимому, так, как нужно. Но это еще не значит, что ПМ-ЭВМ полностью проверена и функционирует правильно. Гарантировать стопроцентную проверку вычислительной системы вообще невозможно, тем более с помощью имеющихся у читателя простейших средств. МП может неправильно выполнять некоторые операции и притом только для некоторых сочетаний кодов, микросхемы запоминающих устройств могут иметь неисправные или ненадежные ячейки, искажающие некоторые сочетания кодов. Отдельные элементы ЭВМ могут работать в предельных режимах по нагрузке, по временным допускам, по питанию, по уровню помех и выдавать сбои при изменении внешних условий. Рано или поздно это выявится в процессе эксплуатации. Однако надлежаще проведенные испытания в начале эксплуатации позволят выявить значительную часть дефектов и устранить их. Первое, что необходимо сделать, - это проверить работоспособность ОЗУ. Простейший способ - записывать в ячейки ОЗУ с помощью клавиатуры различные коды, а затем последовательно считывать их, проверяя совпадение. Такой метод очень трудоемок, так как ПМ-ЭВМ имеет 1024 ячейки ОЗУ, к тому же некоторые ячейки, надежные в статике, в динамике могут выдавать неверные коды. Приведенная ниже программа позволяет автоматически записать заданный код в произвольно заданную зону оперативной памяти. Программа записывается, начиная с ячейки ОЗУ с адресом 014Q OOOQ и занимает 24 ячейки. Зона памяти, в которую заносится код, может начинаться с ячейки 014Q 030Q. Этот адрес заносится в регистровую пару H командой LXIH. Адрес конца зоны заносится в регистровую пару D командой LXID. Устанавливаемый код задается вторым байтом команды MVIM адреса 014Q 007Q и 014Q 024Q. Проверку правильности записи можно произвести вручную, последовательно просматривая память:

```
014000041                MI: LXIH, 014Q 030Q
```

```
014001 030
014002014
014003021                    LXI D, 017Q377Q
014 004 377
014005017
014006066              M2: MVIM, OOOQ
014 007 000
014010043                    INXH
014011 174                    MOV A, H
014012222                    SUBD
014 013 302                   JNZ M2
014014 006
014015  014
014016173                    MOV A, E
014017225                    SUBL
014 020 302                   JNZ M2
014 021 006
014022014
014 023 066                   MVI M, OOOQ
014 024 000
014025303                    JMPM1
014 026 000
014027014
```

Путем незначительного усложнения программы можно автоматизировать проверку правильности записи:

```
014 000 006                   MVI B, OOOQ
014 001 000
014002041              Ml: LXI H, 014Q 055Q
014003055
014 004 014
014 005 021                   LXI D, 020Q OOOQ
014 006 000
014 007 020
014010160              M2: MOV M, B
014011 170                    MOV A, B
014012226                    SUBM
014013302                    JNZM4
014014  035
014015  014
014016043              M3: INX H
014017174                   MOV A, H
014 020 222                   SUB D
014021302                   JNZM2
014022010
014023  014
014024  173                    MOV A, E
014025225                   SUB L
014026302                   JNZ M2
014027010
014030014
014031000                   NOP
014032303                    JMPM1
014033  002
014034  014
014035174              M4: MOV A. H
014036323                   OUT 0010
014037 001
014040175                   MOV A, L
014041323                    OUT OOOQ
014 042 000
014043 176                    MOV A, M
014044323                   OUT002Q
014045 002
014046315                   CALLSKL
014047 177
014050000
014051303                    JMPM3
014052016
014053 014
```

В данной программе проверочный код заносится по адресу 014Q 001Q. Сверка кодов производится последовательно в порядке возрастания адресов. При обнаружении ошибки программа выводит адрес в порт 001 (старшие разряды) и в порт 000 (младшие разряды), содержимое ячейки — в порт 002 и переходит в режим опроса клавиатуры. Нажатие любой кнопки запускает программу на проверку следующих ячеек до новой ошибки, и так далее до конца проверяемой зоны. Для более полной проверки необходимо задавать различные проверочные коды, в предельном случае — от OOOQ до 377Q. Поставив на место команды NOP по адресу 014Q 031Q команду INRB (код 004Q), получим программу автоматической проверки памяти для всех кодов. Читатель" может сам оценить, какая программа удобнее в эксплуатации. Необходимо заметить, что ячейки ОЗУ, отведенные для программы проверки, должны быть заведомо исправными, и чем длиннее программа, тем труднее найти зону для ее размещения.

В результате проверки выясняется, какой частью ОЗУ можно располагать для работы. Лучше всего, когда все ячейки ОЗУ исправны и, значит, программы могут последовательно размещаться в ячейках памяти. Однако и при наличии дефектных ячеек (если их не слишком много) можно использовать почти весь оставшийся объем памяти, обходя дефектные места с помощью команды JMP.

С помощью простейших пробных программ необходимо также проверить исполнение всей системы команд МП. Например, программа MVI A, OPERA ADI, OPERB OUT PORT1 проверяет исполнение команд MVIA, ADI и OUT.

8.4. ПОДГОТОВКА ПМ-ЭВМ К РАБОТЕ

После того как проведенные проверки подтвердили работоспособность микро-ЭВМ, можно приступить к ее эксплуатации. От условий эксплуатации зависит окончательное конструктивное оформление ПМ-ЭВМ. В лабораторных условиях можно оставить все в таком виде, в каком производилась отладка, т. е. соединенные проводами или плоскими кабелями отдельные платы, клавиатуру и лабораторные источники питания, приведя только в порядок проводку и проверив все соединения.

Если предполагается эксплуатировать ПМ-ЭВМ в стационарных условиях класса или аудитории, то желательно конструктивно оформить ее в виде отдельного стенда, на котором должны быть жестко закреплены все узлы и устройства с аккуратно выполненными связями между ними с помощью кабелей наименьшей возможной длины. При этом можно пользоваться либо лабораторными источниками питания, либо встроенными в стенд автономными. ПМ-ЭВМ потребляет (без дополнительных периферийных устройств) от источника + 5 В примерно 1,3 А, от источника + 12 В примерно 30 мА и от источника -5 В примерно 30 мкА. Желательно иметь некоторый, например двукратный, запас по источникам питания, имея в виду подключение дополнительной памяти, периферийных микросхем и устройств. В переносном варианте ПМ-ЭВМ легко размещается вместе с источниками питания в небольшом чемоданчике размерами ЗбОх 250х 80 мм и массой не более 3 кг.

После включения ПМ-ЭВМ переведите переключатель *К18* в положение АВТОМАТ и нажмите кнопку СБРОС. После отпускания кнопки СБРОС на индикаторах портов 001 и 000 высветится адрес начальной ячейки ОЗУ (014Q OOOQ), а на индикаторах порта 002 — содержимое этой ячейки. Нажимая кнопку И, убедитесь, что адрес, индицируемый портами 001 и 000, при каждом нажатии увеличивается на единицу. Снова нажмите кнопку СБРОС и убедитесь, что цифровые клавиши функционируют нормально, т. е. при каждом нажатии индицируемое портом 002 число сдвигается на три двоичных разряда влево и освободившееся место заполняется числом, соответствующим нажатой клавише. Убедитесь также, что при нажатии кнопок *СБ* или *МБ* информация, индицируемая портом 002, перемещается в порты 001 или 000 соответственно. Снова нажмите и отпустите кнопку СБРОС и переведите переключатель *К18* в положение ШАГ. Порт 002 индицирует состояние шины данных, соответствующее исполнению машинного цикла одной из команд программы-монитора от адреса OOOQ 220Q до OOOQ 251Q. При нажатии кнопки *К17* МАШИННЫЙ ЦИКЛ последовательно исполняются команды между данными адресами. Этот процесс повторяется неограниченно долго — до тех пор, пока не будет нажата какая-либо из кнопок клавиатуры от *К1* до *К16,* и тогда произойдет переход к подпрограмме временной задержки. Снова нажмите и отпустите кнопку СБРОС. Порт 002 индицирует код 303Q. Это код команды JMP M1, команды программы-монитора, находящейся по адресу OOOQ OOOQ. Последовательно нажимая кнопку *К17* МАШИННЫЙ ЦИКЛ, можно проследить исполнение программы-монитора по машинным циклам. Просмотреть по шагам исполнение рабочей программы можно только в том случае, если она циклическая. Тогда переключатель *К18* первоначально должен находиться в положении АВТОМАТ. Нажимаем и отпускаем кнопку СБРОС, вводим начальный адрес рабочей программы в соответствии с инструкцией (§ 7.4), нажимаем кнопку П и переключаем *К18* в положение ШАГ. Порт 002 индицирует состояние шины данных, соответствующее исполнению машинного цикла одной из команд рабочей программы. При нажатии кнопки *К17* последовательно исполняются команды рабочей программы. Правильность их

исполнения проверяется по индикаторам порта 002. Не забывайте перед началом работы перевести переключатель *К18* в положение АВТОМАТ и нажать кнопку СБРОС.

9

РАБОТА С ПМ-ЭВМ

9.1. ПРОГРАММИРУЕМЫЙ КАЛЬКУЛЯТОР

Первая идея, которая приходит в голову читателю, получившему в руки ПМ-ЭВМ, т. е. вычислительную машину, — использовать ее для вычислений. Лишь потом, приобретая некоторый опыт, он поймет, что вычисления — весьма малый клочок обширного поля возможных применений микро-ЭВМ.

Система команд микропроцессора КР580ИК80А, использованного в ПМ-ЭВМ, позволяет непосредственно выполнять логические операции и операции сложения и вычитания над двоичными и двоично-кодированными десятичными числами, имеющими формат один или два байта. Тем не менее, используя тот факт, что микро-ЭВМ — программируемое устройство, имеющее память, можно реализовать практически неограниченное множество различных операций (арифметических, логических, функциональных и т. д.) над исходными данными, представленными в произвольно заданном формате. Эти операции обычно оформлены в виде стандартных программ и составляют основу ядра программно-математического обеспечения микро-ЭВМ. Запрограммировав самостоятельно ряд вычислительных задач большей или меньшей сложности, читатель вскоре вынужден будет обратиться к уже имеющимся библиотекам стандартных программ и пакетам прикладных программ, поскольку он убедится, какой колоссальный объем интеллектуального коллективного труда уже вложен в них и что никто в одиночку не в состоянии пройти заново весь этот путь и разработать полностью математическое обеспечение ЭВМ.

Для начала введем в память ПМ-ЭВМ программу:

```
014000076                MVIA, 002Q
014001 002
014 002 306                 ADI, 003Q
014003 003
014 004 323                 OUT, OOOQ
014005 000
014 006 166                 HLT
```

Программа выполняет сложение двоичного числа (в данном случае 002Q), помещенного в ячейку ОЗУ по адресу 014Qt)01Q, с числом (в данном случае 003Q), помещенным в ячейку ОЗУ по адресу 014Q 003Q, и выводит результат (в данном случае 005Q) в порт 000. Для того чтобы эти действия были исполнены, достаточно после ввода программы нажать кнопку СБРОС, а затем кнопку П — на индикаторах порта 000 высветится двоичное число 00000101, эквивалентом которого является восьмеричное число 005 Q. Чтобы сложить другие числа в диапазоне от OOOQ до 377Q, их надо поместить по указанным выше адресам, не меняя остальной части программы, и снова нажать кнопки СБРОС и П. Для вычитания числа, помещенного по адресу 014Q 003Q, из числа, помещенного по адресу 014Q 001Q, достаточно поменять команду ADI (код операции 306Q) на команду SUI (код операции 326Q). Чтобы производить операции над десятичными числами, их надо представить в двоично-кодированном виде, так чтобы каждый байт изображал двухразрядное десятичное число от 0 до 99:

```
десятичное    двоично-кодированное восьмеричное десятичное
00            0000 0000              000
01            0000 0001              001
02            0000 0010              002
97            10010111              227
98            1001 1000              230
99            1001 1001              231
```

В приведенной выше программе для сложения, например, чисел 25 и 47 после команды сложения необходимо поставить команду десятичной коррекции результата сложения DAA:

```
014000076                MVIA.25D
014001 045
014002306                ADI.47D
014 003 107
014 004 047               DAA
```

```
014005323                   OUT, OOOQ
014 006 OOU
014 007 166                  HLT
```

В результате порт 000 после исполнения программы индицирует число 0111 0010B = 162Q=72D.

Для того чтобы проследить исполнение операции десятичной коррекции, введем в ПМ-ЭВМ следующую программу:

```
014000257                        XRA, A
014001 306              Ml:      ADI,001Q
014 002 001
014 003 047                       DAA
014 004 006                       MVI B, 040Q
014005 040
014006315               M2:       CALL DL
014007 277
014 010 000
014011005                         OCR B
014012302                         JNZ, M2
014013 006
014 014 014
014015323                         OUT, 002Q
014016 002
014017303                         JMP, Ml
014020  001
014021  014
```

Программа производит примерно каждые 0,5 с суммирование содержимого аккумулятора с единицей (инкремент), выполняет операцию десятичной коррекции и выводит результат в порт 002. Операция десятичной коррекции дает правильный результат только после операции сложения. Поэтому для выполнения вычитания одного двоично-кодированного десятичного числа из другого необходимо либо заменить вьиитание прибавлением к уменьшаемому вычитаемого, представленного в дополнительном коде, а затем уже производить десятичную коррекцию результата, либо применить эквивалентные этому искусственные приемы. Для примера введем в ПМ-ЭВМ программу вычитания содержимого ячейки 014Q 005Q (здесь 28) из содержимого ячейки 014Q 007Q (здесь 75) :

```
014000076                   MVIA, 99D
014001 231
014002306                   ADI, 01D
014 003 001
014004336                   SBI, 28D
014005 050
014 006 306                  ADI, 75D
014 007 165
014 010 047                  DAA
014011323                   OUT, OOOQ 014 012000
014013166                   HLT
```

В данной программе вычитание числа 28D из числа 99D + ID эквивалентно формированию в аккумуляторе вычитаемого в дополнительном коде. Поэтому прибавление уменьшаемого (здесь 75 D) с последующей десятичной коррекцией дает правильный результат (47 D), индицируемый портом 000.

Использование команд сложения и вычитания с переносом (заемом) позволяет организовать сложение и вычитание целых чисел неограниченной разрядности. Введем, например, в ПМ-ЭВМ программу

```
014 000 021                       LXID, ALPHA
014 001 000
014002015
014003041                         LXIH, BETA
014 004 000
014 005 016
014 006 016                       MVI C, 8H
014007 010
014010257                         XRAA
014011032               Ml:       LDAX D
014012216                         ADCM
014 013 000                       NOP
014014022                         STAXD
014015043                         INXH
014016023                         INXD
014017015                         OCR C
```

```
014020302                       JNZ, Ml
014021 Oil
014 022014
014 023 166                      HLT
```

С помощью этой программы двоичное 64-разрядное число, содержащееся в последовательных ячейках ЗУ, начиная (младшие разряды) с ячейки с символическим адресом ALPHA (здесь 015Q OOOQ), складывается с 64-разрядным числом, содержащимся в последовательных ячейках ЗУ, начиная с ячейки с символическим адресом BETA (здесь 016Q OOOQ), и результат помещается в ячейки ЗУ, начиная с ячейки с адресом ALPHA. Заменив команду NOP по адресу 014Q 013Q командой десятичной коррекции, получим программу сложения двух 16-разрядных десятичных чисел, находящихся в ячейках памяти, начиная с ячеек соответственно ALPHA и BETA.

Для вычитания десятичных 16-разрядных чисел, когда уменьшаемое располагается в ЗУ, начиная с ячейки ALPHA, а вычитаемое - начиная с ячейки BETA, получим такую программу:

```
014 000 021                     LXI D, ALPHA
010 001 000
000 102 015
000 003 041                     LXIH, BETA
000 004 000
014005 016
014 006 016                     MVIC, 8H
014 007 010
014 010 067                     STC
014011076               Ml:     MVI A, 99D
014 012231
014013316                       AC1,0
014014 000
014015226                       SUBM
014 016 353                      XCHG
014017206                       ADDM
014 020 047                      DAA
014021167                       MOVM,A
014 022 353                      XCHG
014 023 023                      INX D
014024043                       INXH
014025015                       DCRC
014026302                       JNZ, Ml
014027011
014030014
014031166                       HLT
```

Результат вычитания помещается в ячейки ЗУ, начиная с ALPHA. В процессе работы с помещенными выше программами мы смогли убедиться в больших неудобствах ввода десятичной информации с помощью восьмеричной клавиатуры, поскольку каждую пару десятичных цифр нужно предварительно перевести в восьмеричный код. Можно написать специальную программу ввода десятичных цифр, использовав тот факт, что при нажатии определенных клавиш вырабатываются не только коды чисел от 0 до 7, но и коды чисел от 8 до 15. Действительно, введя в ПМ-ЭВМ следующую программу:

```
014 000 315             Ml:     CALL, SKL
014001 177
014 002 000
014 003 323                     OUT, OOOQ
014 004 000
014005303                       JMP, Ml
014 006 000
014007 014
```

нажимая разные кнопки и наблюдая индикацию порта 000, убедимся, что при обращении к клавиатуре с помощью команды CALL SKL в аккумулятор помещаются коды в соответствии с табл. 7.7. В частности, кнопке С Б соответствует код 0000 1000, т. е. код числа 08 D, а кнопке МБ — код 0000 1001, т. е. код числа 09D. Следовательно, незначительно изменив текст программы-монитора в части анализа введенного числа, можно вводить и размещать в памяти ПМ-ЭВМ информацию непосредственно в десятичном виде. Предоставим читателю возможность самому составить такую программу и разместить ее в ОЗУ или в дополнительных микросхемах ПЗУ.

Знакомясь с литературой и технической документацией, читатель вскоре обнаружит, что большая часть стандартных программ оперирует не с десятичными числами, а с двоичными. Переход от десятичной системы к

двоичной и обратно осуществляется с помощью специальных программ при вводе и соответственно при выводе информации из ЭВМ.

Рассмотрим следующую программу, переводящую пятиразрядное целое десятичное число в диапазоне от 00000 до 65535, вводимое с клавиатуры ПМ-ЭВМ (для цифры 8 используется кнопка СБ, а для цифры 9 — кнопка МБ), в 16-разрядное двоичное число, содержащееся в регистровой паре H:

```
014000041                       LXI H, OOOQ OOOQ
014001 000
014 002 000
014003016                       MVIC, 0050
014 004 005
014005315               Ml:     CALL SKL
014006 177
014 007 000
014010376                       CPI, 012Q
014 Oil 012
014012322                       JNC, Ml
014013  005
014014  014
014015345                       PUSHH
014016321                       POPD
014 017051                      DADH
014 020051                      DADH
014 021031                      DADD
014 022051                      DADH
014 023 137                     MOVE, A
014 024 026                     MVI D, OOOQ
014 025 000
014 026031                      DADD
014 027015                      OCR C
014 030302                      JNZ,M1
014 031 005
014 032014
014 033 174                     MOV A, H
014 034323                      OUT, 001Q
014 035 001
014036175                       MOV A, L
014037323                       OUT, OOOQ
014 040000
014041 166                      HLT
```

Программа обращается к ПП опроса клавиатуры SKL и при нажатии любой кнопки анализирует введенный код. Если код соответствует цифрам от 0 до 9, содержащееся в регистровой паре H число (команды с адреса 014Q 017Q по 014Q 022Q) умножается на 10 и к нему прибавляется вновь введенное число. Программа с помощью счетчика, организованного на регистре C, отсчитывает пять таких циклов, поэтому вводимое число обязательно должно быть пятиразрядным (например, 00512). Команды с адреса 014Q 033Q используются в демонстрационных целях для вывода преобразованного в двоичный код числа в порты 001 и 000. Если программа должна использоваться в качестве ПП, то по адресу 014Q 033Q должна быть записана команда RET.

Введите программу в ПМ-ЭВМ, нажмите кнопки СБРОС и П, затем введите десятичное число 10 000, последовательно нажимая кнопки 1, О, О, О, О. В портах 001 и 000 высветится число 0010 0111 0001 0000, являющееся двоичным эквивалентом десятичного числа 10 000. Проверьте правильность работы программы, вводя другие десятичные числа, например 00512, 01024, 02048, 04096, 08192, 16384, 32768, 65535: Какие двоичные числа должны индицироваться портами 001 и 000?

Рассмотрим теперь программу преобразования целого двоичного 16-разрядного числа, содержащегося в регистровой паре D, в двоично-кодированное десятичное число, каждый разряд которого помещается, начиная с младшего, в последовательные ячейки ОЗУ, начиная с ячейки, символический адрес которой ED (машинный адрес 015Q OOOQ):

```
014000041                       LXI H, ED
014001  000
014002 015
014003016                       MVIC,005Q
014 004 005
014005066               Ml:     MVI M, OOOQ
014 006 000
014007043                       INXH
014010015                       OCR C
014011302                       JNZ,M1
014012005
014013 014
```

```
014014053                       DCXH
014015001                       LXI B. 33003600
014 016 360
014017 330
014020315                       CALLCF
014 021 060
014022014
014023001                       LXI B, 374Q030Q
014024030
014025 374
014026315                      CALLCF
014027 060
014 030014
014031001                       LXI B, 377Q234Q
014 032 234
014 033 377
014034315                       CALLCF
014 035 060
014036 014
014037001                       LXI B, 377Q366Q
014040366
014 041 377
014042315                       CALLCF
014 043 060
014 044 014
014 045 163                      MOVM, E
014 046 166                      HLT
014 047 021                      LXI D (число)
014050377
014051 377
014052166                       HLT
014060345               CF:      PUSH H
014 061 353                      XCHG
014062011                       DAD B
014063322                      JNC, M2
014 064 074
014065 014
014 066 353                      XCHG
014067341                       POPH
014 070 064                      INK M
014071303                       JMP, CF
014072060
014073 014
014 074 171             M2:       MOV A C
014 075 057                      CMA
014 076 137                      MOVE, A
014077170                       MOV A, B
014100057                       CMA
014 101 127                      MOVD, A
014 102023                       INXD
014103031                       DADD
014104353                       XCHG
014105341                       POPH
014 106 053                      OCX H
014107311                       RET
```

Программа построена на принципе последовательного исчерпывания: каждый десятичный разряд, начиная с десятков тысяч, определяется путем подсчета количества суммирований с преобразуемым числом двоичного числа, соответствующего дополнению до 10 000, затем до 1000, затем до 100, затем до 10, каждый раз до появления переноса. Вычисленные значения засылаются в ячейки соответственно ED + 4, ED + 3, ED + 2, ED + 1. В остатке остаются единицы, засылаемые в ячейку памяти ED.

Если программа должна использоваться в качестве ПП, то по адресу 014Q 046Q должна быть записана команда RET. В данном примере для загрузки регистровой пары D использована небольшая программа, начинающаяся с адреса 014Q 047Q.

Введите программу в ПМ-ЭВМ, нажмите кнопку СБРОС, наберите число 047 и нажмите кнопки МБ и П. Регистровая пара D загружена максимальным двоичным числом (во всех разрядах единицы). Теперь нажмите кнопки СБРОС и П, затем нажмите кнопку СБРОС, наберите число 015 и нажмите кнопку СБ. Порт 002 будет индицировать число 0000 0101, т. е. двоично-десятичный код числа 05. Нажмите кнопку И - порт 002 будет

индицировать 0000 ООН, т. е. 03. Последовательным нажатием кнопки И получим соответственно 05, 05, 06. Это означает, что в ячейках ОЗУ хранится число 65 535, являющееся десятичным эквивалентом максимального 16-разрядного двоичного числа. Загружайте в ячейки 014Q 050Q и 014Q 051Q различные числа. Повторяя описанную выше процедуру, можно убедиться, что преобразование осуществляется правильно.

Чтобы реализовать на ПМ-ЭВМ программируемый калькулятор, выполняющий четыре арифметические действия, приведенные выше программы необходимо дополнить программами умножения и деления (более сложные программы, в том числе программы вычисления элементарных функций, можно найти в справочной литературе, здесь они не приводятся за недостатком места).

9.2. ПРОГРАММИРУЕМОЕ УПРАВЛЯЮЩЕЕ УСТРОЙСТВО

Устройства цифровой автоматики являются областью наиболее массового применения микропроцессоров. В качестве примера конструирования управляю-

щего устройства на базе ПМ-ЭВМ рассмотрим кодовый замок, предназначенный для использования в жилом помещении или в других целях. Поставим задачу так, чтобы использовать минимум добавочного оборудования. Введем в ПМ-ЭВМ такую программу:

```
014 000 006                          MVI B, 006Q
014 001 006
014 002 041                          LXI H, (адр. кода)
014 003 040
014 004 014
014005315              MI:     CALL SKL
014006 177
014 007 000
014010276                      CMPM
014011 302                      JNZ ,M3
014012026
014013 014
014014043                      INXH
014015005                      DCRB
014016302                      JNZ, MI
014017005
014020014
014021076              M2:     MVI A, 001Q
014 022 001
014 023 323                          OUT, OOOQ
014 024 000
014025166                      HLT
014026074              M3:     INRA
014027323                      OUT, 001Q
014 030 001
014031315                      CALLDL
014 032 277
014 033 000
014 034 303                      JMP, M3
014 035 026
014036014
014040001              КОД:    123481
014041 002
014 042 003
014 043 004
014 044 010
014 045 001
```

Здесь с помощью команды MVI B организован счетчик количества цифр в наборе кодового замка. Командой CALL SKL непрерывно опрашивается клавиатура, при нажатии одной из кнопок полученный код сравнивается с кодом, хранящимся в ячейках памяти. Если коды не совпадают, происходит переход по метке М3 к части программы, предназначенной для подачи сигнала тревоги (в порт 001 выводится инкрементируе-мое через каждый 0,01 с содержимое аккумулятора). Если коды совпадают, программа возвращается к опросу клавиатуры (метка Ml) и ожидает нового нажатия кнопки, при котором происходит сравнение полученного кода с содержимым следующей ячейки памяти, и т. д. до тех пор, пока количество нажатий не будет равно запрограммированному командой MVI B. Тогда в порт 000 будет выведено число 001Q, т. е. загорится светодиодный индикатор, соответствующий младшему разряду. Сигнал с младшего разряда порта 000 может быть использован для управления электромагнитом, отпирающим замок.

Введем программу в ПМ-ЭВМ, нажмем кнопки СБРОС и П, а затем поочередно кнопки 1, 2, 3, 4, СБ, 1, соответствующие кодовой комбинации замка (кнопка СБ соответствует цифре 8, а кнопка МБ - цифре 9). При нажатии последней кнопки на индикаторах порта 000 появится число 001Q. Для возврата к исходному состоянию необходимо снова нажать кнопки СБРОС и П. Теперь при наборе кода сделайте

ошибку - тотчас в порте 001 начинают мигать светодиодные индикаторы, а если между выходом порта 001 (вывод 16 микросхемы D18) и источником + 5 В через резистор 200 Ом подключить наушник или репродуктор, послышится тревожное гудение, прекратить которое можно, только нажав кнопку СБРОС.

Вводя разные числа в ячейку ОЗУ по адресу 014Q 001Q, можно изменять количество цифр в кодовой комбинации, а вводя различные числа в ячейки ОЗУ, начиная с 014Q 040Q -задавать различные кодовые комбинации. Таким образом, не используя никакого добавочного оборудования, мы промоделировали работу кодового замка. Для реализации кодового замка, как говорят, "в железе" необходимо вынести в наборное поле замка кнопки, параллельные кнопкам клавиатуры 0-7, СБ, МБ, СБРОС, П. Последние две кнопки необходимы, чтобы при неправильном наборе вернуться к исходному состоянию. Кроме того, нужно организовать управление механизмом замка, а для того чтобы после случайного прекращения питания замок правильно функционировал, программу вместе с кодовой комбинацией необходимо хранить в ПЗУ.

Мы видим, что при всей простоте реализации кодовый замок на базе ПМ-ЭВМ представляет законченную систему цифрового управления: он хранит программу и исходные данные (уставки), принимает входные управляющие воздействия (нажатия кнопок), вырабатывает выходные управляющие сигналы (сигнал тревоги или сигнал разрешения для механизма замка). Читатель может по своему усмотрению усложнить алгоритм работы замка с целью повышения безопасности или с целью упрощения коммуникаций.

9.3. СБОР И ОБРАБОТКА ДАННЫХ

Системы сбора и обработки данных, как правило, являются частью иерархических систем управления или коммуникации и предназначены для приема данных, представленных в виде аналоговых или цифровых сигналов различных уровней и форматов, преобразования их в стандартную цифровую форму, обработки в соответствии с заданным алгоритмом и выдачи обработанных данных на индикационные устройства или в старшую иерархическую ступень системы.

В качестве примера системы сбора и обработки данных рассмотрим конструирование на базе ПМ-ЭВМ счетчика случайных событий. Введем в ПМ-ЭВМ программу

```
014 000 006                    MVI B, OOO.Q
014001 000
014002016                      MVIC, OOOQ
014 003 000
014 004 026                    MVI D, OOOQ
014 005 000
014006315              Ml:     CALL SKL
014 007 177
014 010000
014 Oil 170                    MOV A, B
014012074                      INR A
014 013 047                    DAA
014014323                      OUT, OOOQ
014 015 000
014 016 107                    MOV B, A
014017171                      MOV A, C
014020316                      ACI, 0000
014021 000
014 022 047                    DAA
014023323                      OUT. 0010
014024  001
014025  117                    MOVC, A
014026  172                    MOV A, D
014027316                      ACI, OOOQ
014030000
014031 047                     DAA
014032323                      OUT, 002Q
014 033 002
014034127                      MOVD. A
014035303                      JMP, Ml
014 036 006
014037014
```

После ввода программы нажмем кнопки СБРОС и П. В портах 002, 001, 000 индицируются соответственно 006Q - код первой команды программы и 014Q OOOQ — ее адрес в памяти. Теперь будем нажимать любую кнопку клавиатуры (кроме кнопки СБРОС) - в портах 002, 001, 000 индицируется количество нажатий кнопки, выраженное в двоично-десятичном коде, начиная с 000 001D и кончая 999 999D. Для обнуления счетчика достаточно нажать кнопки СБРОС и П. Таким образом, чисто программным путем нами реализована

простейшая система сбора и обработки информации, регистрирующая нажатия кнопок, подсчитывающая эти нажатия и выдающая на индикацию в двоично-десятичной форме число, равное количеству нажатий.

9.4. РЕАЛИЗАЦИЯ ДИАЛОГОВОГО РЕЖИМА

Диалоговый режим реализует

взаимодействие человека с вычислительной машиной. Наиболее употребительными техническими средствами, обеспечивающими обмен информацией между человеком и машиной в процессе диалога, являются клавишные устройства и устройства визуального отображения (дисплеи). Поскольку ПМ-ЭВМ имеет, хотя и ограниченные по своим возможностям, клавиатуру и светодиодную индикацию, можно попытаться организовать на ней диалоговый режим. Введем в ПМ-ЭВМ следующую программу, предназначенную для оценки реакции оператора в диалоговом режиме:

```
014000041          Ml: LXI H, TAB
014001  110
014002  014
014003257          M2: XRA A
014 004 323              OUT, OOOQ
014005 000
014006323               OUT.001Q
014007 001
014010323               OUT, 003Q
014011 003
014012107               MOV B, A
014 013 117              MOVC, A
014 014 057              CMA
014015323               OUT, 002Q
014 016 002
014017176               MOV A, M
014 020 267              ORA A
014021312               JZ,M1
014 022 000
014023014
0140243-15          M3: CALL DL
014025 277
014 026 000
014027075               DCRA
014030302               JNZ, M3
014031 024
014 032014
014033323               OUT, 002Q
014034 002
014035333          M4: IN, 003Q
014 036 003
014037057               CMA
014 040 267              ORA A
014041312               JZ.M6
014042061
014 043 014
014044171               MOV A, C
014045323               OUT, OOOQ
014 046 000
014047170               MOV A, B
014050323               OUT, 001Q
014051 001
014052333          M5: IN, 003Q
014 053 003
014 054 057              CMA
014055267               ORA A
014056302               JNZ,M5
014 057 052
014 060014
014061315          M6: CALLDL
014062277
014 063 000
014064171               MOV A, C
014 065 074              INR A
```

```
014 066 047          DAA
014067117            MOVC, A
014070170            MOV A, B
014071316            ACI, OOOQ
014 072 000
014073047            DAA
014074107            MOVB.A
014075376            CPI,005Q
014076005
014077332            JC, M4
014 100035
014 101 014
014102043            INXH
014 103 303          JMP, M2
014 104 003
014 105 014
014110377            TAB:
014 111 200
014 112300
014 113 330
014 114000
```

После ввода программы нажмем кнопки СБРОС и П. Порт 002 высветит число 377Q (все единицы), а индикаторы портов 001 и 000 погашены. Через неизвестный оператору интервал времени индикаторы порта 002 гаснут. В этот момент оператор должен нажать любую (кроме СБРОС) кнопку клавиатуры. В портах 001 и 000 высветится выраженное в двоично-десятичном коде время задержки между погасанием индикаторов порта 002 и нажатием кнопки. Порт 001 индицирует десятки и единицы секунд (считая слева), а порт 000 — десятые и сотые доли секунды. После отпускания кнопки через заданный интервал времени индикаторы портов 001 и 000 гаснут, а индикаторы порта 002 снова загораются, чтобы погаснуть через неизвестный интервал времени. Если ни одна кнопка не нажата после погасания индикаторов порта 002, через заданный интервал времени они снова загорятся. Если кнопка нажата до погасания индикаторов, то портами 001 и 000 индицируется нулевая задержка, т. е. не загорается ни один индикатор.

Оценки времени реакции оператора (в секундах):

0,00 — 0,09 — ошибочное значение, Вы поторопились, попробуйте снова;
0,10 — 0,14 — исключительная, почти фантастическая реакция;
0,15 — 0,19 — очень хорошая;
0,20 — 0,24 -нормальная;
0,25 — 0,29 -посредственная;
0,30 — 0,34 — плохая;
0,35 — 5,00 - Вы, вероятно, уснули.

Программа может быть использована как для оценки, так и для тренировки реакции. Время задержки гашения индикаторов порта 002 задается количеством циклов задержки, начинающихся с метки МЗ. В аккумулятор при каждом замере заносятся последовательно числа из таблицы, хранящейся в ячейках памяти, начиная с адреса, задаваемого командой LXIH. Длина таблицы практически неограниченная. Если в таблице помещено число OOOQ, то при его занесении в аккумулятор на очередном шаге происходит возврат к началу программы и, следовательно, к началу таблицы. Цена младшего разряда, задающего задержку числа, равна задержке, вырабатываемой ПП DL, т. е. 10 мс. После того как индикаторы порта 002 погашены (команда по адресу 014Q 033Q) и до нажатия кнопки с помощью счетчика, организованного на регистровой паре В, подсчитывается количество циклов обращения к ПП DL (команды, начиная с метки Мб). В счетчиках используется команда DAA, поэтому результат получается в двоично-десятичном коде. При нажатии кнопки содержимое регистровой пары В выводится в порты 001 и 000. Счетчик циклов снова задействуется после отпускания кнопки, и после достижения заданного числа циклов происходит возврат к метке М2 программы, при этом регистровая пара Н инкрементируется, в результате чего при следующем замере в качестве параметра задержки извлекается следующее число из таблицы. Параметры, определяющие работу программы, помещаются по следующим адресам: 014Q 001Q, 014Q 002Q — адрес начала таблицы; 014Q 076Q — время, в течение которого индикаторы порта 002 погашены; цена младшего разряда — около 1 с.

Обратим внимание на то, что для ввода информации с клавиатуры не использована ПП SKL, так как при обращении к ней невозможно организовать счет времени (до нажатия кнопки процессор не выходит из ПП).

10

РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ПМ-ЭВМ

10.1. КЛАВИАТУРА И ИНДИКАЦИЯ

В предыдущей главе мы познакомились с различными способами применения ПМ-ЭВМ. Было показано, что, не используя никакого дополнительного оборудования, можно реализовать широкий круг практических задач. Читатель, без сомнения, сумеет найти и другие возможности применения ПМ-ЭВМ.

Нетрудно догадаться, что производительность и вычислительная мощность ПМ-ЭВМ могут быть увеличены путем последовательного наращивания внутренней и внешней памяти, устройств ввода/вывода и периферийного оборудования. Читателю предстоит самостоятельно оценить, насколько ПМ-ЭВМ удовлетворяет его потребности, и нужно ли ее усложнять, если стоящие задачи могут быть решены простыми средствами. Всегда следует иметь в виду, что время и трудоемкость разработки программного обеспечения обычно являются решающими факторами в процессе создания систем на базе МП.

ПМ-ЭВМ может быть использована в учебных целях, в качестве устройства отладки программ для вновь разрабатываемых МП-систем, в качестве макетного устройства для отработки аппаратной части, в качестве самостоятельной вычислительной машины и как управляющая ЭВМ. Как рабочий инструмент ПМ-ЭВМ имеет ряд очевидных недостатков: информация в нее вводится только с клавиатуры и только в цифровой форме, двоичная индикация трудна для чтения и интерпретации, при выключении питания теряется записанная в ОЗУ информация, так что при включении надо заново вводить программы и исходные данные.

Заметим, что некоторые из этих недостатков легко устранимы. В частности, нарастив порты ввода и вывода 003, обслуживающие клавиатуру до полных восьми разрядов, получим контактную сетку 8x8, образующую клавиатуру из 64 кнопок, достаточную для ввода алфавитно-цифровой информации. В связи с этим потребуется переделать программу-монитор и использовать специальные программы для обработки алфавитно-цифровой информации.

Для вывода информации можно нарастить порты 002, 001, 000 схемами преобразования двоичного кода в семисегмент-ный (рис. 10.1), при этом индикация на семисегментных индикаторах будет представлена восьмеричными или же десятичными числами. Из тех же портов вывода можно организовать динамическую систему индикации, обеспечивающую вывод любых знаков (цифр, значков и некоторых букв как русского, так и латинского алфавитов) на дисплейное устройство, содержащее до 16 семисегментных индикаторов (рис. 10.2). Программы для индикации обычно включают в состав программы-монитора. В целях изучения способов обслуживания устройств динамической индикации введем в ПМ-ЭВМ следующую программу:

```
014000001          Ml: LXI B, TAB
014 001 060
014002 014
014003041                LXIH, OOOQ001Q
014 004 001
014 005 000
014006012          M2: LDAX B
014007323                OUT, 002Q
014010002
014011 174                MOV A, H
014012323                OUT, 001Q
014 013 001
014014175                MOV A, L
014015323                OUT, OOOQ
014016 000
014017003                INX B
014 020 026               MVI D, 100Q
014021 100
014022315          M3: CALL DL
014023 277
014 024 000
```

```
014025025                       DCRD
014 026 302                      JNZ, M3
014 027022
014030014
014031051                       DADH
014032322                       JNC.M2
014033 006
014 034 014
014035303                       JMP, Ml
014 036 000
014037 014
014 060 000             TAB:
014061 001
014 062002
014063 003
014 064 004
014 065 005
014 066 006
014067 007
014 070010
014071 O11
014 072012
014073 013
014 074 014
014 075 015
014 076 016
014077017
```

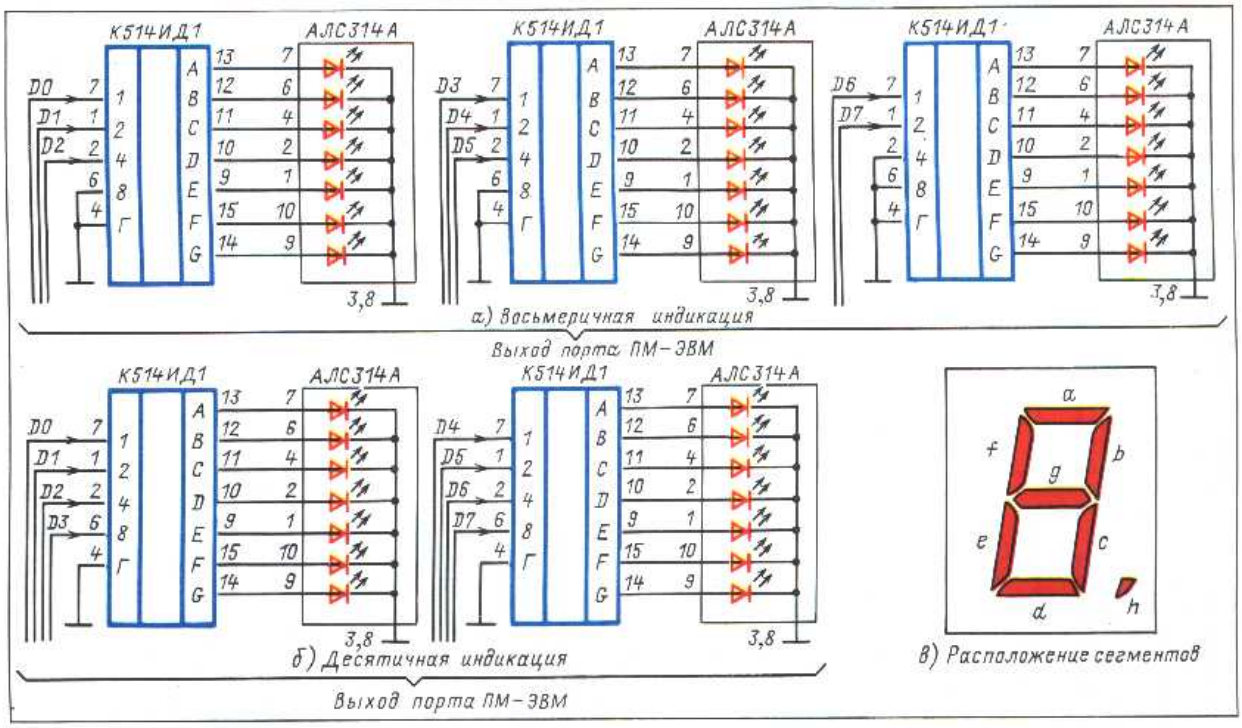


Рис. 10.1. Схемы преобразования двоичного кода в семисегментный

Нажмем кнопки СБРОС и П. В портах 000 и 001 погаснут все индикаторы, за исключением младшего разряда порта 000. Через полсекунды загорится индикатор следующего разряда, а предыдущий погаснет и т. д., пока не загорится индикатор старшего разряда порта 001. При этом каждому зажженному индикатору соответствует код го таблицы TAB, индицируемый портом 002. Цикл сканирования повторяется до тех пор, пока не будет нажата кнопка СБРОС.

Эта программа демонстрационная. Она позволяет отладить схему, изображенную на рис. 10.2, устанавливая рабочий ток индикаторов подбором резисторов и задавая время сканирования значением параметра, хранимого в регистре D (ячейка 014Q 021Q). В программе используются почти все регистры МП, поэтому при применении ее в качестве стандартной ПП вывода на индикацию следует вначале поместить команды PUSH PSW, PUSH B, PUSH D, PUSH H, а в конце программы — команды XRA A,

OUT OOOQ, OUT 001Q, OUT 002Q, POP H, POP D, POP B, POP PSW, RET, исключив команду CALL DL вызова ПП временной задержки и команду возврата к началу программы JMP, Ml.

В соответствии со схемой рис. 10.2 каждому сегменту семи-сегментного индикатора соответствует определенное число, выводимое через порт 002: сегменту *a* - 0000 0001 (001Q), *b* - 0000 0010 (002Q), *c*-0000 0100 (0040), d-0000 1000 (010Q), *e* - 0001 0000 (020Q), f - 0010 0000 (040Q), £- 0100 0000(1000), *h* - 1000 0000 (200Q). Следовательно, изображаемый индикатором символ (цифра, буква или другой знак) представляется восьмиразрядным двоичным числом, являющимся суммой чисел, соответствующих горящим сегментам индикатора. Например, десятичные числа от 0 до 9 представляются следующими числами: 0-0011 1111 (077Q), 1 - 0000 OHO (006Q), 2 -0101 1011 (133Q), 3 - 0100 1111 (117Q), 4 - 01100110(1460), 5 - OHO 1101 (155Q), 6 - 0111 1101 (175Q), 7 - 0000 0111 (007Q), 8 - 0111 1111 (177Q), 9 - OHO 1111 (157Q). С определенными условностями могут быть изображены также буквы латинского и русского алфавитов и различные знаки.

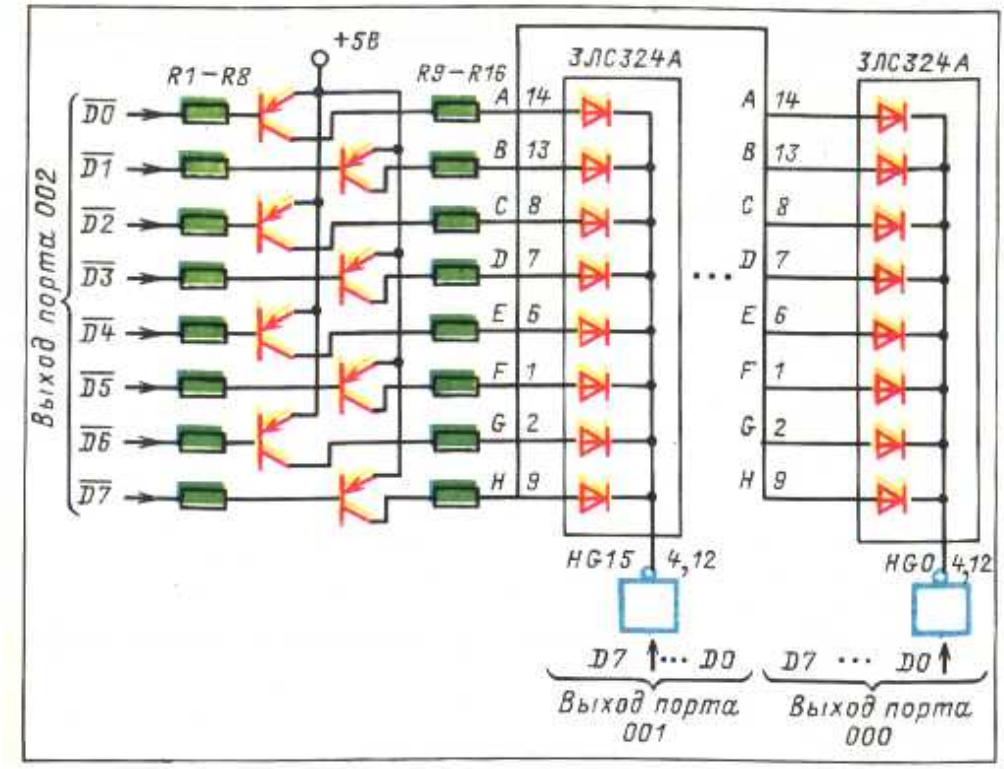


Рис. 10.2. Дисплейное устройство из 16 семисегментных индикаторов

Подключим к ПМ-ЭВМ схему, изображенную на рис. 10.2, и введем следующую программу:

```
014000001          Ml: LXI B, TAB
014 001 060
014 002 014
014 003 305              PUSH B
014 004 036              MVI E, 040Q
014 005 040
014006301          M2: POP B
014007041              LXIH, OOOQ001Q
014010001
014011 000
014012012          M3: LDAX B
014 013 323              OUT, 0020
014014002
014015 174               MOV A, H
014016323              OUT, 001Q
014017 001
014020175              MOV A, L
014021323              OUT, OOOQ
014022000
014 023 003              INX B
014 024 026              MVI D, 2000
014 025 200
014026025          M4: OCR D
014027302              JNZ, M4
```

```
014 030026
014031 014
014032051                       DADH
014033322                       JNC, M3
014034  012
014035  014
014036035                       DCRE
014037302                       JNZ,M2
014 040 006
014041 014
014 042 073                      OCX SP
014 043 073                      DCX SP
014 044 301                      POP B
014 045 003                      INX B
014 046 305                      PUSH B
014047012                       LDAX B
014 050 376                      CPI, 060Q
014 051 060
014052312                       JZ, Ml
014 053 000
014 054 014
014055303                       JMP, M2
014 056 006
014057 014
014 060 077             TAB:
014 061 006
014062133
014063  117
014064  146
014 065 155
014 066 175
014 067 007
014070177
014071 157
014 072 000
014 073 000
014 074 060
```

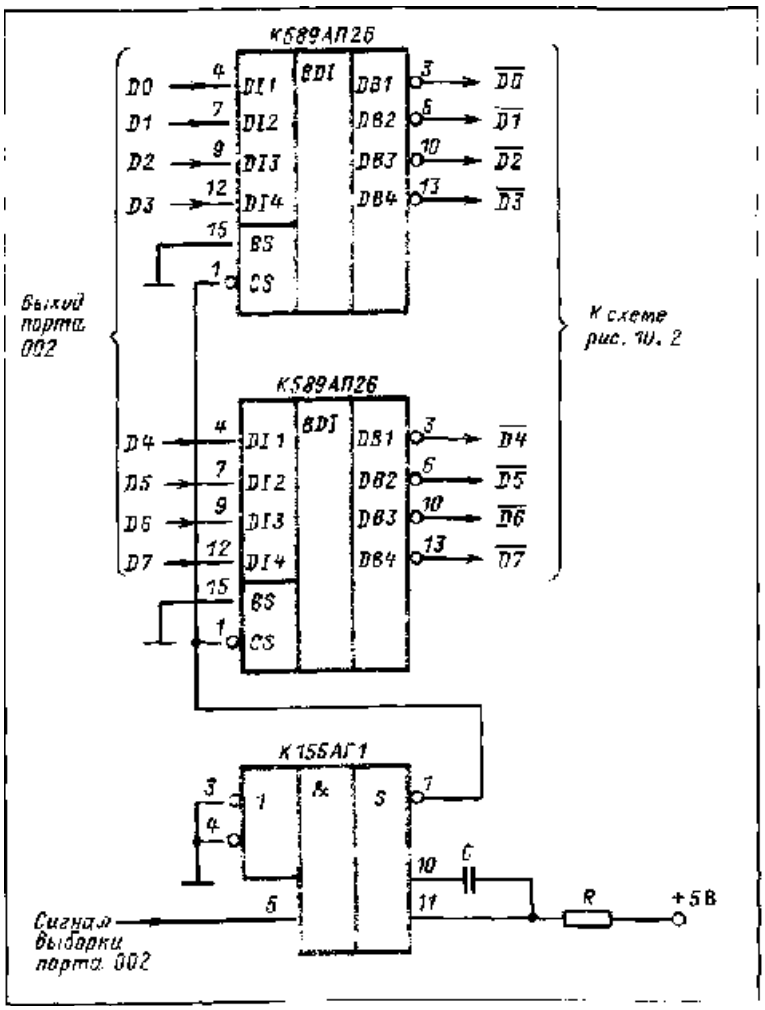


Рис. 10.3. Схема гашения индикаторов

Программа, построенная на базе предыдущей, выводит на дисплей, образованный 16 семисегментными индикаторами, движущийся текст (в данном примере цифры от 0 до 9 и два пробела), хранимый в ОЗУ, начиная с ячейки 014Q 060Q. Конец текста помечается символом, не использованным для обозначения букв и цифр (в данном примере символ II, его код 060Q). Он хранится в ячейке 014Q 051Q и сравнивается с символом текста, выводимым на крайний левый индикатор дисплея. Как только опознан символ конца текста, программа возвращается к началу текста. Время цикла сканирования задается числом в ячейке 014Q 025Q, а время, на которое текст останавливается, — числом в ячейке 014Q 005Q.

Программа может быть использована для подготовки и редактирования текстов, выводимых на внешние устройства.

Во избежание перегорания индикаторов при непредусмотренном прекращении сканирования схему, изображенную на рис. 10.2, следует дополнить устройством гашения индикаторов, построенным на основе одновибратора (рис. 10.3). Устройство отключает индикацию через заданный цепочкой RC интервал времени после подачи импульса выборки порта 002. Таким образом, индикаторы непрерывно (для глаза наблюдателя) горят только при периодической выдаче информации в порт 002.

Рассмотренная схема индикации весьма универсальна и обладает гЪраздо более широкими возможностями для обеспечения диалогового режима работы ПМ-ЭВМ по сравнению с простой двоичной индикацией.

10.2. ВНЕШНЯЯ ПАМЯТЬ

Постепенно осваивая методы программирования и приемы работы на ПМ-ЭВМ, читатель рано или поздно столкнется с необходимостью, во-первых, увеличения объема памяти для размещения все более длинных программ и, во-вторых, сохранения уже отработанных программ в памяти при многократном включении и выключении питания. Поскольку ПМ-ЭВМ построена в соответствии со стандартной архитектурой вычислительных систем на базе МП типа КР580ИК80А, ее прямо адресуемая память может быть легко расширена до максимального объема 64 Кбайта.

Непосредственно на плате ПМ-ЭВМ могут быть дополнительно установлены микросхемы памяти на 2 Кбайта. При этом не потребуется никакого дополнительного оборудования, так как для выдачи сигналов выборки микросхем ОЗУ используются ранее не задействованные выводы дешифратора К155ИД4 (микросхема D11).

Память большего объема потребует для своего размещения дополнительной платы, на которой должны быть установлены свои адресные дешифраторы и другие микросхемы (кроме микросхем памяти), обеспечивающие нормальную работу устройства. В этом случае на плате ПМ-ЭВМ необходимо установить шинные буферы-формирователи для умощнения шин адреса и данных (например, на микросхемах типа К589АП16), причем буферы шины данных должны быть двунаправленными. Аналогично внутреннему буферу шины данных (микросхемы D9 и D10) для управления направлением передачи данных используется буферированный сигнал DBIN.

Для формирования энергонезависимой части памяти могут быть применены микросхемы ПЗУ с однократной записью информации, например типа КР556РТ4, которые уже использованы в ПМ-ЭВМ для хранения программы-монитора. Более удобными могут оказаться микросхемы ППЗУ с электрической записью и ультрафиолетовым стиранием типов К573РФ1, К573РФ2, К573РФ4, К573РФ5. Наконец, могут быть использованы микросхемы ППЗУ с электрической записью и стиранием информации, например типов КР558РР1 и КР558РР11.

Для сохранения информации могут быть использованы внешние носители, такие, например, как перфолента или магнитная лента. В этом случае поступают следующим образом: с помощью специальной программы вывода содержащуюся в ОЗУ информацию передают побайтно (параллельный вывод) или побитно (последовательный вывод) через устройство вывода на перфоратор или на магнитофон. Записанная на внешнем носителе информация может храниться неограниченно долго. При необходимости восстановить в ОЗУ ранее выведенную информацию она считывается с помощью перфосчитывающего устройства или магнитофона и вводится через устройство ввода в ОЗУ микро-ЭВМ под управлением программы ввода. Подчеркнем, что информация (данные, программы), считываемая из внешнего накопителя, обычно не используется непосредственно при исполнении рабочей программы, а сначала помещается в виде одного или нескольких массивов в ОЗУ, а затем уже используется в работе.

10.3. НАКОПИТЕЛЬ НА БАЗЕ БЫТОВОГО МАГНИТОФОНА

Сохранение программ на простом бытовом магнитофоне не только эффективно, но и необходимо для записи более длинных программ без использования дорогих цифровых магнитофонов, а также взамен записи на перфоленту. Дешевые бытовые магнитофоны имеют ограниченную полосу пропускания, высокий уровень шумов и непостоянную скорость протяжки. В отличие от принятой в цифровых магнитофонах схемы, где записываемый сигнал с помощью специального устройства модулируется разными частотами для логического 0 и логической 1, в описываемом здесь методе записи требования, предъявляемые к каналу передачи данных, минимальны. Вместо передачи уровней, соответствующих логическому 0 и логической 1, данные передаются в виде пачек импульсов заданной частоты (тона) с разным количеством импульсов в пачке. Основная частота должна лежать в пределах полосы пропускания магнитофона. Каждый бит передаваемой информации представляется в виде импульсной пачки, за которой следует пауза. Первая треть периода передачи бита всегда занята импульсами, во второй трети либо импульсы продолжаются, либо наступает пауза, что означает соответственно логическую 1 или логический 0, последняя треть периода — всегда пауза (рис. 10.4). Таким образом, значение бита данных задается отношением длины импульсной пачки к длине паузы. (Известны и другие системы записи, например основанные на количестве импульсов в пачке.) Для изучения методов формирования таких импульсных пачек введем в ПМ-ЭВМ следующую учебную программу:

```
014000016                    MVIC, 252Q
014001 252
014002006              Ml: MVIB, 011Q
014003 Oil
014004257              M2: XRA A
014 005 323                   OUT, OOOQ
014 006 000
014007315                    CALL IMP
014010060
014011 014
014012171                    MOV A, C
014013037                    RAR
014014117                    MOV C, A
014015322                    JNC, M3
014 016 026
014017014
```

```
014 020 257                    XRA A
014 021 323                    OUT, OOOQ
014 022 000
014 023 303                    JMP, M4
014 024 032
014025 014
014026076              M3: MVI A, 377Q
014027377
014 030 323                    OUT, OOOQ
014031 000
014032315              M4: CALL, IMP
014 033 060
014 034 014
014035257 .                XRA A
014 036 057                CMA
014 037 323                     OUT, OOOQ
014 040 000
014041315                  CALL, IMP
014 042 060
014043 014
014044171                  MOV A, C
014 045 323                OUT, 002Q
014 046 002
014 047 005                DCR B
014050302                  JNZ,M2
014 051 004
014052014
014 053 166                HLT
014 060 026             IMP: MVI D, 010Q
014061 010
014062257                  XRA A
014063323              M5: OUT, 001Q
014 064 001
014 065 036                MVI E, 040Q
014 066 040
014067315              M6: CALL DL
014 070 277
014071 000
014072035                  DCRE
014073302                  JNZ.M6
014 074 067
014075 014
014076074                  INK A
014 077 025                OCR D
014100302                  JNZ,M5
014 101 063
014 102 014
014103311                  RET
```

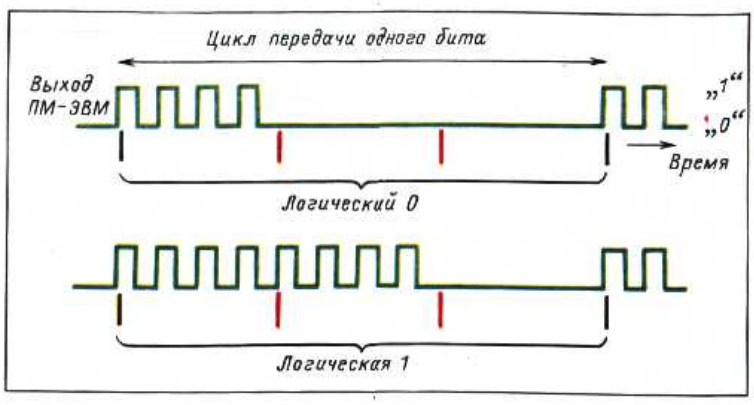


Рис. 10.4. Диаграмма передачи одного двоичного разряда

Программа, начиная с младшего разряда, преобразует байт информации, хранящийся в регистре С, в последовательность посылок, наблюдаемую в портах 000 и 001. Индикаторы порта 000 погашены в тех частях периода передачи каждого бита, когда должна выдаваться импульсная пачка, и горят, когда она не

должна выдаваться, т. е. логическому 0 соответствует последовательность: не горит — горит — горит, а логической 1 — не горит — не горит — горит. Импульсы с заданной частотой следования появляются в младшем разряде порта 001, а в целом индикаторы порта 001 подсчитывают эти импульсы. Частота следования импульсов — примерно 2 Гц. Порт 002 позволяет наблюдать преобразуемое число, каждый разряд которого последовательно анализируется программой путем сдвига в разряд переноса. В ячейке 014Q 003Q хранится параметр, заносимый в регистр B, задающий количество преобразуемых разрядов числа. Здесь этот параметр равен девяти, так как кроме восьми информационных пачек для обозначения конца передачи числа передается импульсная пачка, соответствующая нулю. Подпрограмма формирования импульсной пачки имеет метку IMP. Количество импульсов в пачке равно половине числа, хранящегося в ячейке 014Q 061Q, которое обязательно должно быть четным. Длительность импульса в пачке определяется задержкой, задаваемой числом в ячейке 014Q 066Q. В учебной программе длительность импульса для наблюдаемости задана равной произведению этого числа на задержку, определяемую ПП DL. Для использования ГШ в качестве рабочей необходимо из ПП IMP изъять обращение к ПП DL, из основного текста изъять команду вывода в порт 002 (адреса с 014Q 044Q по 014Q 046Q), а вместо команды HLT ввести команду RET. Чтобы сформировать сигнал, который можно подавать непосредственно на вход записи магнитофона, используем схему, представленную на рис. 10.5. Эта схема позволяет регулировать уровень записи и ликвидирует постоянную составляющую сигнала.

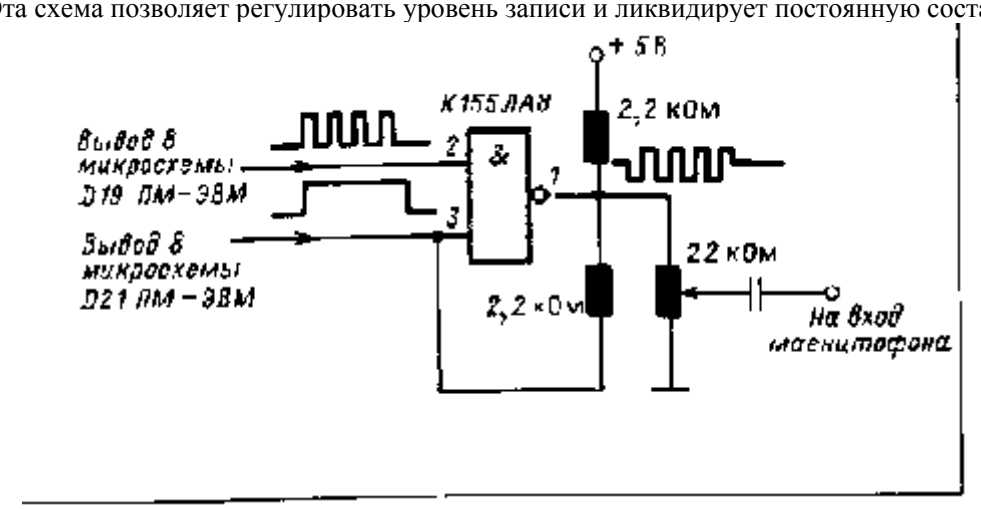


Рис. 10.5. Схема формирователя сигнала для записи на магнитофон

Зададимся приемлемой для записи и воспроизведения на бытовом магнитофоне частотой импульсов в пачках, равной 2 кГц. Тогда легко подсчитать число, заносимое в регистр E. Действительно, программный цикл от команды DCR E до команды JNZ занимает 15 тактов. В ПМ-ЭВМ тактовая частота равна 1 МГц. Следовательно, чтобы получить задержку, равную половине периода повторения импульсов 1/(2000-2) = 250 мкс, необходимо задать 250/15 ~ 16 программных циклов. В регистр E надо занести 020Q = 16. Если количество импульсов в пачке равно четырем (как задано в нашей программе), скорость передачи информации составит 2000/(4-3) ~ 167 бит/с.

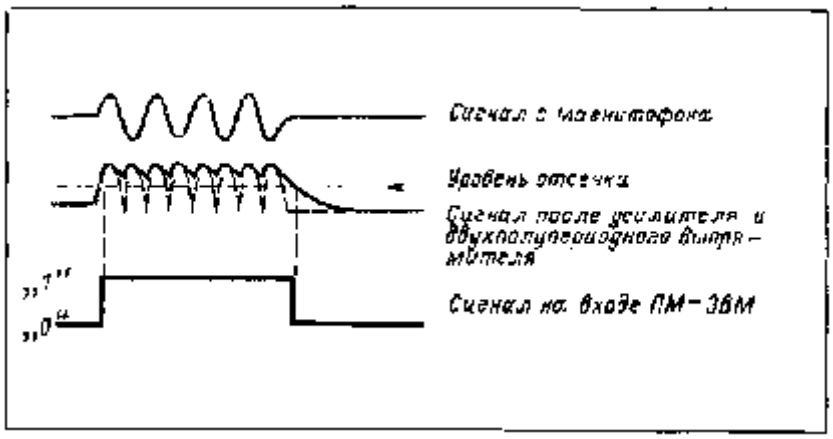


Рис. 10.6. Формирование выходного сигнала с магнитофона

Для ввода информации с бытового магнитофона необходимо из полученного сигнала сформировать нормальные логические посылки на ТТЛ-уровнях, причем воспроизведению импульсной пачки должен соответствовать уровень логической 1, а паузе -уровень логического 0 (рис. 10.6). Количество импульсов в пачке не принимается во внимание. Такую схему легко реализовать на операционных усилителях или на

обычных транзисторах. Формирователь может быть подключен к одному из входов порта 003, например выводу *4* микросхемы D23. В этом случае дальнейшие операции имеют чисто логический характер, легко реализуемый программным путем. Введем в ПМ-ЭВМ следующую демонстрационную программу, которую потом можно преобразовать в рабочую ПП ввода информации с бытового магнитофона:

```
014 000 006          MAG: MVI B, 010Q
014001 010
014 002 026                MVI D, OOOQ
014 003 000
014004315            Ml: CALL IMP
014005 120
014 006 014
014007332                  JC.M1
014010004
014011 014
014012315                  CALL IMP
014013 120
014014014
014015332                  JC.M1
014 016 004
014017014
014020315            M2: CALL IMP
014021 120
014022014
014023322                  JNC, M2
014 024 020
014025014
014026315                  CALL IMP
014027 120
014030014
014031 322                 JNC, M2
014032020
014033 014
014034025            M3: OCR D
014035315                  CALL IMP
014 036 120
014037014
014040332                  JC, M3
014041 034
014042014
014043315                  CALL IMP
014 044 120
014045014
014046332                  JC, M3
014 047 034
014 050 014
014051024            M4: INR D
014052315                  CALL IMP
014053  120
014054  014
014055322                  JNC.M4
014056 051
014057014
014060315                  CALL IMP
014061 120
014 062014
014 063 322                 JNC, M4
014064051
014065 014
014 066 172                 MOV A, D
014067.027                  RAL
014070171                  MOV A, C
014071037                  RAR
014072117                  MOVC, A
014 073 026                 MVI D, OOOQ
014 074 000
014075005                  OCR B
014076302                  JNZ,M3
014077034
```

```
014 100014
014 101 171                    MOV A, C
014 102 323                    OUT, OOOQ
014 103 000
014104257                      XRAA
014 105 117                    MOVC, A
014106303                      JMP.MAG
014 107000
014 110014
014120036              IMP: MVIE, 377Q
014 121 377
014122035              M5: DCR E
014123302                  JNZ, M5
014 124 122
014 125 014
014126333                      IN,003Q
014 127 003
014130037                     RAR
014 131 311                    RET
```

Программа поразрядно формирует в регистре С число, последовательно вводимое через младший разряд порта 003 с помощью ПП IMP, которая производит опрос порта в заданном темпе. В основной части программы в регистре D подсчитываются опросы и формируется знаковый разряд числа в зависимости от соотношения длительности пачки и паузы: если пачка длиннее паузы — в знаковом разряде 1, если короче — в знаковом разряде 0. Биты знакового разряда последовательно заносятся в регистр С. Темп опроса задается числом по адресу 014Q 121Q, заносимым в регистр Е. В данном случае задержка равна 15-256 = 3840 мкс, а частота опроса соответственно равна 1/0,003840 ~ 260 Гц. Следовательно, при длине импульсной пачки до 0,5 с в регистре D накопится число, не большее 130, что гарантирует отсутствие переполнения регистра.

Перед вводом программы в ПМ-ЭВМ подключим к выводу *4* микросхемы D23 вывод *16* микросхемы D16 второй ПМ-ЭВМ, в которую введена описанная выше программа вывода, настроенная на длительность импульсной пачки не более 0,5 с. Вместо второй ПМ-ЭВМ можно подключить приведенный на рис. 10.6 формирователь и магнитофон, на ленту которого предварительно записаны с ПМ-ЭВМ импульсные последовательности для одиночных байтов, но тогда мы сталкиваемся с проблемами отладки всего тракта: ПМ-ЭВМ — выходной формирователь — магнитофон в режиме записи, а затем магнитофон в режиме воспроизведения — входной формирователь — ПМ-ЭВМ. В то же время при непосредственной связи между двумя ПМ-ЭВМ отпадают проблемы согласования уровней и при этом могут быть заданы любые временные характеристики сигналов.

Настроим программу вывода передающей ПМ-ЭВМ на выдачу одного байта информации из регистра С, т. е. используем программу в том виде, как она приведена в тексте. Для вывода байта необходимо нажать кнопки СБРОС и П. Если до этого в принимающей ПМ-ЭВМ были также нажаты кнопки СБРОС и П, через 3 — 4 с на индикаторах ее порта 000 можно будет наблюдать код числа, идентичный коду, хранящемуся в регистре С передающей ПМ-ЭВМ. Чтобы снова передать байт информации, нужно снова нажать кнопки СБРОС и П. Меняя содержимое регистра С передающей ПМ-ЭВМ, можно проверить правильность передачи для любых кодов. После такой проверки можно произвести проверку правильности передачи массива информации. Для этого в память передающей ПМ-ЭВМ нужно дополнительно ввести управляющую программу и внести в основную программу указанные выше изменения. Управляющая программа имеет вид

```
014110041              MM1: LXI H, ADM
014 111 000
014 112 015
014113116              MM2: MOV C.M
014114315                  CALL M1
014 115 002
014 116 014
014117054                  INR L
014120302                  JNZ,MM2
014 121 113
014 122014
014 123 166                HLT
```

Здесь ADM = ADMHQ ADMLQ - символический адрес начала массива чисел, заносимый в регистровую пару H; Ml - метка начала программы вывода информации. Максимальная длина передаваемого массива - 256 байт, если он начинается с адреса ADMHQ OOOQ и кончается адресом ADMHQ 377Q.

Для проверки правильности передачи необходимо нажать кнопки СБРОС и П принимающей ПМ-ЭВМ, нажать кнопку СБРОС, набрать адрес начала управляющей программы (метка ММ1) и нажать кнопку П передающей ПМ-ЭВМ. На индикаторах порта 000 принимающей ПМ-ЭВМ будут последовательно появляться коды передаваемого массива. Коды массива могут быть занесены в ОЗУ принимающей ПМ-ЭВМ. Для этого

вместо команды MOV C, A по адресу 014Q 105Q необходимо занести команду RET и ввести дополнительно управляющую программу вида

```
014  041                    MA1: LXIH,ADM
014 141 000
O14 143 161                 MA2: MOVM,C
014 144 315                      CALL MAG
014 145 000
014 146 014
014 147 054                      iVrS » л
014 150 302                      JNZ, MA2
014 151 143
014 152 014
014 153 166                      HLT
```

Теперь для запуска программы принимающей ПМ-ЭВМ необходимо после нажатия кнопки СБРОС набрать адрес начала управляющей программы (метка MA1), а затем нажать кнопку П. Принимаемые коды по-прежнему наблюдаются на индикаторах порта 000, но после окончания передачи правильность принятого массива легко проверить, считывая его последовательно на индикаторах порта 002 принимающей ПМ-ЭВМ с помощью клавиатуры. После такой проверки можно уже отладить передачу информации с рабочей скоростью порядка 150 бит/с, затем включить между передающей и принимающей ПМ-ЭВМ выходные и входные формирователи, снова проверить передачу и лишь после этого включать магнитофон.

10.4. ДИСПЛЕЙ НА БАЗЕ БЫТОВОГО ТЕЛЕВИЗОРА ИЛИ ОСЦИЛЛОГРАФА

Телевизионный дисплей — одно из наиболее мощных средств общения человека с вычислительной машиной. Информационная емкость телевизионного экрана велика, а скорость изменения информации согласована со скоростью реакции человека. Здесь мы будем иметь в виду либо телевизионный приемник с входом, позволяющим подавать обычный видеосигнал, либо специальный видеомонитор, в котором разделены входные каналы синхронизации горизонтальной, вертикальной разверток и модуляции яркости. Модуляторы, предназначенные для подачи сигнала на антенный вход телевизора, не рассматриваются.

Читатель должен строго соблюдать правила техники безопасности при работе с таким высоковольтным прибором, как телевизор, особенно если он будет каким-либо образом переделывать его!

В телевизоре два генератора развертки отклоняют электронный луч, модулированный по интенсивности. Первый перемещает луч горизонтально и быстро возвращается. Период составляв! 64 мкс (частота 15,625 кГц), причем четвертая часть периода занята обратным ходом луча. Вторая отклоняющая система перемещает луч вертикально с периодом 20 мс (50 Гц). Таким образом, на экране появляются 312 горизонтальных линий (строк), и если импульс обратного .вертикального хода луча формируется во время появления 313-й строки, то обеспечивается обьиная стандартная развертка в 625 строк.

Для алфавитно-цифрового дисплея МП-системы не требуется разрешающая способность в 625 строк, вполне достаточно 312, что упрощает схему дисплея. Устройство, управляющее телевизионным (ТВ) монитором, вырабатывает три сигнала. Фронт импульса синхронизации горизонтальной развертки H определяет начало обратного хода луча. Минимальная длительность импульса H зависит от ТВ монитора. Фронт импульса вертикальной синхронизации V определяет начало обратного хода луча по вертикали. Модуляционный сигнал интенсивности Z в случае цифрового дисплея является двоичным, т. е. принимает только два значения. Телевизионные мониторы с цифровым управлением имеют три независимых входа для сигналов H, V и Z. Большинство ТВ-мониторов и некоторые бытовые телевизоры имеют вход для смешанного сигнала C с более жесткими ограничениями по длительности и положению синхроимпульсов. Сигнал на антенном входе бытового телевизора эквивалентен высокочастотно-модулированному полному сигналу C.

Разрешающая способность экрана по горизонтали ограничена шириной полосы пропускания и электронной трубкой. Для обычного ТВ-приемника полоса пропускания не превышает 5 МГц.

Любой символ может быть сгенерирован на экране путем соответствующей модуляции сигнала Z. На практике знакогенераторы строятся на интегральных микросхемах, обладающих заранее запрограммированной памятью, ставящей в соответствие каждому коду символа последовательность точек, представляющих знак. Эта последовательность формируется на строке путем модуляции сигналом Z.

Важный параметр - время отклика, т. е. полное время на генерацию символа с момента задания адреса в ЗУ. Нужно достигнуть хорошего компромисса между плотностью символов, их читаемостью на экране и простотой цепей управления. Для хорошей читаемости каждый символ должен быть окружен достаточным пространством.

Необходимые сигналы формируются цепочкой делителей. Первый делитель определяет количество точек на строке для каждого символа, второй - количество символов на строке с учетом фиктивных знаков - пропусков во время обратного хода луча. Этот счетчик генерирует синхроимпульсы горизонтальной развертки и гашения обратного хода луча H, следующий - количество строк развертки на формирование одного ряда символов с

учетом разделительных линий. Последний делитель определяет количество рядов символов на экране, включая невидимые на экране строки, когда генерируются синхроимпульс вертикальной развертки и гашения обратного хода луча V. Параллельный код с .выхода знакогенератора преобразуется с помощью сдвигового регистра в последовательность битов, задающих рисунок данного символа на строке. Выходы Q и R счетчиков соответствуют адресам символов на экране (рис. 10.7).

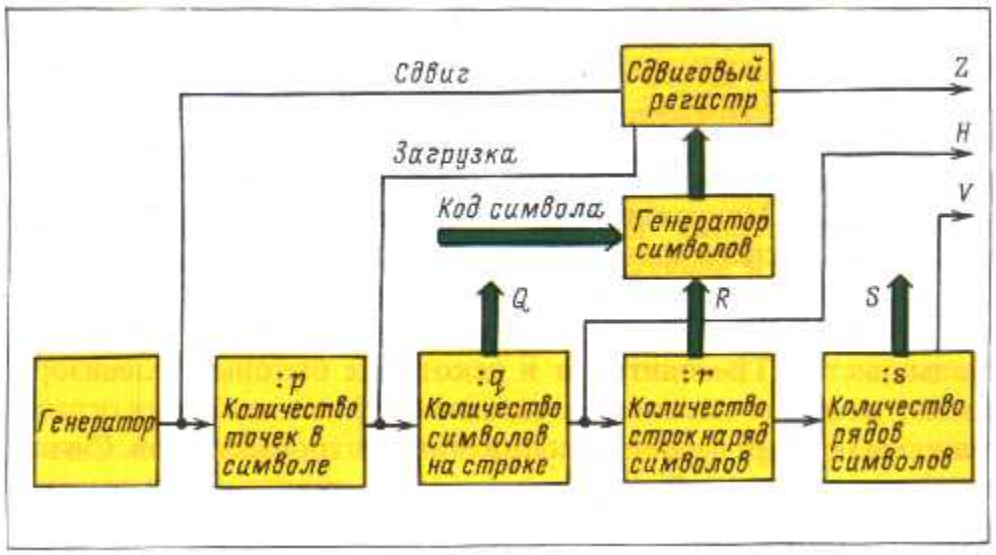


Рис. 10.7. Схема генератора знаков

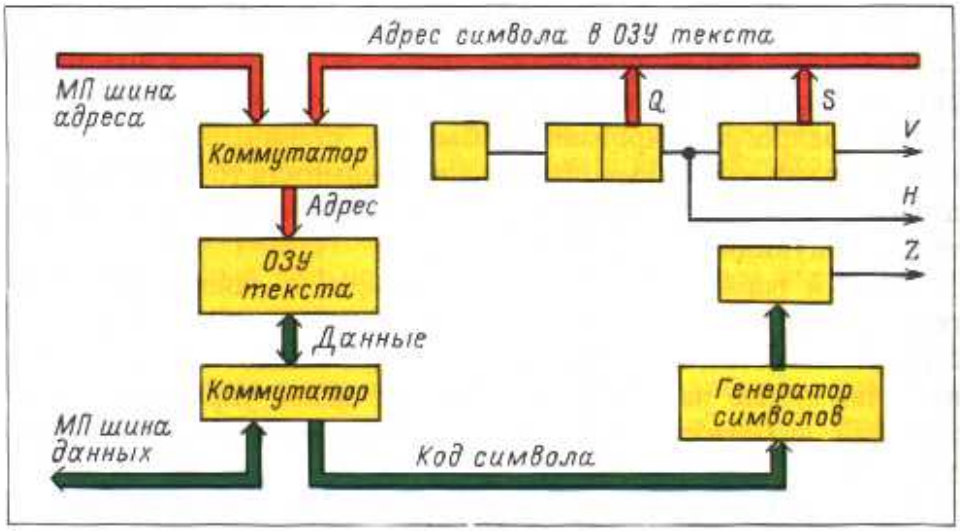


Рис. 10.8. Схема приставки для формирования буквенно-цифровых символов на экране ТВ

Страница текста запоминается в специальном местном ЗУ, Символы передаются на дисплей параллельно, содержимое памяти передается на знакогенератор при синхронизации с дели-

телей (рис. 10.8). Процессор может модифицировать содержимое ЗУ текста. При этом блокируется обращение к ЗУ от дисплея.

10.5. ПРОСТОЙ ГРАФИЧЕСКИЙ ДИСПЛЕЙ

Запоминающее устройство на 256 слов по 8 бит позволяет отображать 2048 точек на экране в виде сетки 64x32 элемента. В этом случае знакогенератор не требуется, поскольку каждый бит из памяти прямо отображается без преобразования. Схема такого устройства индикации дана на рис. 10.9. Каждая точка формируется из восьми соседних горизонтальных строчек развертки.

Логика работы ТВ-дисплея довольно проста и может быть без труда реализована даже на такой простой вычислительной машине, как ПМ-ЭВМ. Вся сложность в том, что быстродействие ПМ-ЭВМ, да и многих других машин более высокого класса, недостаточно, чтобы уложиться в жесткие рамки телевизионного стандарта. Выход заключается либо в создании специализированного периферийного процессора — контроллера ЭЛТ, каковыми являются рассмотренные выше устройства (промышленностью выпускается специализированная микросхема КР580ВГ75 - программируемый контроллер ЭЛТ), либо в отступлении от стандарта, если требования к информационной емкости дисплея, как в нашем случае, не велики.

Покажем, что, не используя практически никакой дополнительной аппаратуры, можно реализовать вывод поля памяти объемом 256 байтов (сетка 64x32 элемента) на экран обычного осциллографа. Подключим к ПМ-ЭВМ схему, показанную на рис. 10.10. Здесь порт 000 служит для параллельной выдачи содержимого ячеек памяти. Сдвиговый регистр преобразует параллельный код в последовательный. Сигнал с выхода сдвигового регистра может быть подан непосредственно на вход Z модуляции яркости осциллографа. Порт 001 служит для вывода синхроимпульса строчной развертки H, а порт 002 — для вывода синхроимпульса кадровой развертки *V*. Кадровая и строчная развертки могут быть сформированы с помощью простых пассивных цепей непосредственно из соответствующих синхроимпульсов. Выдаваемые портами 000, 001 и 002 сигналы формируются с помощью следующей программы:

```
014 000 176          Ml: MOV A, M 7 - количество тактов
014 001 323              OUT,OOOQ10 014 002 000
014 003 167              MOV M, A 7
014 004 000              NOP          4
014 005 054              INR L 5
014 006 035              DCRE 5
014 007 302              JNZ,M1 10
014 010 000
014 011 014
Итого: 48 тактов
```

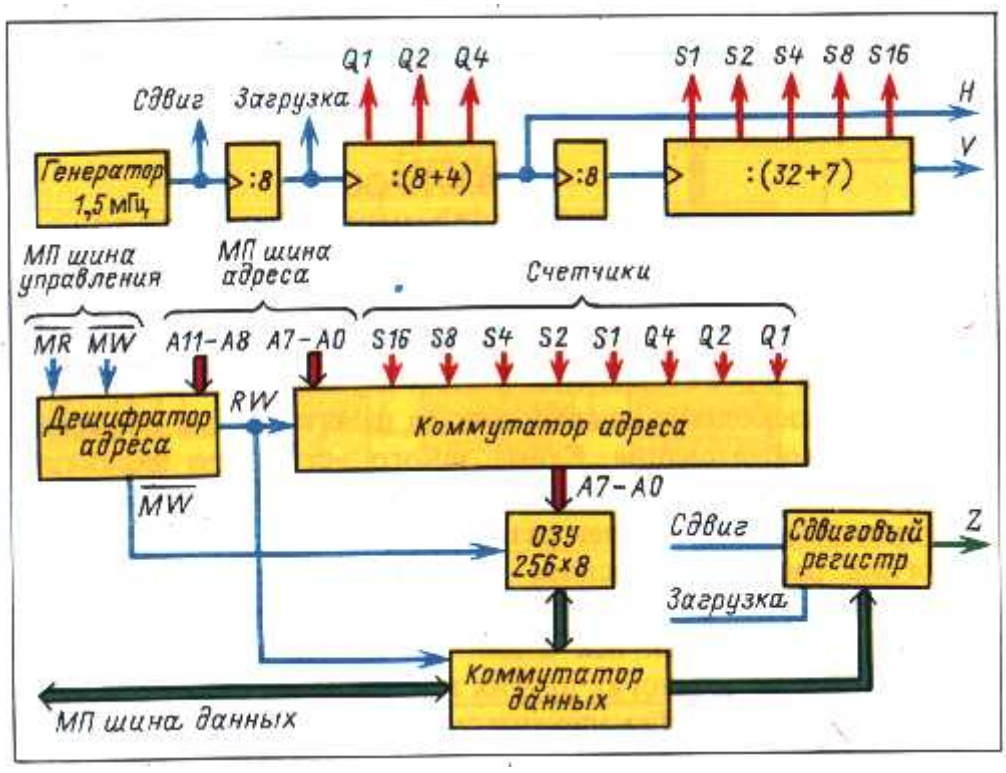


Рис. 10.9. Схема приставки для формирования графической информации на экране ТВ

Эта часть программы выводит в порт 000 очередной байт из памяти. Команды MOV M, A и NOP вставлены, чтобы получить выраженную в тактах длительность этого фрагмента программы кратной восьми. В итоге получаем 48 тактов. Это значит, что стоящий на выходе сдвиговый регистр должен выдавать очередной разряд числа через каждые 6 тактов, чтобы завершить выдачу всех восьми разрядов за 48 тактов. После восьми циклов прохождения первого фрагмента программы содержимое регистра E станет

равным нулю и начнет исполняться следующий фрагмент программы, предназначенный для выдачи синхроимпульса строчной развертки, здесь же происходит и загрузка регистра E:

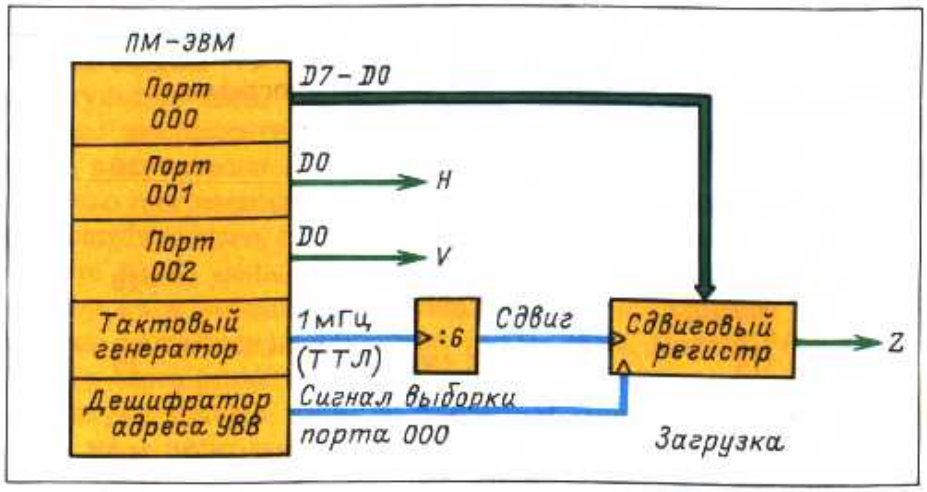


Рис. 10.10. Схема приставки для формирования графической информации на экране осциллографа

```
014012257          XRAA              4
014013323          OUT.OOOQ           10
014 014 000
014015323          OUT,001Q          10
014016 001
014017036          MVIE, 010Q        7
014 020 010
014 021 000          NOP               4
014 022 000          NOP               4
014 023 000          NOP               4
014 024 000          NOP               4
014 025 000          NOP               4
014 026 000          NOP               4
014 027 000          NOP               4
014 030 000          NOP               4
014031000          NOP               4
014032057          CMA               4
014033323          OUT, 001Q         Ю
014 034 001
014035025          DCRD              5
014036302          JNZ,M1            Ю
014037 000
014040014                            ----------
Итого: 56.тактов
```

Этот фрагмент программы имеет продолжительность 96 тактов, т. е. вдвое больше времени выдачи одного байта из памяти. Следовательно, общая продолжительность цикла строчной развертки составляет 48-8 + 96 = 480 тактов (т. е. 480 мкс). Количество строк определяется числом 32 (040Q), загружаемым в регистр D следующим фрагментом программы:

```
014041257          XRA, A              4
014042323          OUT, 002Q            10
014 043 002
014 044 026          MVI D, 040Q       7
014 045 040
014046041          LXIH, ADM         10
014 047 000
014050015
014051315          CALLINP           17
014 052 X
014053 Y
014 054 057          CMA               4
014055323          OUT, 002Q         10
014 056 002
014057303          JMP, Ml           10
014 060 000
014061 014
Итого: 72 + время исполнения ПП ввода IMP (3840 тактов)
```

В этом фрагменте происходит выдача синхроимпульса кадровой развертки и задается адрес начала массива информации, выводимой на дисплей. Отведя на обратный ход луча время, равное четверти длительности прямого хода луча для вертикальной развертки, получим период кадровой развертки, равный 480-32 + 480-8 = 19 200 мкс. Отсюда получаем частоту кадровой развертки, равную примерно 50 Гц, что нас вполне удовлетворяет. В последнем фрагменте программы, длящемся 3840 мкс, предусмотрен вызов ПП ввода информации в ОЗУ с клавиатуры или внешнего устройства для изменения картинки на экране дисплея. Эта ПП должна иметь строго определенную длительность, необходимую для стабильности параметров развертки. Если в этом промежутке времени исполняются другие программы неизвестной длительности, надо организовать систему с внешними прерываниями.

10.6. ЗВУКОВАЯ СИГНАЛИЗАЦИЯ

В рассмотренной в § 9.2 конструкции кодового замка мы уже использовали звуковую сигнализацию для подачи сигнала тревоги. Для этого наушник или репродуктор подключался через резистор 200 Ом к младшему разряду порта 001 между выводом *16* микросхемы D18 и источником +5 В. Определенный звуковой тон получался в результате того, что в порт 001 выводилось содержание регистра, инкрементируемого в постоянном темпе. Частота сигнала, получаемого с младшего разряда порта, равна половине частоты инкрементирования и задается временной задержкой соответствующего программного цикла. В следующем разряде частота вдвое меньше и т. д. Следовательно, подключая репродуктор не к младшему разряду порта, а к старшим, получим тон на соответствующее число октав ниже. Подавая на репродуктор через резисторы сигналы с разных разрядов порта, получим звуки одного тона, но с разным содержанием гармоник. Если же в несколько портов выводить содержимое регистров, к которым с постоянной частотой прибавляются различные числа, то, подключив репродуктор, к выходам портов через резисторы, можно получить различные аккорды.

Таким образом, звуки разных частот можно получать, прибавляя к содержимому регистра разные числа или же меняя частоту суммирования.

Подключим репродуктор и введем в ПМ-ЭВМ следующую программу:

```
014000061          Ml: LXISP,TAB
014001 036
014002014
014003301          M2: POP B
014 004 170              MOV A, B
014 005 267              ORA A
014006312              JZ.M1
014 007 000
014010014
014011171          M3: MOVA.C
014012267              ORA A
014013312              JZ,M4
014014 023
014015  014
014016034              INRE
014017173              MOV A, E
014020323              OUT, 0010
014 021 001
014022171              MOV A, C
014023075          M4: DCRA
014024302              JNZ,M4
014025 023
014026014
014 027 005              UCR B
014030302              JNZ,M3
014031  Oil
014032 014
014 033 303                JMP, M2
014 034 003
014035 014
014 036 175        TAB: до
014037 145
014040160                  ре
014041161
014 042 144                ми
014043 177
```

```
014 044 136                    фа
014 045 207
014 046 124                    соль
014 047 230
014050113                    ля
014 050 252
014051102                    си
014 052 277
014 053 077                    до
014054 313
014 055 000
014 056 000
```

Нажмем кнопки СБРОС и Ц. Репродуктор начинает наигрывать гамму. Воспроизводимая мелодия запрограммирована таблицей, начинающейся с метки TAB. Таблица содержит пары чисел, последовательно извлекаемые процессором из памяти с помощью команды POP B. Первое число из пары заносится в регистр C и задает высоту тона, второе число заносится в регистр B и задает длительность звучания ноты. Если в регистре B оказывается нуль, программа возвращается к началу таблицы. Если в регистре C оказывается нуль, формируется пауза. Подсчитав длительности программных циклов, нетрудно определить, какие числа в таблице соответствуют различным нотам, и запрограммировать любую мелодию. Надо иметь в виду следующее: для того чтобы различные ноты имели одинаковую длительность звучания, произведение первого числа из пары на второе должно быть постоянным.

Читатель может усложнить программу, чтобы получить более интересное звучание. Он может также самостоятельно разработать программу "музыкального ящика", позволяющую самому наигрывать различные мелодии, пользуясь клавиатурой ПМ-ЭВМ. Мир машинной музыки необъятен, и начав с простейших программ, читатель при желании может достичь в этой области новых и оригинальных результатов.

10.7. ДРУГИЕ ВОЗМОЖНОСТИ ПМ-ЭВМ

В этой главе мы рассмотрели расширение возможностей ПМ-ЭВМ в части клавиатуры и индикации, памяти, внешнего накопителя и ТВ-дисплея. Обслуживание всех этих устройств осуществлялось программным путем, без значительного усложнения схемы самой ПМ-ЭВМ и ее устройств ввода/вывода. Программное обслуживание внешних устройств со стороны центрального процессора требует больших затрат машинного времени за счет исполнения основной программы и снижает производительность вычислительной системы. Использование прерываний, прямого доступа к памяти (ПДП) и программируемых периферийных микросхем позволяет повысить производительность и гибкость вычислительной системы. Периферийные программируемые микросхемы являются специализированными периферийными процессорами, берущими на себя часть задач центрального процессора. При этом техническая сложность системы зачастую не возрастает или даже уменьшается.

Отечественной промышленностью выпускается широкая номенклатура периферийных программируемых микросхем, со-гласуемых с МП типа КР580ВМ80А.

Использование программируемых микросхем в конкретной МП-системе связано: 1) с заданием режимов ее работы в системе, адресацией и определением электрических связей с внутренними шинами МП-системы и периферийными устройствами; 2) с вводом фрагмента программы, задающей режимы работы микросхемы (программа инициализации); 3) с обращением к микросхеме в процессе исполнения рабочей программы.

Количественный рост МП-системы ведет и к ее архитектурному усложнению. Возникает необходимость каким-либо образом стандартизовать связи микро-ЭВМ. Известно много стандартов внутрисистемных шин МП-систем. Для МП серии КР580ИК80А наиболее известны шины MULTIBUS и 8=100, позволяющие объединять в систему периферийные, процессорные блоки и блоки памяти различных изготовителей.

Большая МП-система требует специального математического обеспечения (МО). Если при написании и отладке программ емкостью до 1 Кбайт можно с успехом обойтись программированием в кодах, то программы большего размера пишутся обычно на языках различных уровней, в том числе и на специализированных языках высокого уровня. Для перевода таких программ на машинный язык необходимы программы-интерпретаторы, а для отладки — программы редактирования и отладки. Чтобы весь комплекс программ мог успешно функционировать, необходимо использовать какую-либо операционную систему (ОС). Все это доводит общую емкость МО до нескольких сотен килобайт и требует для хранения специальных накопителей. Поэтому не имеет смысла, как говорят, "с нуля" проектировать большую МП-систему. Гораздо разумнее использовать уже имеющийся опыт, взяв в качестве образца готовую промышленную систему.

Для любой системы, состоящей из заданного набора элементов, есть предел сложности, определяемый надежностью входящих в нее элементов, надежностью связей между элементами, допусками при проектировании, программным обеспечением, архитектурой системы, внешними условиями и теми функциями, которые система должна выполнять. Не ставьте перед собой сверхзадач и не переусложняйте систему, иначе она будет неработоспособной. Имейте мужество отказаться от дальнейшего наращивания уже имеющейся системы, когда ее надежность падает до неприемлемого уровня, чтобы заново перепроектировать систему на современном идеологическом и техническом уровне.

Рассмотрим коротко характеристики некоторых программируемых микросхем, находящих широкое применение в МП-системах.

Микросхема программируемого последовательного интерфейса типа КР580ВВ51А является универсальным синхронно-асинхронным приемопередатчиком (УСАПП), предназначенным для осуществления связи в последовательном формате между МП-системой и внешним абонентом. Микросхема используется как периферийное устройство, программируемое центральным процессором МП-системы почти для любого стандартного протокола последовательной передачи данных. Микросхема принимает данные от МП-по шине данных в параллельном формате и преобразует их в непрерывный поток последовательных данных для передачи абоненту. Микросхема может одновременно с этим принимать последовательные данные, преобразовывать их в параллельный формат и передавать микропроцессору. Она сигнализирует микропроцессору о готовности к приему нового слова, предназначенного для передачи абоненту в последовательном формате, а также о готовности передать МП-принятое от абонента слово. МП-может в любой момент времени считать из микросхемы слово состояния, указывающее на возможные ошибки при приеме информации и содержащее признаки наличия управляющих сигналов.

Микросхема типа КР580ВИ53 представляет программируемый таймер/счетчик, предназначенный для работы в качестве периферийного устройства МП-системы. Микросхема состоит из трех независимых 16-битовых счетчиков с максимальной частотой счета 2 МГц. Каждый счетчик микросхемы может работать в одном из шести запрограммированных режимов. С помощью микросхемы таймера/счетчика решается одна из наиболее часто встречающихся в МП-системах задач — генерирование точных программно задаваемых временных задержек взамен организации холостых программных циклов. Пользователь подключает соответствующим образом микросхему, загружает один из счетчиков нужным числом, после чего в результате подачи команды счетчик отсчитывает заданную задержку и вырабатывает сигнал прерывания для МП-системы. Микросхема может выполнять функции не только генератора задержки, но и программируемого генератора заданной частоты, счетчика событий, датчика реального времени и др.

Микросхема типа КР580ВВ55А представляет собой программируемый периферийный интерфейс, предназначенный для приема в МП-систему и выдачи из нее информации в параллельном коде, функции микросхемы в системе задаются программно, поэтому, как правило, не требуется подключать какие-либо дополнительные логические схемы. Предназначенные для ввода и вывода внешней информации 24 ножки микросхемы могут быть индивидуально запрограммированы группами по 12 и использованы в трех различных режимах работы. В первом из них (Режим 0) в каждой группе из 12 ножек часть может быть запрограммирована на ввод, а часть — на вывод. Во втором режиме (Режим 1) группа программируется так, что 8 ножек служат для ввода или вывода, а 3 ножки из оставшихся 4 предназначены для сигналов квитирования (запрос — подтверждение) и прерывания. В третьем режиме (Режим 2) из 8 ножек организуется 8-разрядная двунаправленная шина, а 5 ножек (с за-емом из другой группы) предназначены для сигналов квитирования и прерывания.

Микросхема типа КР580ВТ57 представляет собой четырех-канальный программируемый контроллер прямого доступа к памяти (ПДП), предназначенный для организации высокоскоростного обмена данными между периферийными устройствами и памятью МП-системы, построенной на базе микропроцессора КР580ВМ80А. Функция микросхемы заключена в основном в

мы имеют информационную емкость 2К х 8 и программируются побайтно импульсами на ТТЛ-уровне. Микросхема приводится в исходное состояние (когда во всех разрядах стоят единицы) путем облучения ультрафиолетовым источником света, в качестве которого можно использовать с соответствующими предосторожностями обычную бытовую кварцевую лампу, а в качестве программатора — саму ПМ-ЭВМ.

ПРИЛОЖЕНИЯ

Система команд микропроцессора КР580ИК80А

ПРИЛОЖЕНИЕ1

Мнемокод команды	Описание команды	Длина команды байт	, Изменяемые флаги	Число тактов
INRr	Увеличение содержимого регистра г на единицу	1	Z,S,P,AC	5
INRM	Увеличение на единицу содержимого ячейки	1	Z,S,P,AC	10

	памяти по адресу, указанному в регистрах H и L			
DCRr	Уменьшение содержимого регистра г на единицу	1	Z,S,P, AC[2]	5
DCRM	Уменьшение на единицу содержимого ячейки памяти по адресу, указанному в регистрах H иЬ	1	Z, S,P, AC[2]	10
MOVrl,r2	Пересылка данных из регистра г 2 в регистр rl	1	-	5
MOV M, г	Пересылка данных из регистра г в ячейку памяти по адресу, указанному в регистрах H и L	1	-	7
MOV г, M	Пересылка данных в регистр г из ячейки памяти по адресу, указанному в регистрах H и L	1	-	7
ADDr	Суммирование содержимого регистра г и содержимого аккумулятора	1	Z,S,P,C,AC	4
ADDM	Суммирование содержимого ячейки памяти по адресу, указанному в регистрах H и L, с содержимым аккумулятора	1	Z,S,P,C, AC	7
ADCr	Сложение с учетом переноса содержимого регистра г и содержимого аккумулятора	1	Z,S,P, C, AC	4
ADCM	Сложение с учетом переноса содержимого ячейки памяти по адресу, указанному в ре-	1	Z,S,P,C,AC	7

Продолжение прилож. 1

Мнемокод команды	Длина Описание команды команды, Изменяемые флаги байт			Число тактов
	ги страх H и L, с содержимым аккумулятора			
SUB г	Вычитание содержимого регистра г из содер- 1 Z,S,P, C , AC жимого аккумулятора			4
SUBM	Вычитание из содержимого аккумулятора 1 Z, S, P, C , AC содержимого ячейки памяти по адресу, указанному в регистрах H и L			7
SBBr	Вычитание с заемом содержимого регистра г 1 Z, S, P, C , AC из содержимого аккумулятора			4
SBBM	Вычитание с заемом из содержимого аккуму- 1 Z,S,P,C , AC лятора содержимого ячейки памяти по адресу, указанному в регистрах H и L			7
ANA г	Поразрядное И над содержимым регистра г 1 Z , S, P, C = 0, AC = 0 и аккумулятора			4
ANAM	Поразрядное И над содержимым аккумуля- 1 Z , S, P, C = 0, AC = 0 тора и ячейки памяти по адресу, указанному в регистрах H и L			7
XRAr	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над со- 1 Z, S, P, C = 0, AC = 0 держимым регистра г и аккумулятора			4
XRAM	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над со- 1 Z, S, P, C = 0, AC = 0 держимым аккумулятора и ячейки памяти по адресу, указанному в регистрах H и L			7
ORAr	Поразрядное ИЛИ над содержимым регист- 1 Z,S,P,C = 0, AC = 0 ра г и аккумулятора			4
ORAM	Поразрядное ИЛИ над содержимым аккуму- 1 Z, S, P, C = 0, AC = 0 лятора и ячейки памяти по адресу, указанному в регистрах H и L			7
CMP г	Сравнение содержимого регистра г и аккуму- 1 (Z, S, P, C, AC)[3] лятора			4
CMPM	Сравнение содержимого аккумулятора и ячей- 1 (Z, S, P, C, AC)[3] ки памяти по адресу, указанному в регистрах H и L			7
INXrp	Увеличение на единицу содержимого пары ре- 1 - гистров гр (B, D, H, SP)			5
DCXrp	Уменьшение на единицу содержимого пары 1 - регистров гр (B, D, H, SP)			5
DADrp	Сложение содержимого пары регистров тр 1 C (B, D, H, SP) с содержимым пары регистров H и L и хранение результата в H и L			10
POPrp	Выдача данных из стека в пару регистров гр 1 (Z, S, P, C, AC)[6]			10

	(B, D, H) или в аккумулятор и регистр признаков PSW		
PUSH гр	Занесение в стек содержимого пары регистров гр (B, D, H) или содержимого аккумулятора и регистра признаков PSW	1	11
STAXгр	Запись содержимого аккумулятора в ячейку памяти, косвенно адресуемую парой регистров гр (B, D)	1	7
LDAX гр	Запись в аккумулятор содержимого ячейки памяти, косвенно адресуемой парой регистров гр (B, D)	1	7
RNZ	Возврат из подпрограммы при отсутствии нуля (флаг нуля в состоянии 0)	1	5/11
RZ	Возврат из подпрограммы при наличии нуля (флаг нуля в состоянии 1)	1	5/11
RNC	Возврат из подпрограммы при отсутствии переноса (флаг переноса в состоянии 0)	1	5/11
RC	Возврат из подпрограммы при наличии переноса (флаг переноса в состоянии 1)	1	5/11

Продолжение прилож. 1

Мнемокод команды	Описание команды	Длина команды, байт	Изменяемые флаги	Число тактов
RPO	Возврат из подпрограммы при отсутствии четности (флаг четности в состоянии 0)	1		5/11
RPE	Возврат из подпрограммы при наличии четности (флаг четности в состоянии 1)	1		
RP	Возврат из подпрограммы при положительном результате (флаг знака в состоянии 0)	1		5/11
RM	Возврат из подпрограммы при отрицательном результате (флаг знака в состоянии 1)	1		5/11
RET	Безусловный возврат из подпрограммы	1	C^4	5/11
RLC	Циклический сдвиг содержимого аккумулятора влево	1	C^4	
RRC	Циклический сдвиг содержимого аккумулятора вправо	1	C^4 C	
RAL	Циклический сдвиг содержимого аккумулятора влево, включая флаг переноса	1	C^4	
RAR	Циклический сдвиг содержимого аккумулятора вправо, включая флаг переноса	1	C^4	
XCHG	Обмен данными между парами регистров H, L HD.E	1		4
XTHL	Обмен данными между двумя верхними ячейками стека и парой регистров H, L (сначала самая верхняя обменивается с L , затем следующая с H)	1		18
SPHL	Передача в указатель стека содержимого регистров H и L	1		
PCHL	Передача в счетчик команд содержимого пары регистров H и L с последующим выполнением программы с адреса, равного новому содержимому счетчика команд (по существу команда перехода)	1		5
HLT	Останов программы	1		7
NOP	Отсутствие операции	1		4
DI	Запрет на прерывание программы	1		4
EI	Разрешение прервать программу	1		4
DAA	Перевод двоичного представления содержимого аккумулятора в двоично-десятичный код	1		4
CMA	Поразрядное инвертирование содержимого аккумулятора	1		4
STC	Установка флага переноса C в единицу	1	C = 1	4
CMC	Инвертирование значения флага переноса	1	C=C	4

RSTA	Повторный запуск программы с адреса 1 - $8A_{10} = (OA0_8)$		11
ADI<B2>	Сложение содержимого байта B2 с содержи- мым аккумулятора	2	Z,S,P, C, AC 7
ACI <B2>	Сложение с учетом переноса содержимого байта B2 с содержимым аккумулятора	2	Z,S, P, C, AC 7
SUI <B2>	Вычитание содержимого байта B2 из содер- жимого аккумулятора	2	Z, S, P, C^1 , AC^2 7
SBI <B2>	Вычитание с заемом содержимого байта B2	2	Z, S, P, C^1 , AC^2 7
ANI <B2>	из содержимого аккумулятора Поразрядное И над содержимым байта B2 и содержимым аккумулятора	2	Z , S, P, C = 0, AC = 0 7
XRI <B2>	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над со- держимым байта B2 и содержимым аккуму- лятора	2	Z, S, P, C = 0, AC =0 7

Продолжение прилож. 1

Мнемокод команды	Описание команды	Длина команды, байт	Изменяемые флаги	Число тактов
ORI<B2>	Поразрядное ИЛИ над содержимым байта B2 и содержимым аккумулятора	2	Z,S,P, C = 0, AC=0	7
CPI <B2>	Сравнение содержимого байта B2 с содержи- мым аккумулятора	2	(Z, S, P,C, AC)[3]	7
IN <B2>	Ввод данных в аккумулятор из порта ввода, определяемого адресом в байте B2	2	-	10
OUT<B2>	Вывод данных из аккумулятора в порт вы- вода, определяемый адресом в байте B2	2	-	10
MVI r <B2>	Занесение содержимого байта B2 в регистр r	2	-	7
MVI M <B2>	Запись содержимого байта B2 в ячейку памя- ти по адресу, указанному в регистрах H и L	2	-	7
JNZ<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при отсутствии нуля (флаг нуля в состоянии 0)	3	-	10
JZ<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при наличии нуля (флаг нуля в состоянии 1)	3	-	10
JNC<B2><B3> JC<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при отсутствии переноса (флаг переноса в состоянии 0) Переход в программе к выполнению команды по адресу в B2, B3 при наличии переноса (флаг переноса в состоянии 1)	3 3	-	10 10
JPO<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при наличии нечетности (флаг четности в состоянии 0)	3	-	10
JPE<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при наличии четности (флаг четности в состоянии 1)	3	—	10
JP<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при положительном результате (флаг знака в состоянии 0)	3		10
JM<B2><B3>	Переход в программе к выполнению команды по адресу в B2, B3 при отрицательном результате (флаг знака в состоянии 1)	3		10
JMP<B2><B3>	Безусловный переход к команде по адресу в B2, B3	3	—	10
CNZ<B2><B3>	Вызов подпрограммы при отсутствии нуля (флаг нуля в состоянии 0)	3		11/17[5]
CZ<B2><B3>	Вызов подпрограммы при наличии нуля (флаг нуля в состоянии 1)	• 3	~	11/17[5]
CNC<B2><B3>	Вызов подпрограммы при отсутствии перенос, i3 (флаг переноса в состоянии 0)		—	И/17[5]
CC<B2><B3>	Вызов подпрограммы при наличии переноса (флаг переноса в состоянии 1)	3	—	И/17[5]
CPO<B2><B3>	Вызов подпрограммы при наличии нечетности (флаг четности в состоянии 0)	3	—	11/17[5]
CPE<B2><B3>	Вызов подпрограммы при наличии четности (флаг четности в состоянии 1)	3	—	И/17[5]

CP<B2><B3>	Вызов подпрограммы при наличии положительного результата (флаг знака в состоянии 0)	3 И/17[5]
CM<B2><B3>	Вызов подпрограммы при наличии отрицательного результата (флаг знака в состоянии 1)	3 И/17[5]

Продолжение прилож. 1

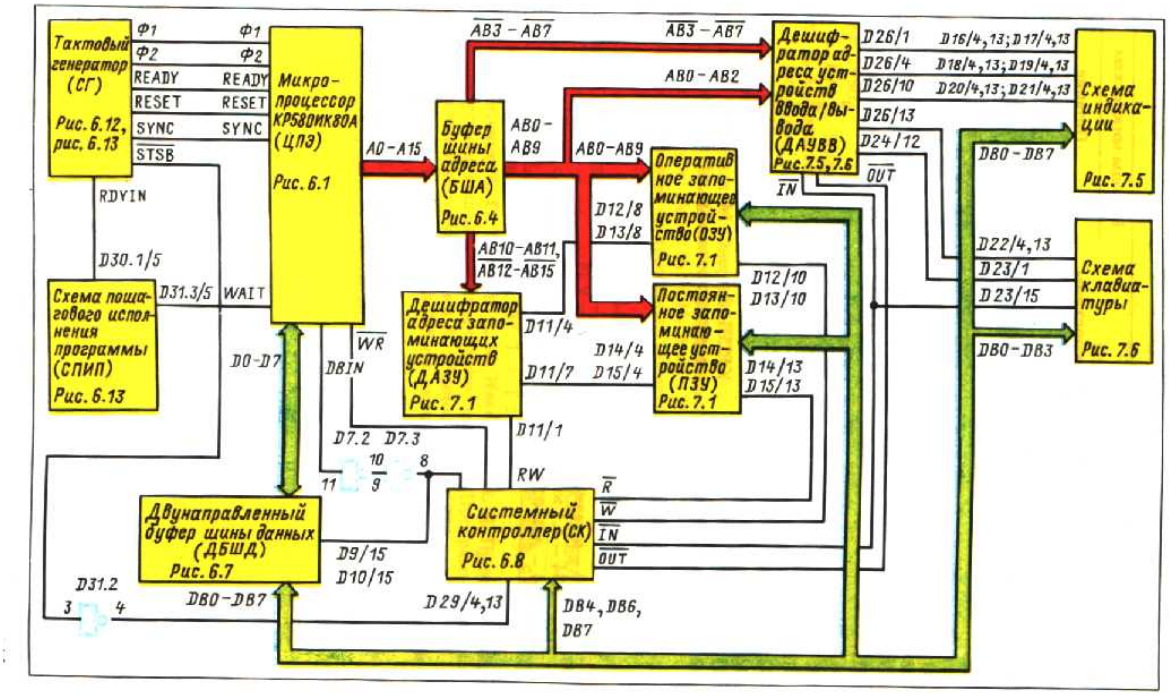
Мнемокод команды	Длина Описание команды команды, Изменяемые флаги байт	Число тактов
CALL<B2><B3>	Вызов подпрограммы из памяти по адресу, указанному в байтах B2, B3	3 - 17
LXIгр<B2><B3>	Занесение содержимого двух байтов B2, B3 в пару регистров гр (B, D, H, SP)	3 - 10
STA<B2><B3>	Запись содержимого аккумулятора в ячейку памяти по адресу в B2, B3	3 - 13
LDA<B2><B3>	Запись в аккумулятор содержимого ячейки памяти по адресу в B2, B3	3 — 13
SHLD<B2><B3>	Занесение содержимого регистров H и L в память: содержимое L пересылается в ячейку по адресу в B 2, B3; содержимое регистра H - в ячейку, адрес которой на единицу больше	3 - 16
LHLD<B2><B3>	Загрузка в регистры H и L содержимого ячеек памяти: в L пересылаются данные из ячейки памяти по адресу в B2, B3; в H — из ячейки, адрес которой на единицу больше	3 - 16

Примечания [14]:

1. Флаг устанавливается при наличии заема в старший разряд, в противном случае сбрасывается.
2. Флаг устанавливается при наличии заема из старших четырех разрядов в младшие, в противном случае сбрасывается.
3. Флаг нуля Z устанавливается, если содержимое регистра или байта данных совпадает с содержимым аккумулятора; флаги знака и переноса S и C устанавливаются, если содержимое регистра или байта данных больше содержимого аккумулятора; флаг вспомогательного переноса AC устанавливается, если содержимое младших четырех разрядов регистра или байта данных больше содержимого соответствующих разрядов аккумулятора; флаг четности *P* устанавливается, если байт разности между содержимым аккумулятора и содержимым регистра или байта данных содержит четное число единиц.
4. Состояние флага равно значению выдвигаемого из аккумулятора двоичного разряда.
5. Большее значение (за косой чертой) указывает на число тактов при выполнении условий, меньшее — при невыполнении.
6. По команде POP PSW флаги устанавливаются в соответствии со значением разрядов слова, хранящегося в стеке; при выдаче данных из стека в другие регистры флаги не меняются.

Приложение 2. СХЕМА ПМ-ЭВМ

Приложение 2. Схема ПМ-ЭВМ



Приложение 3. Таблица элементов, используемых в схеме ПМ-ЭВМ

Обозначение микросхемы	Тип микросхемы	Выводы микросхем
		Питание
		+ 5 В общее
D1	КР580ИК80А	Данные в тексте
D2	КР580ГФ24	Данные в тексте
D3-D7, D31	К155ЛН1	14 7
D8	К155ЛА1	14 7
D9, DIG, D23	К589АП16	16 8
Dll, D24	К155ИД4	14 7
D12, D13	КР541РУ2	18 9
D14, D15	КР556РТ4	16 8
D16-D22, D29	К155ТМ7	5 12
D25	К155ЛА2	14 7
D26	К155ЛЕ1	14 7
D27, D28	К155ЛАЗ	14 7
D30	К155ТМ2	14 7

Примечания:
КВ - 9 МГц
С1 - 0,15 мкФ
R1-R6 - 1 кОм
R7 — R30 — в зависимости от типа VI — V24
R31-R34- 5,1 кОм
V1-V24 - АЛ102А, АЛ102Г, АЛ112А-М

СПИСОК
РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Нестеров П. В. Микропроцессоры 1. Архитектура и ее оценка. М.: Высшая школа, 1984.
2. Вайда Ф., Чакань А. Микро-ЭВМ: Пер. с венгер./ Под ред. В. В. Ста-шина. М.: Энергая, 1980.
3 Шаньгин В. Ф., Костин А. Е. Микропроцессоры 2. Организация вычислительных процессов на микро-ЭВМ. М.: Высшая школа, 1984.
4. Горячев А. В., Шишкевич А. А. Микропроцессоры 6. Информационно-управляющие вычислительные системы. М.: Высшая школа, 1984.
5. Клингман Э. Проектирование микропроцессорных систем. М.:
Мир, 1980.
6. Вуд А. Микропроцессоры в вопросах и ответах: Пер. с англ./ Под ред. Д. А. Поспелова. М.: Энергоатомиздат, 1985.
7. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
8. Трачик В. Дискретные устройства автоматики: Пер. с польск./ Под ред. Д. А. Поспелова. М.: Энергия, 1978.

9. Коффрон Дж. Технические средства микропроцессорных систем: Практический курс: Пер. с англ. М.: Мир, 1983.
10. Интегральные микросхемы: Справочник/ Под ред. Б. В. Тараб-рина. М.: Радио и связь, 1983.
11. Григорьев В. Л Программное обеспечение микропроцессорных систем. М.: Энергоатомиздат, 1984.
12. Назаров Н. А. Программатор для микросхем К556РЕ4// В помощь радиолюбителю. 1983. Вып. 83. С. 26-31.
13. Хоровиц П., Хилл У. Искусство схемотехники: Пер. с англ./ Под ред. М. В. Гальперина. М.: Мир, 1983. Т. 1, Т 2.
14. Алексенко А. Г., Галицын А. А., Иванников А. Д. Премирование радиоэлектронной аппаратуры на микропроцессорах: Программирование, типовые решения, методы отладки. М.: Радио и связь, 1984.
15. Иванов Б. Н. Самодельный блок питания// В помощь радиолюбителю. 1983. Вып. 84. С. 62-73.
16. Васильев Н. П., Горовой В. Р. Микропроцессоры 5. Аппаратурно-программные средства отладки. М.: Высшая школа, 1984.

ОГЛАВЛЕНИЕ

Научно-популярное издание

БУРЕЕВ ЛЕВ НИКОЛАЕВИЧ
ДУДКО АЛЕКСЕЙ ЛЬВОВИЧ
ЗАХАРОВ ВАЛЕРИЙ НИКОЛАЕВИЧ

Простейшая микро-ЭВМ. Проектирование. Наладка. Использование

Редактор *Д. А. Поспелов* Редактор издательства З. *И. Михеева*
Художественные редакторы *Т. А. Дворецкова, Ю. В. Созанская*
Технические редакторы Я. *Н. Хотулева, О. Д. Кузнецова*
Корректор *Л. С. Тимохова*

ББК 32.97 Б 91 УДК 681.31-181.48
Рецензент В. Ф. Корнюшко

Буреев Л. Н. и др.
В 91 Простейшая микро-ЭВМ: Проектирование. Наладка. Использование/ Л. Н. Буреев, А. Л. Дудко, В. Н. Захаров. — М.: Энергоатомиздат, 1989. — 216 с.: ил. — (Научно-попул. б-ка школьника).
ISBN 5-283-01482-7

На примере простейшей микро-ЭВМ рассматриваются возможности микропроцессорной техники. Построение описываемой в книге микро-ЭВМ доступно радиолюбителям - участникам школьного кружка. Составляющие микропроцессорной техники рассматриваются комплексно и иллюстрируются примерами.

Для широкого круга читателей, не обладающих специальной подготовкой в области электроники, вычислительной техники.

2405000000- 269
Б ---------- 2о4-оо
051 (01)-89
ISBN 5-283-01482-7
ББК 32.97

ИБ № 1723

Набор выполнен в издательстве. Подписано в печать с оригинала-макета 18.11.88. Т-20630. Формат 84 х 108 1/32. Бумага офсетная № 1. Гарнитура Пресс Роман. Печать офсетная. Усл. печ. л. 11,34. Усл. кр.-отт. 45,99. Уч.-изд. л. 12,95. Тираж 160 000 экз. Заказ 99, Цена 85 к.
Энергоатомиздат, ИЗ 114, Москва, М-114, Шлюзовая наб., 10.
Предприятие малообъемной книги дважды ордена Трудового Красного Знамени Ленинградского производственного объединения «Типография имени Ивана Федорова» Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 192007, Ленинград, ул. Боровая, 51.

OCR Pirat