Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра автоматического управления

А.П. Пашкевич, О.А. Чумаков, С.В. Лукьянец

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Конспект лекций
для студентов специальностей
I – 53 01 07 «Информационные технологии и управление
в технических системах»
дневной формы обучения

В 2-х частях

Часть 1

Минск 2005

УДК 681.3.06 (075.8)
ББК 32.97 я 73
П 22

Р е ц е н з е н т ы :
доцент кафедры ЭВМ БГУИР
канд. техн. наук Н.И. Силков,
зав. кафедрой автоматизации производственных процессов
и электротехники БГТУ
канд. техн. наук, доцент И.Ф. Кузьмицкий

Пашкевич А.П.

П 22 Микропроцессорные системы управления: Конспект лекций для студ. спец. I-53 01 07 «Информационные технологии и управление в технических системах» дневн. формы обуч.: В 2 ч. Ч. 1 / А.П. Пашкевич, О.А. Чумаков, С.В. Лукьянец. – Мн.: БГУИР, 2005.– 68 с.: ил.

ISBN 985-444-864-9 (ч. 1)

В конспекте лекций рассматривается комплекс вопросов, связанных с теорией, проектированием, технической реализацией и применением микропроцессоров и микроЭВМ в цифровых системах управления. Приводятся сведения по таким основным компонентам микропроцессорных систем, как память, устройства ввода/вывода, и др. На конкретных примерах раскрываются принципы программной реализации алгоритмов управления техническими системами.

УДК 681.3.06 (075.8)
ББК 32.97 я 73

ISBN 985-444-864-9 (ч. 1)
ISBN 985-444-865-7

СОДЕРЖАНИЕ

Введение

Использование микропроцессоров и микроЭВМ существенно повышает уровень автоматизации процессов управления в технических системах. Функциональная гибкость, высокая надежность, малые габариты и стоимость микропроцессорных средств обусловили целесообразность их применения в различной аппаратуре, в том числе в системах локальной автоматики. В связи с этим существенно изменился процесс проектирования цифровых систем управления. Это вызвано как использованием более сложных функциональных компонентов, так и применением новых архитектурных решений, основанных на замене некоторых аппаратных средств программными модулями. Поэтому проектирование современных средств автоматизации требует от разработчика знания вычислительной техники и программирования на качественно новом уровне, с учетом специфики объектов управления в технических системах.

В то же время при проектировании цифровых систем управления на базе микроЭВМ разработчику приходится решать задачи, многие из которых возникают и при проектировании классической вычислительной техники. Среди них – организация процессорных элементов и обеспечение их взаимодействия с памятью, построение каналов обмена информацией между микроЭВМ и внешними устройствами, согласование функционирования элементов системы, имеющих различную скорость работы.

В данном курсе лекций рассматриваются архитектура, принципы функционирования и обработки информации в микропроцессорных системах управления. Приводятся базовые сведения о построении подсистемы памяти, организации ввода/вывода информации и системе команд восьмиразрядного микропроцессора. В заключительном разделе раскрываются принципы программной реализации алгоритмов генерации импульсов, двухпозиционных и пропорциональных регуляторов, а также алгоритмов контурного управления. Поскольку происходящее сегодня быстрое обновление технических средств делает нецелесообразным описание конкретных устройств, то в данном курсе лекций изложены базовые принципы построения микропроцессорных систем управления.

1. Цифровые системы управления на базе микропроцессоров и микроконтроллеров

1.1. Принципы построения цифровых систем управления

В современных системах управления широко применяются микропроцессоры, микроконтроллеры и другие средства вычислительной техники. Такая элементная база позволяет унифицировать аппаратные средства, а также реализовать сложные алгоритмы управления, обеспечивающие высокие качественные характеристики для широкого круга технических объектов.

Структура аппаратных средств типовой микропроцессорной системы управления приведена на рис. 1. Система содержит микроЭВМ, в состав которой входят микропроцессор, память, а также устройства связи с объектом (интерфейсы), которые обеспечивают преобразование цифровых кодов на выходе микроЭВМ в сигналы, воспринимаемые объектом управления, а также преобразование выходных сигналов датчиков в двоичные коды, поступающие в микроЭВМ. Такое управляющее устройство с некоторым тактом (интервалом квантования) на основании информации о текущем состоянии объекта, а также командных сигналов рассчитывает управляющее воздействие.

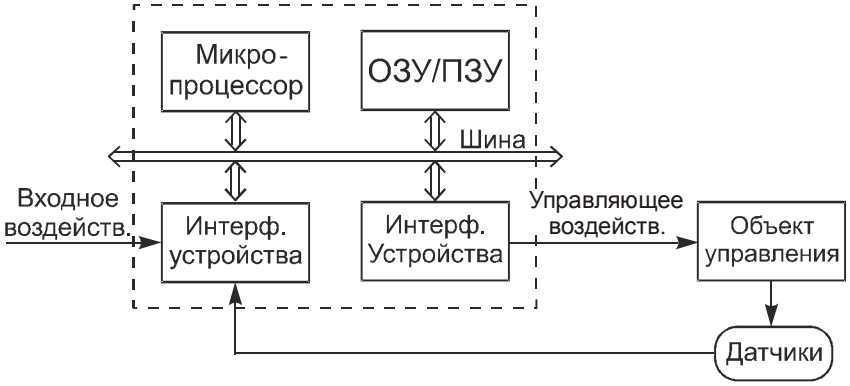


Рис. 1. Система управления на базе микроЭВМ

При построении систем управления на базе микропроцессоров и микроЭВМ следует учитывать следующие особенности программной реализации алгоритмов управления:

- запаздывание, вносимое микроЭВМ;
- временное и амплитудное квантование сигналов;
- возможности реализации сложных логических и вычислительных процедур, обеспечивающих адаптацию к изменениям параметров объектов и возмущающих воздействий.

Следует заметить, что цифровые системы управления на базе микропроцессоров и микроЭВМ иногда называют устройствами на основе «программируемой» (гибкой) логики, что означает возможность их

перенастройки для управления различными объектами путем изменения программы. Альтернативным методом построения цифровых систем управления является использование «жесткой» логики, при которой ядром устройства является заказная БИС, реализующая алгоритмы управления для узкого круга объектов. Очевидно, что второй метод на сегодняшний день является более дорогостоящим, однако он широко используется при массовом производстве устройств управления для типовых объектов либо при создании систем управления объектами специального назначения. Для промышленного применения наиболее рациональным является первый метод, основанный на применении микропроцессоров и микроконтроллеров.

1.2. Эволюция средств вычислительной техники

История создания программируемых вычислительных устройств началась еще в 1830 г. с идеи, предложенной английским математиком Чарльзом Бэбиджем (Charles Babage). Практическое же появление вычислительных машин стало возможным только в XX в. в связи с развитием электроники. При этом можно проследить несколько этапов развития ЭВМ.

Первый этап (до 1955 г.) – *ламповые* ЭВМ, масса которых достигала 30 т, число электронных ламп – 18 тыс., потребляемая мощность – 150 кВт (мощность, достаточная для небольшого завода), объем памяти – 20 10-разрядных десятичных чисел, время выполнения операций: сложения – 0,0002 с, умножения – 0,0028 с. Числа в ЭВМ вводились с помощью перфокарт и набора на переключателях, а программа задавалась соединением гнезд на специальных наборных полях. Для ускорения процесса подготовки программ стали создавать первые языки автоматизации программирования (языки символического кодирования).

Второй этап (до 1965 г.) – появление ЭВМ, построенных *на транзисторах*, привело к уменьшению их габаритов, массы, энергопотребления и стоимости, а также к увеличению надежности и производительности. На этом этапе были созданы специальные алгоритмические языки для инженерно-технических и экономических расчетов, а также операционные системы (комплексы служебных программ, обеспечивающих лучшее распределение ресурсов ЭВМ при исполнении пользовательских задач). Эволюция операционных систем шла в направлении обработки пакетов заданий, а также мультипрограммного режима обработки данных.

Третий этап (до 1970 г.) – прогресс в области технологии производства интегральных микросхем (ИС) позволил повысить производительность и снизить стоимость универсальных ЭВМ, а также создать малогабаритные, простые, дешевые и надежные машины – мини-ЭВМ. Мини-ЭВМ первоначально предназначались для замены аппаратно-реализованных контроллеров (*устройств управления*) в контуре управления каким-либо объектом, в автоматизированных системах управления технологическими процессами, системах сбора и обработки

экспериментальных данных, различных управляющих комплексах на подвижных объектах и т.д. Низкая цена серийной мини-ЭВМ, большое число серийных устройств связи с объектом управления и хорошее программное обеспечение обусловили экономическую эффективность использования таких устройств.

Четвертый этап (до 1978 г.) – успехи в развитии электроники привели к созданию больших интегральных схем (БИС), где в одном кристалле размещалось несколько десятков тысяч электрических элементов. Это позволило разработать более дешевые ЭВМ, имеющие большую память и меньший цикл выполнения команды. Но так как затраты на программирование оставались высокими, то на первый план вышла задача экономии человеческих, а не машинных ресурсов.

В 1971 г. фирмой Intel был изготовлен первый микропроцессор i4004 – БИС, в котором полностью размещался простой четырехразрядный процессор. Появились управляющие устройства, построенные на одной или нескольких БИС, содержащих процессор, память, схемы сопряжения с датчиками и исполнительными органами в объекте управления.

Пятый этап (с 80-х г.г.) – улучшение технологии производства БИС позволило изготавливать дешевые электронные схемы, содержащие сотни тысяч элементов в кристалле – схемы сверхбольшой степени интеграции (СБИС). Появилась возможность создать устройство с габаритами массового телевизора, в котором размещались микроЭВМ, клавиатура, экран, дисковый накопитель, а также схемы сопряжения с малогабаритным печатающим устройством, измерительной аппаратурой, другими ЭВМ и т.д. Наиболее широкое применение нашли микроЭВМ в гибких системах автоматизации производства и научных исследований. В настоящее время развитие идет в направлении как повышения вычислительной мощности компьютеров, так и создания однокристальных микроконтроллеров, ориентированных на управление различными техническими объектами.

2. Архитектура управляющей микроЭВМ

2.1. Основные понятия и определения

Микропроцессор – программно-управляемое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки. Обычно микропроцессор (МП) состоит из одной (иногда нескольких) интегральных схем и имеет доступ к внешней памяти. Он также обеспечивает передачу информации между компонентами ЭВМ и внешней средой. МП является основой любой микроЭВМ и производится по технологии больших интегральных схем (БИС).

МикроЭВМ – вычислительное или управляющее устройство, содержащее микропроцессор, оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), таймер, порты ввода/вывода, генератор тактовых импульсов, блок питания и другие элементы.

Однокристальный микроконтроллер – БИС, содержащая в себе МП, ОЗУ, ПЗУ небольшого объема, таймеры/счетчики, порты ввода/вывода, и ориентированная на решения задач управления. Появлению микроконтроллеров способствовало совершенствование технологии производства микроэлектроники, что позволило интегрировать на одном кристалле большинство функциональных блоков управляющей микроЭВМ. В большинстве микроконтроллеров используется восьмиразрядное вычислительное ядро с упрощенной системой команд. Память физически и логически разделена на память программ и память данных. Подсистема ввода/вывода микроконтроллеров часто имеет аналого-цифровые и цифроаналоговые преобразователи для возможности ввода сигналов от датчиков и вывода сигналов на исполнительное устройство.

Цифровой процессор сигналов (Digital Signal Processor, DSP) – это специализированный микроконтроллер, ориентированный на решение задач цифровой фильтрации в режиме реального времени. Обычно DSP имеет мощное вычислительное ядро, спроектированное, однако, только для решения узкоспециализированных задач. По этой причине сигнальные процессоры имеют сравнительно невысокую стоимость по сравнению с обычной микроЭВМ, реализующей аналогичные функции. Применяемые, например в области телекоммуникаций DSP имеют производительность до 1,6 млрд. операций/с. С архитектурной точки зрения такие процессоры могут представлять собой аналоговые функциональные преобразователи сигналов. Часто они выполняют функции аналоговых схем (например производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т.д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т.д.).

2.2. Архитектура микропроцессора

Особенность МП как устройства с программируемой логикой заключается в подчиненности его аппаратного состава (или структуры совокупности элементов, составляющих МП, и связей между ними) принципу программируемости. Это означает, что функции, реализуемые МП, определяются не столько его структурой, сколько последовательностью управляющих слов (команд), поступающих из программной памяти на входы МП. При изменении этой последовательности изменяется и функция, выполняемая МП. Поэтому разработчик микроЭВМ при анализе функциональных возможностей МП должен учитывать не только его структуру, но и возможности программной реализации функций. Для комплексной характеристики возможностей МП будем пользоваться понятием архитектуры.

Под *архитектурой* процессора понимается его абстрактное представление в терминах основных функциональных модулей, т.е. его логическая организация. В это понятие входят структура, перечень программно-доступных элементов, система команд, методы адресации и т.д. Архитектура не определяет особенности технологии, аппаратной реализации, временные диаграммы, тактовую частоту, параметры электропитания и т.д. Одним словом, термин «архитектура» используется для описания возможностей, предоставляемых МП, а термин «организация» определяет, как эти возможности реализованы.

Все ЭВМ содержат следующие функциональные блоки, имеющие свою архитектуру: процессор, состоящий из арифметико-логического устройства (АЛУ) и устройства управления (УУ), память (оперативная – ОЗУ и постоянная – ПЗУ), устройства ввода/вывода информации. Устройство управления формирует управляющие сигналы Y_i для работы РОН и АЛУ, а те, в свою очередь, формируют признаки (или флаги) P_i. Объединение функциональных блоков в ЭВМ осуществляется посредством системы шин: шины данных, по которой осуществляется обмен данными между блоками ЭВМ, шины адреса, используемой для передачи адресов, по которым осуществляется обращение к различным устройствам ЭВМ, и шины управления для передачи управляющих сигналов (рис. 2).

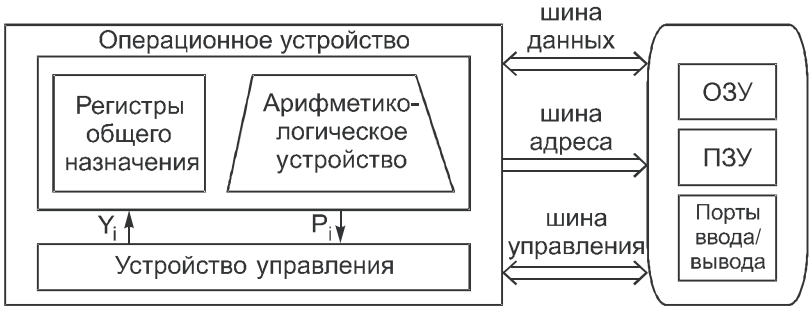


Рис. 2. Обобщенная структурная схема ЭВМ

Связь с пользователем осуществляется с помощью пульта управления, который позволяет выполнить такие действия, как пуск ЭВМ; останов, под

действием которого прекращается поступление сигналов с генератора тактовых импульсов и процессор переходит в состояние ожидания; загрузка начального адреса программы; ее пошаговое выполнение при отладке.

МП можно классифицировать по нескольким признакам:

1. По **количеству БИС** различают однокристальные, многокристальные и многокристальные секционные микропроцессоры.

В *однокристальном* МП все его аппаратные средства реализованы в виде одной БИС, имеющей фиксированную разрядность и жесткую систему команд. Параметры однокристальных микропроцессоров улучшаются по мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса. Однако возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса. Поэтому при разбиении его логической структуры на функционально законченные части можно реализовать процессор в виде нескольких работающих автономно БИС с фиксированной разрядностью и гибкой системой команд. Такой МП называется *многокристальным*.

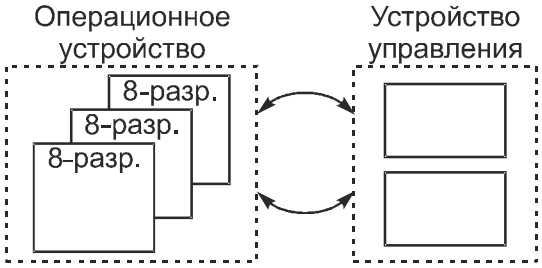


Рис. 3. Многокристальный секционный процессор

При создании высокопроизводительных многоразрядных микропроцессоров требуется множество аппаратных средств, не реализуемых в доступных БИС, поэтому возникает необходимость в дальнейшем функциональном разбиении логической структуры МП вертикальными и горизонтальными плоскостями. В результате такого разделения на конструктивно законченные части создаются условия реализации каждой функции в виде отдельной БИС. Все они образуют комплект *многокристального секционного микропроцессора* (рис. 3) с произвольной разрядностью и гибкой системой команд. Таким образом, микропроцессорная секция – это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций.

2. По **организации внешних шин** различают микропроцессоры с *раздельными* или *совмещенными шинами* адреса и данных.

3. По **организации внутренних шин** различают микропроцессоры с одной, двумя и тремя внутренними шинами.

В *одношинных* МП все его элементы имеют одинаковый интерфейс и подключены к единой информационной шине, по которой передаются коды данных, адресов и управляющих сигналов. В многошинных процессорах его элементы группами подключаются к своей информационной шине. Это позволяет осуществить одновременную передачу информационных сигналов по нескольким (или всем) шинам. Такая организация систем усложняет их конструкцию, однако, увеличивает производительность.

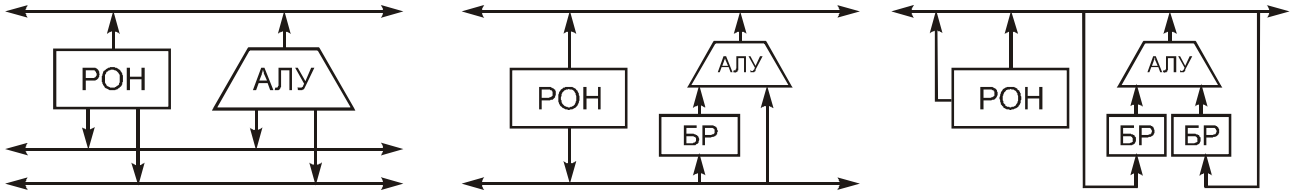
4. По **организации стека** различают микропроцессоры со *встроенным*, расположенным на кристалле МП и *автономным стеком*, который реализован в оперативной памяти.

5. По **характеру временной организации работы** микропроцессоры делят на синхронные и асинхронные.

В *синхронных* микропроцессорах начало и конец выполнения операций задаются устройством управления (время выполнения операций в этом случае не зависит от вида выполняемых команд и величин операндов). *Асинхронные* микропроцессоры позволяют начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции. Для более эффективного использования каждого устройства микропроцессорной системы в состав асинхронно работающих устройств вводят электронные цепи, обеспечивающие автономное функционирование устройств. Закончив работу над какой-либо операцией, устройство вырабатывает сигнал запроса, означающий его готовность к выполнению следующей операции. При этом роль естественного распределителя работ принимает на себя память, которая в соответствии с заранее установленным приоритетом выполняет запросы остальных устройств по обеспечению их командной информацией и данными.

2.3. Организация шин микропроцессорных систем

Все функциональные блоки, как в самом микропроцессоре, так и в микроЭВМ, объединяются с помощью набора проводников, называемого шиной. Посредством шины данных осуществляется обмен информацией между блоками ЭВМ. Шина адреса используется для передачи адресов, по которым осуществляется обращение к различным устройствам ЭВМ. Шина управления необходима для передачи управляющих сигналов. Различают внутренние (в самом микропроцессоре) и внешние шины, а также МП с одной, двумя и тремя внутренними шинами (рис. 4).



а б в

Рис. 4. Топология:
а – трехшинная, б – двухшинная, в – одношинная

Трехшинная топология не требует буферных регистров, поэтому возможно выполнение арифметических и логических операций за один такт, включая выборку операндов из РОН и запись результатов в один из регистров. Этот способ помимо высокого быстродействия имеет еще одно важное достоинство –

отсутствие буферных регистров. Главный недостаток такой топологии заключается в значительной занимаемой площади шин на кристалле (до 18 %).

Рационального баланса между числом внутренних шин и числом элементов микропроцессора можно достичь при двухшинной организации, которая при меньшей площади, занимаемой шинами на кристалле, требует введения по меньшей мере одного буферного регистра. Это значит, что арифметические и логические операции в таком МП будут выполняться не менее чем за два такта:

1) загрузка буферного регистра одним из операндов;

2) выполнение операции в АЛУ над содержимым буферного регистра и одного из РОН; запись результата в РОН.

Наконец, возможна организация МП на основе только одной шины. Наименьшая площадь, занимаемая шиной по сравнению с рассмотренными выше вариантами, позволяет в максимальной степени усложнить архитектуру МП при фиксированной площади кристалла. Однако необходимость введения не менее двух буферных регистров увеличивает цикл выполнения операций уже до трех тактов:

1) загрузка буферного регистра одним из операндов;

2) загрузка второго буферного регистра вторым операндом;

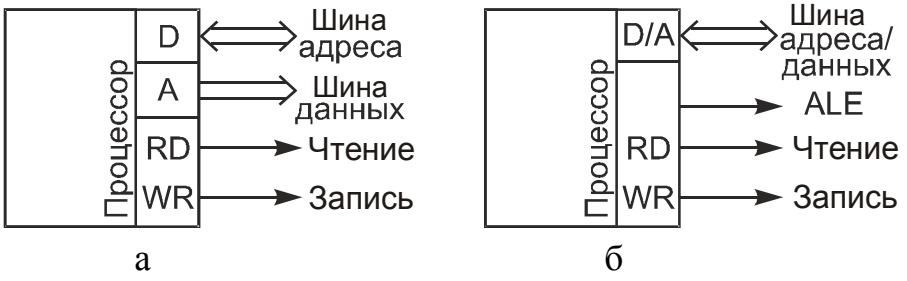
3) выполнение операции в АЛУ над содержимым буферных регистров и запись результата в РОН.

При построении вычислительной системы необходимо учитывать, что выходные линии шин МП позволяют подключать не более одной-двух TTL-нагрузок, поэтому необходимо использовать внешние буферные усилители (шинные формирователи). Внешние шины обеспечивают связь микропроцессора с ОЗУ, ПЗУ и портами вода/вывода. При их организации существует два ограничения:

1) на количество внешних выводов БИС;

2) на нагрузочную способность линий.

Ограничение количества внешних выводов корпуса микросхемы привело к появлению микропроцессоров с совмещенной (мультиплексированной) шиной (рис. 5), по которой в разные моменты времени передаются сигналы как адреса, так и данных.



а б

Рис. 5. Микропроцессор:
а – с раздельными шинами, б – с совмещенными шинами

При использовании шинной организации как внутри кристалла, так и при подключении нескольких БИС к одной шине возникает дополнительная трудность, связанная со способом связи нескольких элементов с одним проводником общей шины. В качестве примера проанализируем способы организации общей шины в МП, выполненном по схеме на рис. 4, в). С каждым проводником общей шины связаны три входа (РОН, буферный регистр и регистр сдвига) и два выхода (РОН, АЛУ). Известны три способа решения этой задачи (рис. 6): логическим объединением, объединение с помощью схем с открытым коллектором и объединение с использованием схем с тремя устойчивыми состояниями.

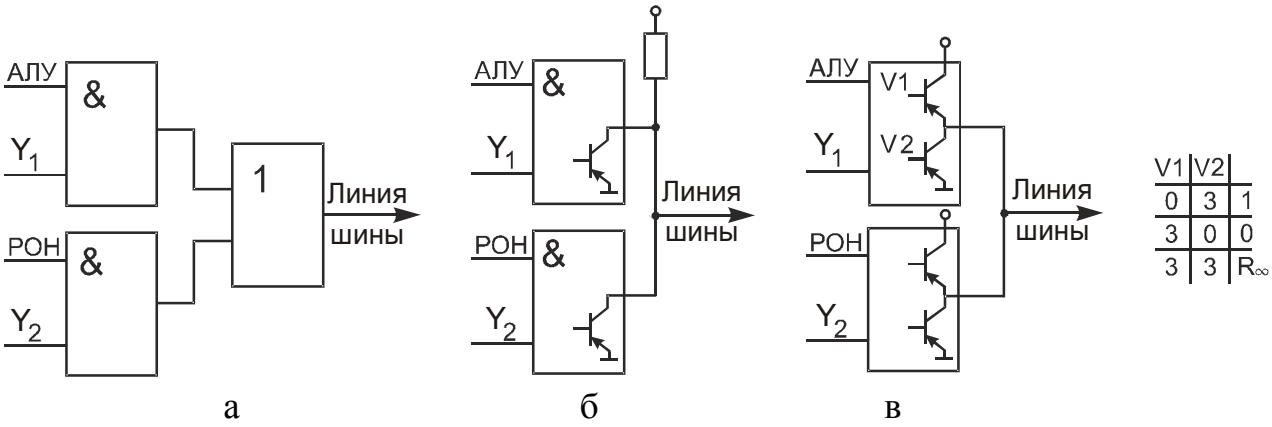


Рис. 6. Подключение нескольких источников к одной шине:
а – логическим объединением, б – «монтажным ИЛИ», в – объединением на базе схем с тремя устойчивыми состояниями

Логическое объединение (рис. 6, а) выполняется с помощью схемы ИЛИ, на входы которой при подаче управляющего сигнала Y_1 на общей шине появляется выходной сигнал РОН, а при подаче сигнала Y_2 – выходной сигнал АЛУ. Этот способ может использоваться при создании внутренней шины (на кристалле). Объединение с помощью *схем с открытым коллектором* характеризуется электрическим соединением выходов нескольких логических элементов. Поэтому этот способ часто называют «монтажным ИЛИ» (рис. 6, б). При простом соединении выходов элементов отпадает необходимость в схеме ИЛИ, используемой при логическом соединении, а следовательно, нет принципиального ограничения на число объединяемых выходов. Это позволяет применять данный способ при организации не только внутренних, но и внешних шин, учитывая, однако, что количество объединяемых линий ограничено конечным сопротивлением закрытых транзисторов, что фактически ограничивает область применимости этого способа организации.

Логическим его развитием, устраняющим указанный недостаток, является использование в качестве нагрузочного резистора нелинейного элемента. Объединение *с использованием схем с тремя состояниями* отличается именно таким характером нагрузки. Третье состояние обеспечивается, когда оба транзистора одного каскада закрыты. Этот способ широко применяются при

организации внешних шин, которые реализуются в виде дорожек печатной платы или плоского кабеля.

2.4. Обработка информации в микропроцессоре

Рассмотрим процесс обработки информации в микропроцессоре на примере упрощенной схемы, представленной на рис. 7.

Микропроцессор имеет:

• РОН – регистры общего назначения, которые используются для хранения данных и адресов. Кроме РОН в МП имеются специальные регистры: счетчик команд, указатель стека и др.;

• БР – буферный регистр, используется для промежуточного хранения операнда, иногда выполняет функции регистра сдвига;

• АЛУ – комбинационная схема, построенная на логических элементах И, ИЛИ, НЕ, выполняющая арифметические и логические преобразования информации.

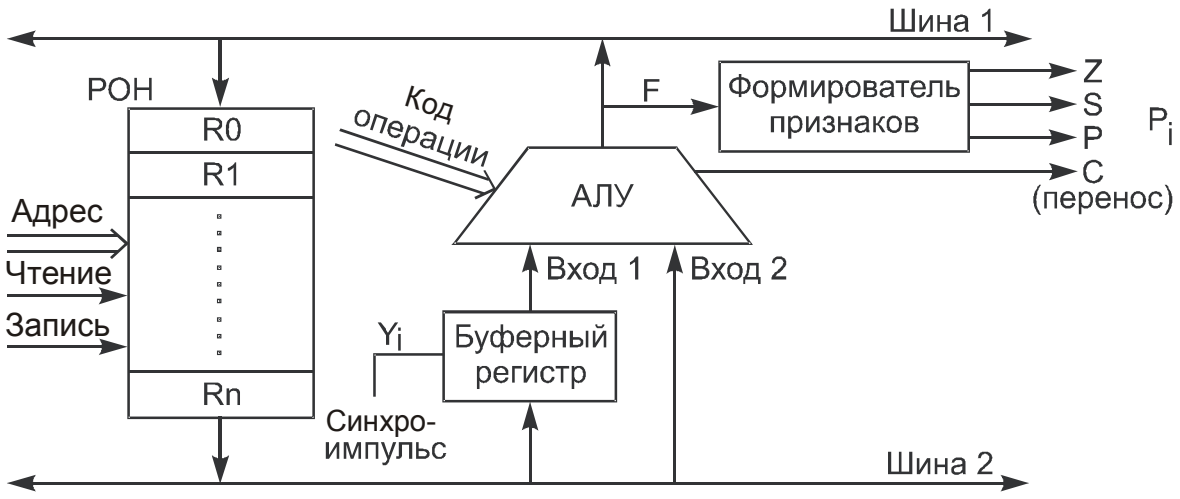


Рис. 7. Упрощенная схема микропроцессора

Выполнение команд разбивается на такты и осуществляется в операционном устройстве под управлением УУ. Рассмотрим пример выполнения операции сложения $R_0 + R_1 \rightarrow R_0$:

Такт 1. $R_0 \rightarrow$ Шина 2$\rightarrow$БР ; данные R_0 по шине 2 передаются в БР

Такт 2. $R_1 \rightarrow$ Шина 2$\rightarrow$ АЛУ$_{Вх}$; данные R_1 по шине 2 передаются в АЛУ$_{Вх}$

Такт 3. Сложение в АЛУ

Такт 4. АЛУ$_F \rightarrow$ Шина 1$\rightarrow P_0$; результат АЛУ$_F$ по шине 1 передается в P_0

Эта схема универсальна, т.к. позволяет выполнять множество различных операций. Для работы операционного устройства необходимо формировать управляющие сигналы, обеспечивающие выдачу синхроимпульсов для чтения, записи, адресацию РОН и настройку АЛУ на конкретную операцию. При выполнении арифметических, логических и некоторых других операций в

специальном регистре слова-состояния программы формируются признаки (флаги) результата, необходимые для операций условного перехода. Реализация более сложных команд требует использования специальных блоков (умножения, деления, и др.). Для ускорения выполнения определенных операций вводятся дополнительно специальные операционные узлы (например циклические сдвигатели).

2.5. Управление обработкой информации

Управление операциями в ЭВМ осуществляет устройство управления с помощью управляющих сигналов, генерируемых по командам программы. Коды операции команд программы, воспринимаемые управляющей частью микропроцессора, расшифрованные и преобразованные в ней, дают информацию о том, какие операции требуется выполнить, где в памяти расположены данные, куда необходимо направить результат и где расположена следующая за выполняемой команда.

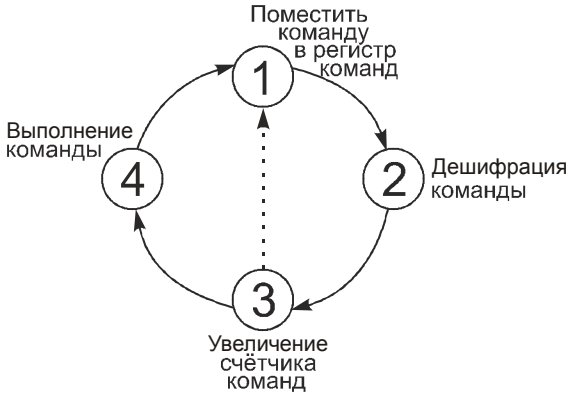


Рис. 8. Процесс выполнения команды в МП

В соответствии с циклом фон Неймана (рис. 8), устройство управления в рамках тактовых интервалов (такт – минимальный рабочий интервал, в течение которого совершается одно элементарное действие; цикл – интервал времени, в течение которого выполняется одна машинная операция) осуществляет: *выборку команды*; ее *дешифрацию* с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов); установление временной последовательности всех функциональных управляющих сигналов; генерацию управляющих импульсов и передачу их на управляющие шины функциональных частей микроЭВМ и логические элементы между ними; анализ результата операции и *формирование адреса следующей команды*; *выполнение команды*. При этом фаза выборки может повторяться, если команда имеет длину больше одного байта.

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: *аппаратный* и *микропрограммный*.

Выполнение операций в машине сводится к элементарным преобразованиям информации (передача информации между узлами в блоках, сдвиг информации в узлах, логические поразрядные операции, проверка условий и т.д.) в логических элементах, узлах и блоках под воздействием функциональных управляющих сигналов устройства управления. Элементарные преобразования, не разложимые на более простые, выполняются в течение одного такта сигналов синхронизации и называются *микрооперациями*.

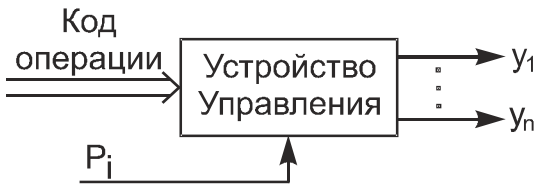


Рис. 9. Аппаратное устройство управления

В *аппаратных* (схемных) устройствах управления (рис. 9) каждой операции соответствует свой набор логических схем, вырабатывающих определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени. Для каждого КОП Y_i выдаются в течение нескольких тактов. УУ оптимизируется для конкретной системы команд, которую в дальнейшем изменить нельзя. При этом способе построения устройства управления реализация микроопераций достигается за счет однажды соединенных между собой логических схем, поэтому ЭВМ с аппаратным устройством управления называют ЭВМ с жесткой логикой управления. Это понятие относится к фиксации системы команд в структуре связей ЭВМ и означает практическую невозможность каких-либо изменений в системе команд ЭВМ после ее изготовления.

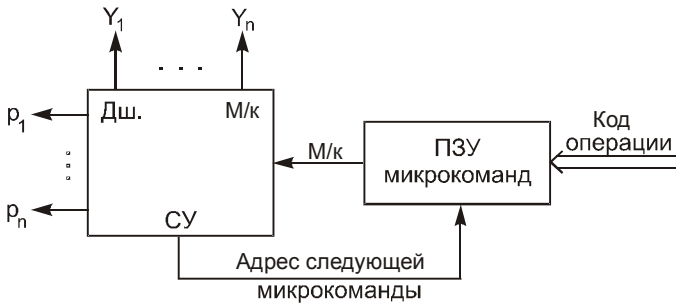


Рис. 10. Микропрограммное устройство управления:

М/к – микрокоманда;

ДШ – дешифратор;

СУ – схема управления

При *микропрограммной* реализации устройства управления (рис. 10) в состав последнего вводится ЗУ, каждый разряд выходного кода которого определяет появление функционального сигнала управления. Поэтому каждой микрооперации ставится в соответствие свой информационный код – микрокоманда. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Набор микроопераций называют микропрограммами. Способ управления операциями путем последовательного считывания и интерпретации микрокоманд из ЗУ (наиболее часто в виде микропрограммного ЗУ используют быстродействующие программируемые логические матрицы), а также использования кодов микрокоманд для генерации функциональных управляющих сигналов называют микропрограммным, а микроЭВМ с таким способом управления – микропрограммной, или с *гибкой логикой управления*. При этом Y_i для всех команд и всех тактов записываются в ПЗУ в виде микрокоманд. Кроме того, в каждую микрокоманду записывается адрес следующей микрокоманды. Микрокоманды имеют разрядность 50…100, поэтому для сокращения длины их шифруют, а затем дешифрируют. Адрес первой микрокоманды определяет КОП, а каждая микрокоманда соответствует одному такту.

К микропрограммам предъявляют требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, а второе связано с желанием уменьшить объем используемого оборудования. Учет фактора быстродействия ведет к расширению микропрограмм, поскольку усложнение последних позволяет сократить время выполнения команд программы.

2.6. Архитектура 8-разрядного микропроцессора

Типичным представителем 8-битных однокристальных микропроцессоров является разработанный фирмой Intel в 1974 г. i8080 или его советский аналог К580ИК80. Кристалл процессора производится по технологическим нормам 6 мкм, вмещает 6000 транзисторов и имеет тактовую частоту 2 МГц, а более поздний его вариант i8080A (КР580ВМ80А) – 2,5 МГц. Процессор снабжен 8-разрядной шиной данных, 16-разрядной шиной адреса, с помощью которой адресует $2^{16} = 64$ Кбайт памяти, 256 устройств ввода и 256 устройств вывода. МП работает от трех источников питания – +5, +12 и –5 В и рассеивает мощность 1,25 Вт. Длительность такта при частоте 2 МГц составляет 0,5 мкс, при этом быстродействие – 500 000 коротких операций (регистр-регистр) в секунду.

Структурная схема КР580ВМ80А представлена на рис. 11.

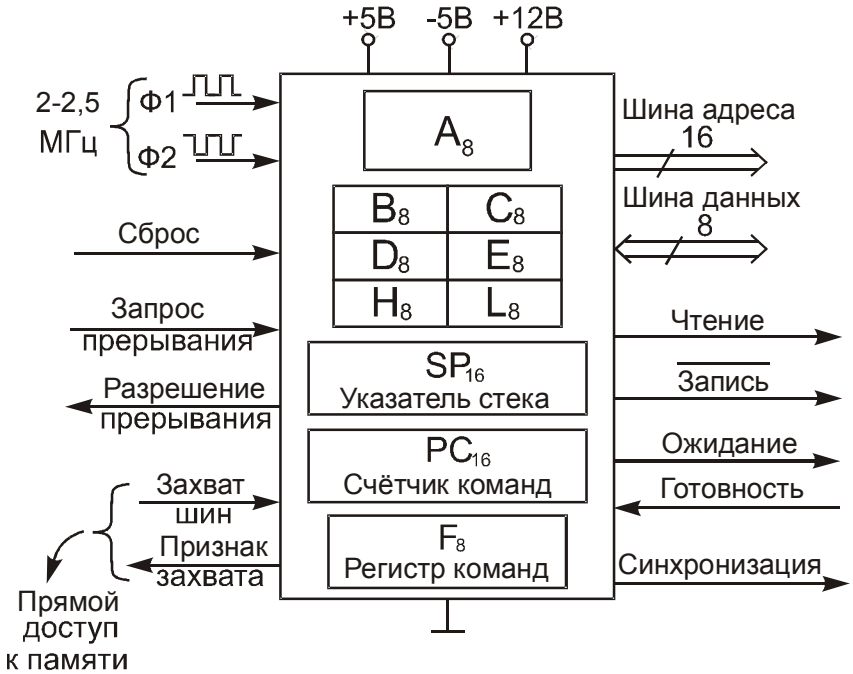


Рис. 11. Структурная схема КР580ВМ80А

Арифметическо-логическое устройство (АЛУ) обеспечивает выполнение арифметических, логических операций и операций сдвига над двоичными данными, представленными в дополнительном коде, или над двоично-десятичными данными. Устройство содержит схему десятичной коррекции, позволяющую производить операции десятичной арифметики. По результатам операций в АЛУ формируется ряд признаков, которые записываются в регистр

флагов F. Признак переноса C устанавливается в единицу, если в результате выполнения команды появляется перенос из старшего разряда. Дополнительный признак переноса AC устанавливается в единицу при возникновении переноса из третьего разряда. Используется в командах десятичной арифметики. Признак четности P устанавливается в единицу, если число единиц в разрядах результата четное. Признак нуля Z устанавливается в единицу, если результат равен нулю. Признак знака S указывает знак числа и равен единице, если число отрицательное, или нулю, если число положительное.

Блок регистров производит прием, хранение и выдачу различной информации, участвующей в процессе выполнения программы, и содержит счетчик команд (Program Counter, PC), указатель стека (Stack Pointer, SP), регистры общего назначения (РОН), регистры временного хранения и регистр адреса. 16-разрядный счетчик команд хранит текущий адрес команды. Содержимое счетчика команд автоматически увеличивается после выборки каждого байта команды. 16-разрядный указатель стека содержит начальный адрес памяти, используемый для хранения и восстановления содержимого программно-доступных регистров МП. Содержимое указателя стека уменьшается, когда данные загружаются в стек, и увеличивается, когда данные выбираются из стека. 8-разрядные регистры общего назначения B, C, D, E, H, L могут применяться как накопители и указатели (16-разрядный адрес операнда определяется содержимым пары регистров). Основной однобайтный регистр процессора – это аккумулятор A. Над его содержимым выполняется наибольшее количество арифметических и логических команд, а также команды ввода/вывода. Регистры временного хранения W, Z используются для приема и временного запоминания второго и третьего байт команд переходов, передаваемых с внутренней магистрали ЦПУ в счетчик команд. Эти регистры являются программно-недоступными. 16-разрядный регистр адреса принимает и хранит в течение одного машинного цикла адрес команды или операнда и выдает его через буфер адреса на однонаправленную выходную магистраль A0-A15. Буфер адреса выполнен в виде выходных формирователей, имеющих на выходе состояние «Отключено».

Устройство управления формирует комплекс управляющих сигналов, организующих выполнение поступившей в МП команды, и состоит из регистра команд, программируемой логической матрицы (ПЛМ) и схемы управления узлами. 8-разрядный регистр команд осуществляет прием и хранение команды, поступающей по шине данных. Программируемая логическая матрица дешифрирует код операции команды и формирует микрооперации в соответствии с микропрограммой выполнения команды. Схема управления узлами вырабатывает для различных узлов микропроцессора необходимые управляющие сигналы. 8-разрядный буфер данных обеспечивает ввод команд и данных в МП, вывод данных через формирователи, имеющие высокоомное состояние.

Сигнал «Сброс» обнуляет счетчик команд PC (но не регистры), что эквивалентно выполнению программы с адреса 0000H.

Схема синхронизации и управления состояниями МП формирует машинные такты и циклы, которые координируют выполнение всех команд, и вырабатывает сигнал SYNC «Синхронизация», определяющий начало каждого машинного цикла. Для выполнения команды требуется 1–5 машинных циклов, каждый из которых может состоять из 3–5 тактов (Т1-Т5). Длительность каждого из них соответствует периоду следования противофазных тактовых импульсов Ф1, Ф2.

При выполнении команды *в такте Т1* содержимое счетчика команд выдается на адресную шину, а на шину данных – слово-состояние машинного цикла (ССМЦ), сопровождаемое сигналом синхронизации, по которому это слово записывается в буферный регистр внешних схем управления. ССМЦ позволяет различить 10 типов машинного цикла (чтение первого байта команды, ввод/вывод, чтение/запись в память, чтение/запись в порты ввода/вывода). *В такте Т2* содержимое счетчика команд увеличивается на единицу и осуществляется анализ управляющих сигналов «Готовность» и «Захват шин». *В такте Т3* данные принимаются из памяти по шине данных и записываются в регистр команд (если выполняется цикл выборки команды) или в один из регистров МП. *В тактах Т4 и Т5*, если они необходимы, выполняются действия над операндами. Если команда включает несколько циклов, то по завершении текущего машинного цикла процессор переходит к такту Т1 следующего цикла.

3. Система команд микропроцессора

3.1. Классификация команд

Под *командой* понимают совокупность сведений, необходимых процессору для выполнения определенного действия при реализации программы. Множество команд, реализуемых в ЭВМ, образует *систему команд*, выбор которой является важнейшей задачей проектирования ЭВМ. Система команд определяет область применения и эффективность микропроцессорной системы управления. Несмотря на то, что подавляющее большинство алгоритмов может быть реализовано посредством ограниченного набора команд, большинство ЭВМ имеет 60–120 базовых команд. Под базовой понимают команду, которая определяет выполняемую операцию без учета модификаций данной команды за счет использования различных режимов адресации. Например, МП КР580ВМ80А имеет 78 базовых команд, однако с учетом модификаций число команд равняется 224. Это позволяет в ряде случаев существенно сократить длину программ, а следовательно, уменьшить время решения задачи и размер программы в памяти. Таким образом, система команд определяет возможности машины.

Теоретически ограничения на число команд ЭВМ нет; например, при введении команд из нескольких слов можно выделить больше бит под код операции. Каждый дополнительный бит в коде операции удваивает число команд. С другой стороны, чем сложнее команда, тем быстрее выполняется программа из-за сокращений числа обращений к памяти.

Классификация команд по основным признакам представлена на рис. 12.

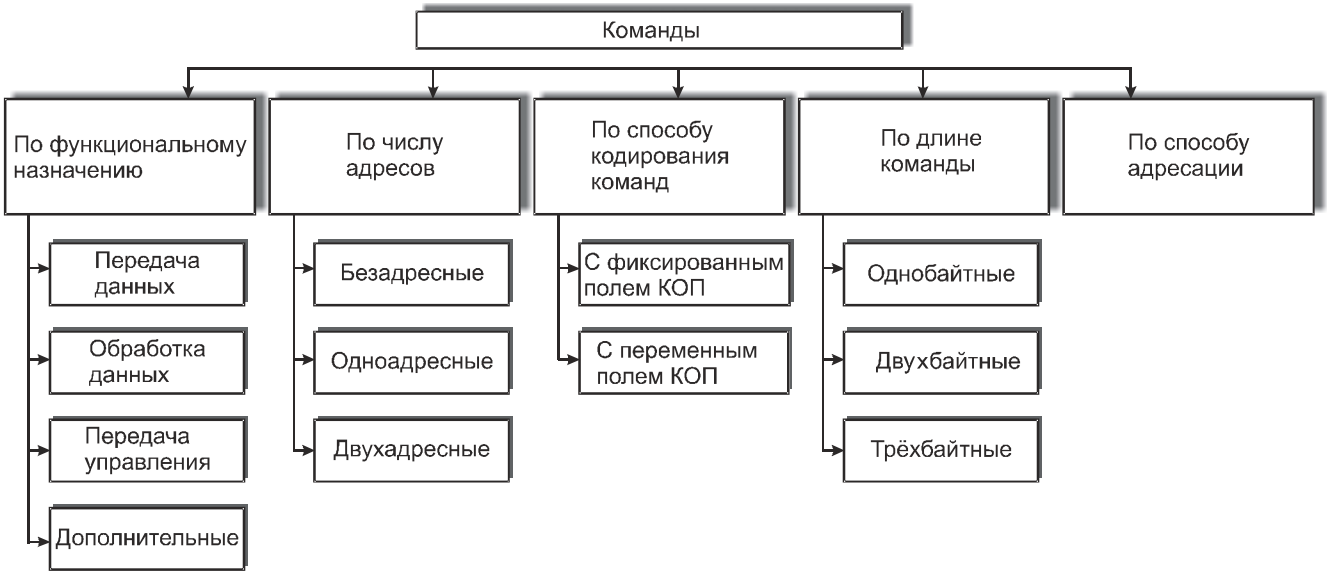


Рис. 12. Классификация команд

Систему команд рассматриваемого микропроцессора КР580ВМ80А можно классифицировать по трем основным признакам:

1. *Длине команды* (одно-, двух- и трехбайтные);

2. *Функциональному назначению* (передачи, обработки данных, команды управления);

3. *Архитектурным признакам* (операции с регистрами, памятью и портами).

Современные тенденции развития ЭВМ показывают, что фирмы-разработчики микропроцессоров стараются создавать дополнительные наборы команд на основе уже существующих, сохраняя программную преемственность с предыдущими поколениями процессоров. Такие ресурсоемкие задачи, как расчет трехмерной графики, компрессия/декомпрессия аудио-видеоданных и другие, используют дополнительные наборы команд (3DNow, MMX, SSE, и др.), оптимизированные под соответствующие приложения.

3.2. Методы адресации

Для взаимодействия различных модулей в микроЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда – отдельные адресные пространства регистров, устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. С другой стороны, при разработке микропроцессоров стараются использовать коды операций минимальной длины, что приводит к возникновению проблемы идентификации данных из-за короткого машинного слова. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти – это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес. В микропроцессоре КР580ВМ80А используется пять методов адресации:

1. **Прямая** – в команде задается адрес ячейки памяти, где расположен операнд; он указывается во втором (младшая часть адреса) и в третьем (старшая часть) байтах команды. К этой группе также относятся команды, в которых задается адрес порта ввода/вывода:

STA 8020H – требует четырех обращений к памяти;

IN 05H – требует двух обращений к памяти.

2. **Прямая регистровая** – в команде задается адрес регистра или пары регистров, где находится 8- или 16-битный операнд:

MOV A, B – требует одного обращения к памяти;

CMP C.

3. **Непосредственная** – операнд содержится в самой команде:

MVI A, 08H – требует двух обращений к памяти;

LXI M, 8020H – требует трех обращений к памяти.

4. **Косвенная** – адрес M ячейки памяти, где расположен операнд, определяется содержимым парного регистра, явно или неявно указанного в команде:

MOV A, M – пересылка в A из ячейки памяти, на которую указывает HL;

LDAX B – загрузка A из ячейки памяти, на которую указывает пара BC.

5. **Неявная** – адрес операнда не указывается в явном виде, а определяется кодом операции:

ADD B; A ← A+B, аккумулятор не задается в явном виде.

Следует отметить, что в одной команде могут использоваться два различных метода адресации, например, в команде MVI A, 08H используется прямая регистровая адресация для приемника и непосредственная для источника. В системах реального времени для повышения скорости вычислений программ необходимо максимально использовать *регистровую* адресацию.

3.3. Формат команд

Формат команды определяет ее структурные элементы, каждый из которых интерпретируется определенным образом при выполнении команды. Среди таких элементов (полей) выделяют:

- код операции, определяющий выполняемое действие;
- адрес ячейки памяти, регистра процессора, внешнего устройства;
- режим адресации;
- операнд при использовании непосредственной адресации;
- код анализируемых признаков для команд условного перехода.

Почти во всех форматах команд первые биты отводятся для кода операции, но далее форматы команд разных ЭВМ сильно отличаются друг от друга. Остальные биты должны определять операнды или их адреса, и поэтому они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

Число бит, отводимое под КОП, является функцией полного набора реализуемых команд. При использовании фиксированного числа бит под КОП для кодирования всех m команд необходимо в поле КОП выделить $\log_2 m$ двоичных разрядов. Так как информация берется только из одной ячейки, эта ячейка называется источником; ячейка, содержимое которой изменяется, называется приемником.

Средняя длина команды в типичной программе для 8-разрядного микропроцессора равна двум байтам, а для программ более поздних 16-разрядных процессоров типа i8086 она равна 4,1. Поэтому на программах с большим

количеством логических операций 8-разрядные процессоры незначительно уступают 16-разрядным.

Положение полей в микропроцессоре КР580ВМ80А переменное, и в зависимости от команды, назначение поля может иметь следующее значение:

Byte 1 – содержит код операции, длину команды, адреса регистров (рис. 13);

Byte 2 – содержит адрес порта ввода/ вывода, 8-разрядный операнд или младшую часть 16-разрядного операнда;

Byte 3 – содержит старшую часть 16-разрядного операнда.

Рис. 13. Формат первого байта команды КР580ВМ80А

Многобайтовая команда должна размещаться в последовательно расположенных ячейках памяти.

Адрес приемника (DDD) и адрес источника (SSS) задают регистр или ссылку на ячейку памяти (признак косвенной адресации), при этом адреса регистров кодируются следующим образом (табл. 1):

Таблица 1

Кодировка адресов регистров

Регистр R	**Код SSS или DDD**	**Регистр R**	**Код SSS или DDD**	**Регистровая пара RP**	**Код RP**
B	000	H	100	BC	00
C	001	L	101	DE	01
D	010	M	110	HL	10
E	011	A	111	SP	11

3.4. Команды пересылок

Команды пересылки выполняют передачу данных из регистра в регистр и размещение данных в памяти. Они не формируют флаги, выполняются за один машинный цикл и кодируются одним байтом по следующим правилам:

1) сначала указывается приемник, затем источник (в архитектуре фирмы DEC – наоборот);

2) если источник или приемник – это ячейка памяти, то ее адрес размещается в регистровой паре HL.

MOV R1, R2 – пересылка (копирование) содержимого регистра R2 в регистр R1. Режим адресации – регистровый. Команда выполняется за один машинный цикл и имеет формат

01’DDD’SSS,

где 01 – код операции пересылки, DDD – номер регистра назначения, а SSS – номер регистра приемника. Например, команда MOV A, B; A ← B, примет следующий двоичный вид: 01’111’000, что соответствует шестнадцатеричному коду 78H.

MOV R, M – пересылка содержимого ячейки памяти, адрес которой находится в регистровой паре HL в регистр R. Режим адресации – косвенный регистровый. Так как требуется обращение к памяти, то команда выполняется за два машинных цикла. Система команд данного микропроцессора не имеет команд для передачи данных между ячейками памяти.

MVI – пересылка (MoVe Immediately) однобайтного непосредственного операнда, содержащегося во втором байте команды в регистр или память. Режим адресации – непосредственный. Команда состоит из двух байт, выполняется за два машинных цикла и имеет формат

00’DDD’110.

Например, команда MVI B,05H; B ← 05H представляет собой два последовательно расположенных в памяти байта – код операции (06H) и (05H). Команда примет следующий двоичный вид:

Byte 1 – 00’000’110; 06H; код операции;

Byte 2 – 00000101; 05H; операнд.

В других процессорах для пересылки непосредственного операнда используется обозначение MOV … , #05H.

LDA – (LoaD Accumulator) загрузка аккумулятора 8-разрядным операндом из ячейки памяти, адрес которой указан во втором и третьем байтах команды. Режим адресации – прямой. Команда выполняется за четыре машинных цикла и имеет код $0011’1010_2$ или 3AH. Например, команда LDA 8203H; A ← M(8203H) примет вид

Byte 1 – 0011’1010; 3AH; код операции;

Byte 2 – 0000’0011; 03H; младшая часть адреса;

Byte 3 – 1000’0010; 82H; старшая часть адреса.

STA – сохранение содержимого аккумулятора в ячейке памяти, адрес которой указан во втором и третьем байте команды. Режим адресации – прямой. Команда выполняется за четыре машинных цикла и имеет код $0011’0010_2$, или 32H.

LXI – загрузка регистровой пары 16-разрядным непосредственным операндом, содержащимся во втором и третьем байтах самой команды. Режим адресации – непосредственный. Команда выполняется за три машинных цикла и имеет формат

00’RR’0001,

где RR – обозначает код регистровой пары (см. табл. 1). Например, команда LXI H,8203H; HL ← 8203H будет иметь вид

Byte 1 – 0011’1010; 21H; код операции;

Byte 2 – 0000’0011; 03H; младшая часть 16-разрядного операнда;

Byte 3 – 1000’0010; 82H; старшая часть 16-разрядного операнда.

3.5. Команды ввода/вывода

Команды ввода/вывода позволяют переместить содержимое аккумулятора в порт вывода и, наоборот, из порта ввести данные в аккумулятор. Напомним, что микропроцессор КР580ВМ80А способен адресовать 256 портов ввода и 256 портов вывода, поэтому адреса портов могут принимать значения от 00H до FFH.

IN – ввод из порта в аккумулятор. Команда состоит из двух байт – кода операции и номера порта. Например, IN 02H; A ← Port 02H. Режим адресации – прямой.

OUT – вывод из аккумулятора в порт. Например, OUT 04H; Port 04H ← A.

3.6. Арифметические команды

Арифметические команды выполняются над содержимым аккумулятора и операндом, указанным в команде. Результат помещается в аккумулятор. КР580ВМ80А имеет следующие команды сложения/вычитания:

ADD – сложение содержимого регистра или ячейки памяти с содержимым аккумулятора;

ADC – сложение с учетом переноса;

ADI – сложение содержимого аккумулятора с непосредственным операндом;

ACI – сложение с непосредственным операндом с учетом флага С.

Например:

ADD B; A ← A + B

ADC C; A ← A + C + flag C

ADI 05H; A ← A + 05H

ACI 05H; A ← A + 05H + flag C

Сложение чисел с разрядностью более одного байта производится программными методами. Например, сложение двух 16-разрядных чисел выполняется в два этапа – сначала складываются младшие байты командой ADD, а затем старшие командой ADC (с учетом переноса).

Рассмотрим операции вычитания. В силу внутренних особенностей АЛУ не обладает возможностями вычитания, оно осуществляет сложение, представляя вычитаемое в форме дополнительного кода и затем складывая его. При арифметических операциях байт может интерпретироваться как:

1) двоичное число без знака в диапазоне от 0 до 255_{10}, или 2^8;

2) число со знаком от -128_{10} до $+127_{10}$ в котором старший (седьмой) бит означает положительность или отрицательность числа;

3) двоично-десятичное число без знака от 00 до 99;

4) двоично-десятичное число со знаком от –50 до 49.

Таким образом, весь диапазон чисел можно представить в виде круга, правую половину которого составляют положительные числа, а начиная с 80H по FFH – отрицательные (рис. 14). При кодировании отрицательных чисел используется *дополнительный код*, который может быть образован двумя способами:

1. $ДК(a) = \overline{a} + 1$, где $\overline{a}$, – это число, представленное в двоичном коде, в котором единицы инвертированы в нули и наоборот. Например, число 05H=0000’0101. Тогда –05H = = 1111’1010 + 1 = 1111’1011 = FBH.

2. $ДК(a) = FF - a + 1$. Число FFH в этой формуле – это максимальное шестнадцатеричное число, которое может быть представлено с помощью восьми бит.

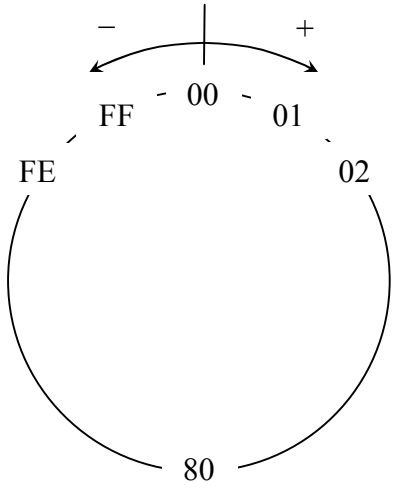


Рис. 14. Диапазон чисел

Каждая команда вычитает содержимое регистра или ячейки памяти из содержимого аккумулятора:

SUB – вычитание содержимого регистра или ячейки памяти из содержимого аккумулятора;

SBB – вычитание содержимого регистра или ячейки памяти с учетом заёма;

SUI – вычитание непосредственного операнда из содержимого аккумулятора;

SBI – вычитание непосредственного операнда из содержимого аккумулятора с заёмом.

Например:

SUB B; A ← A – B

SBB C; A ← A – C – flag C

SUI 05H; A ← A – 05H

SBI 05H; A ← A – 05H – flag C

Отдельную подгруппу арифметических команд составляют команды увеличения/уменьшения на единицу (инкремента/декремента):

INR – инкремент содержимого регистра или ячейки памяти;

DCR – декремент содержимого регистра или ячейки памяти,

например, DCR M; M(HL) ← M(HL) – 1;

INX – инкремент содержимого регистровой пары;

DCX – декремент содержимого регистровой пары.

Особенность этих команд, в отличие от команд сложения/вычитания в том, что команды INR и DCR не формируют флаг C, а команды INX и DCX вообще не формируют флаги.

При операциях ввода/вывода может потребоваться работа в десятичной системе счисления. Отличие двоично-десятичного кода от шестнадцатеричного в том, что каждая десятичная цифра кодируется одной тетрадой. Тогда $25_{10} = 0010'0101_{2/10} = 0001'1001_2 = 19H$. Микропроцессор имеет команду для преобразования числа из шестнадцатеричной системы счисления в двоично-десятичную:

DAA – десятичная коррекция аккумулятора.

Особенность этой команды в том, что она применяется только после команды шестнадцатеричного сложения ADD. Коррекция выполняется над каждой тетрадой числа следующим образом:

1) если после сложения значение младшей тетрады больше 9 или установился флаг AC, то к содержимому аккумулятора прибавляется 6;

2) если после сложения значение старшей тетрады больше 9 или установился флаг C, то к старшей тетраде прибавляется 6.

Например:

$$\begin{array}{rr} \text{ADD} & 38\text{H} \\ & \underline{54\text{H}} \\ & 8\text{CH} \\ + & 6 \\ & \underline{\quad\quad} \\ & 92\text{D} \end{array}$$

При вычитании команда DAA работает некорректно, поэтому необходимо использовать сложение с дополнительным кодом и коррекцию.

3.7. Команды логических операций

Логические команды составляют еще одну группу команд процессора КР580. Они сведены в табл. 2 и содержат команды И (логическое умножение), ИЛИ (логическое сложение), Исключающее ИЛИ, НЕ (инверсия) и сдвига. Здесь именно аккумулятор составляет ядро большинства операций. Как и в рассмотренных ранее командах, режим адресации и здесь влияет на способ и место нахождения других данных в системе.

Таблица 2

Таблица истинности логических операций

Вход	Логическое И	Логическое ИЛИ	Исключающее ИЛИ
0 0	0	0	0
0 1	0	1	1
1 0	0	1	1
1 1	1	1	0

ANA – поразрядное логическое И содержимого регистра и аккумулятора;

ANI – поразрядное логическое И содержимого аккумулятора и непосредственного операнда;

ORA – поразрядное логическое ИЛИ содержимого регистра и аккумулятора;

ORI – поразрядное логическое ИЛИ содержимого регистра и непосредственного операнда;

XRA – поразрядное логическое Исключающее ИЛИ содержимого регистра и аккумулятора (сумма по модулю два);

XRI – поразрядное логическое Исключающее ИЛИ содержимого регистра и непосредственного операнда (сумма по модулю два).

Логические команды позволяют установить в единицу, сбросить в ноль, инвертировать и проверить интересующие нас биты. **Установка бита** в единицу производится формированием *маски*, которая определяет позицию требуемого бита. Маска для установки бита – это байт с нулями во всех битах, кроме искомого. Затем выполняется операция ИЛИ с содержимым аккумулятора и полученной маской. Аналогично, **сброс** бита в ноль осуществляется операцией И над маской, с единицами во всех битах, кроме требуемого, обозначенного нулем. После этих операций следует команда перехода по условию состояния флага Z.

Для примера запишем программы изменения содержимого пятого бита регистра B:

```
;Установка в "1"      ;Сброс в "0"          ;Инверсия бита
MVI  A,00100000B      MVI  A,11011111B      MVI  A,00100000B
ORA  B                ANA  B                XRA  B
MOV  B,A              MOV  B,A              MOV  B,A
```

Другой пример – проанализируем содержимое пятого бита регистра B:

```
MOV  A,B                  b7  b6  b5  b4  b3  b2  b1  b0
ANI  00100000B            0   0   1   0   0   0   0   0
                          ---------------------------------
                          0   0   b5  0   0   0   0   0
```

Условный переход, если флаг Z = 1.

При выполнении логических команд формируются флаги S, Z, P. Флаги C, AC обнуляются. Это дает возможность определить некоторые свойства операнда в аккумуляторе, не изменяя его самого, – знак числа, равно ли оно нулю и четность/нечетность количества единиц (команды ANA A, ORA A). Команда XRA A производит обнуление аккумулятора.

CMA – инвертирование содержимого аккумулятора.

Эта команда может использоваться для вычисления дополнительного кода:

```
CMA       ; инвертирование аккумулятора;
INR  A    ; увеличение аккумулятора на единицу.
```

3.8. Команды сдвига

Микропроцессор КР580ВМ80А имеет четыре команды циклического сдвига. Арифметический и логический сдвиги формируются программно (рис. 15).

RLC – циклический сдвиг содержимого аккумулятора на одну позицию влево. Младший бит и флаг С принимают значение вытесненного бита, т.е. бывшего старшего бита;

RRC – циклический сдвиг содержимого аккумулятора на одну позицию вправо. Старший бит и флаг С принимают значение вытесненного бита, т.е. бывшего младшего бита;

RAL – циклический сдвиг содержимого аккумулятора на одну позицию влево вместе с флагом С. В младшем бите устанавливается содержимое флага С;

RAR – циклический сдвиг содержимого аккумулятора на одну позицию вправо вместе с флагом С. В старшем бите устанавливается содержимое флага С.

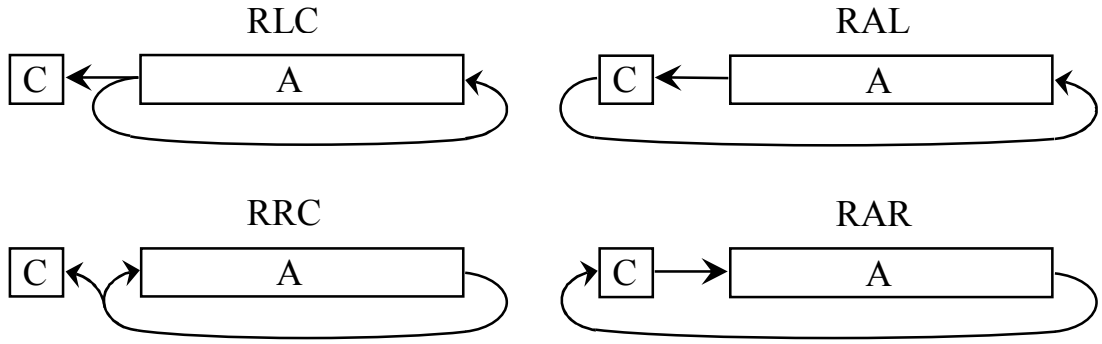


Рис. 15. Выполнение команд циклического сдвига

Эти команды позволяют реализовать арифметический сдвиг, т.е. умножение и деление на два (рис. 16).

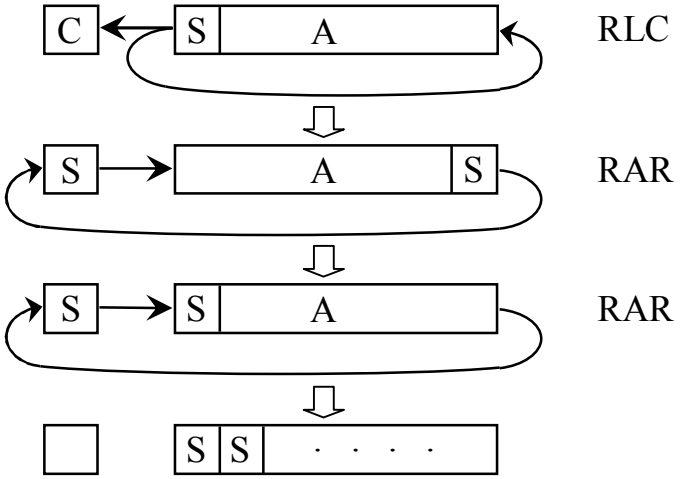


Рис. 16. Арифметический сдвиг вправо, реализующий деление на 2

Особенность арифметического сдвига состоит в неизменности старшего (знакового) разряда. Для этого необходимо выполнить циклический сдвиг влево, минуя флаг переноса, и двукратный циклический сдвиг вправо через флаг

переноса. После второго сдвига в аккумуляторе восстанавливается исходное число, а во флаге переноса помещается его знак. Третий сдвиг формирует число, соответствующее арифметическому сдвигу вправо.

Замечание. При арифметическом сдвиге вправо положительных чисел предельное значение составляет 00H; при сдвиге влево отрицательных чисел предельное значение составляет FFH = –1.

3.9. Команды сравнения

Команды сравнения используются для формирования флагов перед условным переходом. При этом выполняется вычитание, но его *результат не запоминается в аккумуляторе*.

CMP – сравнение содержимого регистра или ячейки памяти и аккумулятора;

CPI – сравнение байта с содержимым аккумулятора,

например:

CMP B; flags (A – B);

CPI 05H; flags (A – 05H).

3.10. Команды передачи управления

Эта группа содержит команды перехода по какому-либо условию, изменяя последовательный ход программы. Имеются команды двух типов: условного и безусловного переходов. Безусловный переход просто выполняют операцию, определенную счетчиком команд; условные – проверяют состояние одного из флагов процессора для определения необходимости в ветвлении.

JMP – безусловный переход. Управление передается команде по адресу, указанному во втором и третьем байтах команды;

JZ – переход, если флаг Z = 1;

JNZ – переход, если флаг Z = 0;

JC – переход, если флаг C = 1;

JNC – переход, если флаг C = 0;

JP – переход, если флаг S = 0;

JM – переход, если флаг S = 1;

JPE – переход по четности, если флаг P = 1;

JPO – переход по нечетности, если флаг P = 0,

например:

JMP 8200H; (PC) ← (байт 3) (байт 2)

Примечание. В других процессорах имеются переходы с коротким адресом, а также условные переходы по комбинациям флагов (с учетом переполнения).

3.11. Команды работы с подпрограммами

Как и команды передачи управления, эта группа команд позволяет осуществить безусловный переход либо вызов подпрограммы по условию. Иногда их называют командами переходов с возвратом.

CALL – безусловный вызов подпрограммы.

По команде CALL текущее значение счетчика команд записываются в стек, а в счетчик команд загружается новый адрес (из второго и третьего байт команды).

RET – возврат из подпрограммы. Содержимое вершины стека переписывается в счетчик команд.

Процессор КР580ВМ80А имеет также команды вызова подпрограмм и возврата из них по некоторому условию (состоянию флагов), которые аналогичны командам перехода. Например, команда CNZ осуществляет вызов подпрограммы при условии $Z = 0$, а команда RC производит возврат из подпрограммы при условии $C = 1$.

3.12.Специальные команды

DI – запрет прерываний;

EI – разрешение прерываний;

HLT – останов (до появления прерывания);

NOP – пустая операция.

Примечание. Последовательность команд DI … HLT приводит к зависанию, выход из которого возможен по сигналу «Сброс».

4. Состав микропроцессорного комплекта КР580

Микропроцессорный комплект серии КР580 предназначен для создания широкого класса средств вычислительной техники и обработки информации. Он включает в себя микропроцессор и ряд интерфейсных схем. Интерфейс – это совокупность устройств и правил связи двух объектов. В данном курсе лекций интерфейс – это совокупность программно-аппаратных средств, обеспечивающих сопряжение микропроцессора, памяти и устройств ввода/вывода. Существует ряд стандартов на интерфейсы, которые определяют временные диаграммы, тип разъема, уровни сигнала и т.д.

В состав комплекта КР580 входят следующие БИС:

- БИС общего назначения:

  КР580ВМ80А – центральное процессорное устройство;

  КР580ГФ24 – генератор тактовых импульсов;

  КР580ВК28 – системный контроллер и шинный формирователь;

  КР580ИР82, ИР83 – стробируемые регистры;

  КР580ВА86, ВА87 – шинные формирователи.

- Универсальные интерфейсные БИС:

  КР580ВВ51 – последовательный программируемый интерфейс;

  КР580ВВ55 – параллельный программируемый интерфейс;

  КР580ВИ53 – программируемый таймер;

  КР580ВВ57 – программируемый контроллер ПДП;

  КР580ВН59 – программируемый контроллер прерываний.

- БИС контроллеров устройств:

  КР580ВГ75 – контроллер дисплея на ЭЛТ;

  КР580ВВ79 – контроллер клавиатуры и матричного дисплея;

  КР580ВК91 – интерфейс КОП (приборный интерфейс);

  КР580ВТ42 – контроллер динамического ОЗУ;

  КР580ВР43 – расширитель ввода/вывода.

В настоящее время микропроцессор КР580ВМ80А уже практически не применяется, однако интерфейсные БИС семейства КР580 широко используются в микроконтроллерах и системах управления.

4.1. Генератор тактовых импульсов КР580ГФ24

Для синхронизации компонентов микропроцессорной системы используется генератор тактовых импульсов (ГТИ) КР580ГФ24 (рис. 17). Он формирует тактовые импульсы Ф1, Ф2 частотой до 2,5 МГц, амплитудой 12 В и тактовые импульсы Ф2TTL амплитудой 5В для TTL-схем. Генератор также осуществляет привязку фронтов сигналов «Готовность» и «Сброс» и формирует сигнал «Строб

состояния». Для работы ГТИ необходим кварцевый резонатор с частотой колебаний в 9 раз большей, чем частота выходных тактовых импульсов ГТИ.

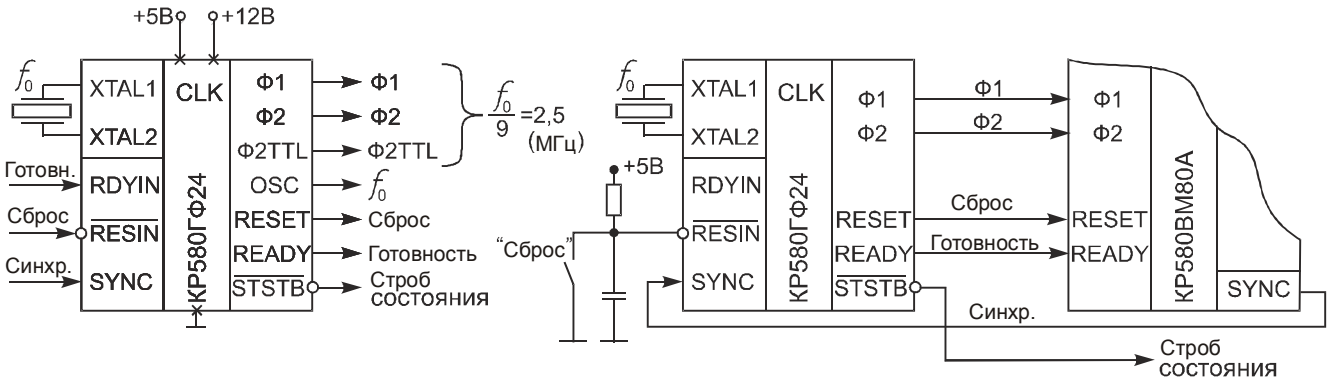


Рис. 17. Генератор тактовых импульсов КР580ГФ24

4.2. Системный контроллер и шинный формирователь КР580ВК28

В системах управления, построенных на МП КР580ВМ80 без системного контроллера, обращения к памяти и портам ввода/вывода не различимы. Это приводит к тому, что все порты адресуются так же, как и ячейки памяти, и вместо команд IN/OUT необходимо использовать MOV M,A, MOV A,M, так как данные на шине сопровождаются одним и тем же сигналом процессора WR/RD. Для исключения этой неоднозначности в каждом машинном цикле слово состояния процессора содержит информацию о направлении потока данных. Микропроцессорный комплект КР580 имеет в своем составе системный контроллер КР580ВК28, формирующий управляющие сигналы для памяти, портов ввода/вывода и подтверждения прерывания. При этом на выводе STSTB ГФ24 (рис. 18) в момент действия сигнала Ф1 формируется импульс, сопровождающий байт состояния микропроцессора. По этому сигналу системный контроллер дешифрирует слово-состояния процессора и генерирует набор управляющих сигналов.

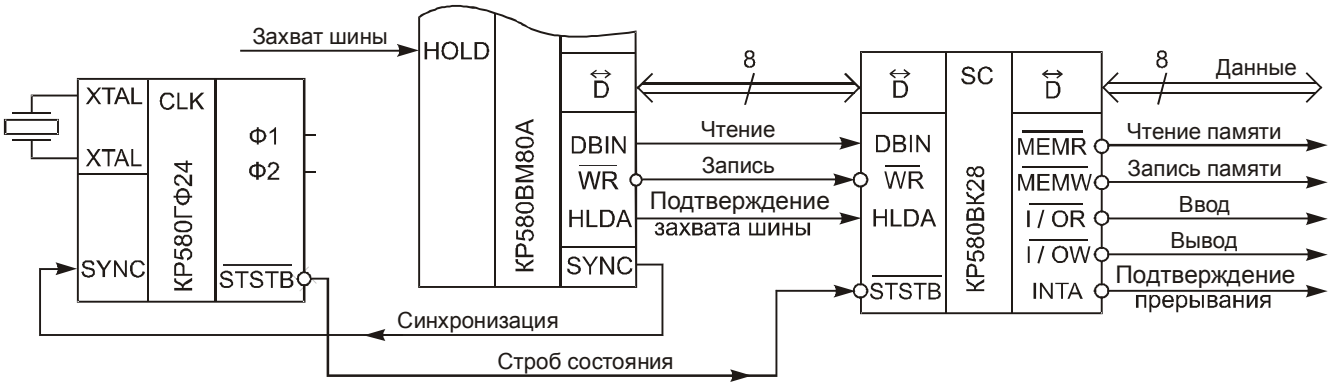


Рис. 18. Системный контроллер и шинный формирователь КР580ВК28

Другое назначение этой микросхемы связано с тем, что выходы процессора могут быть нагружены не более чем на один TTL-вход, поэтому системный контроллер выполняет функцию шинного формирователя, усиливая сигналы на шине данных.

4.3. Буферные регистры КР580ИР82, КР580ИР83

Для временного хранения, реализации схем фиксации, буферизации и мультиплексирования используются буферные регистры КР580ИР82 и КР580ИР83. На выходах ИР83 генерируются инвертированные данные (рис. 19), кроме того, выходы обоих регистров имеют усилители, что позволяет использовать их для повышения нагрузочной способности шин.

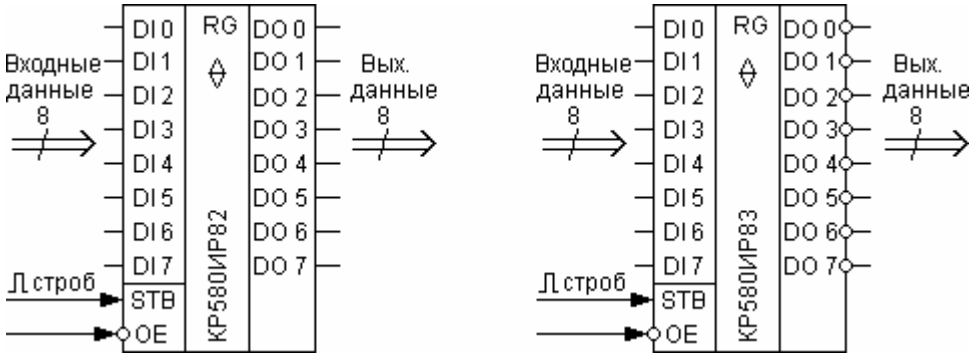


Рис. 19. Буферные регистры ИР82 и ИР83

Запись входных данных в регистры производится при переходе сигнала *STB* с логической единицы в нуль. Логическая единица на входе *OE* переключает регистры в высокоимпедансное состояние, при котором его шины фактически отключаются от процессора.

4.4. Шинные формирователи КР580ВА86 и КР580ВА87

Известно, что устойчивая работа процессора КР580ВМ80 гарантируется, если его выходы нагружены не более чем на один TTL-вход. Однако системы управления состоят из множества различных компонентов, подключаемых к процессору. Для усиления нагрузочной способности его шин используются шинные формирователи КР580ВА86 и КР580ВА87 (рис. 20). Они представляют собой 8-разрядные двунаправленные приемопередатчики с третьим состоянием, обеспечивающие ток нагрузки до 32 мА. Микросхема ВА86 отличается от ВА87 инвертированными выходами и имеет двунаправленный канал А, подключаемый к шине процессора, и двунаправленный канал В, подключаемый к различным контроллерам, памяти или портам. При логической единице на входе *T* информация передается со стороны *A* в сторону *B*, а при логическом нуле – наоборот. Вход *OE* служит для перевода формирователей в высокоимпедансное состояние, при котором шины фактически отключаются от процессора.

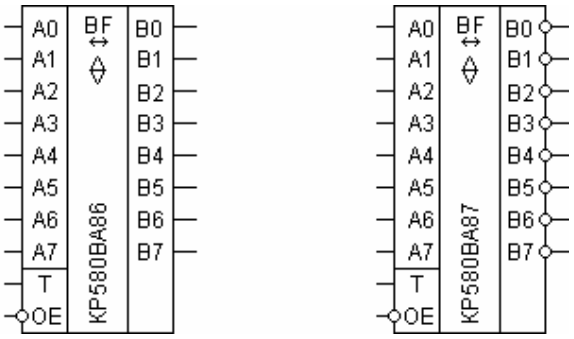


Рис. 20. Шинные формирователи

5. Память микропроцессорных систем

5.1. Классификация запоминающих устройств

Памятью называется совокупность технических средств, предназначенных для записи, хранения и считывания информации в виде цифрового кода. Отдельные элементы памяти получили название запоминающих устройств (ЗУ). Основная память микропроцессорной системы состоит из ЗУ двух видов: – оперативного – ОЗУ (RAM, Random Access Memory) и постоянного – ПЗУ (ROM, Read Only Memory) (рис. 21).

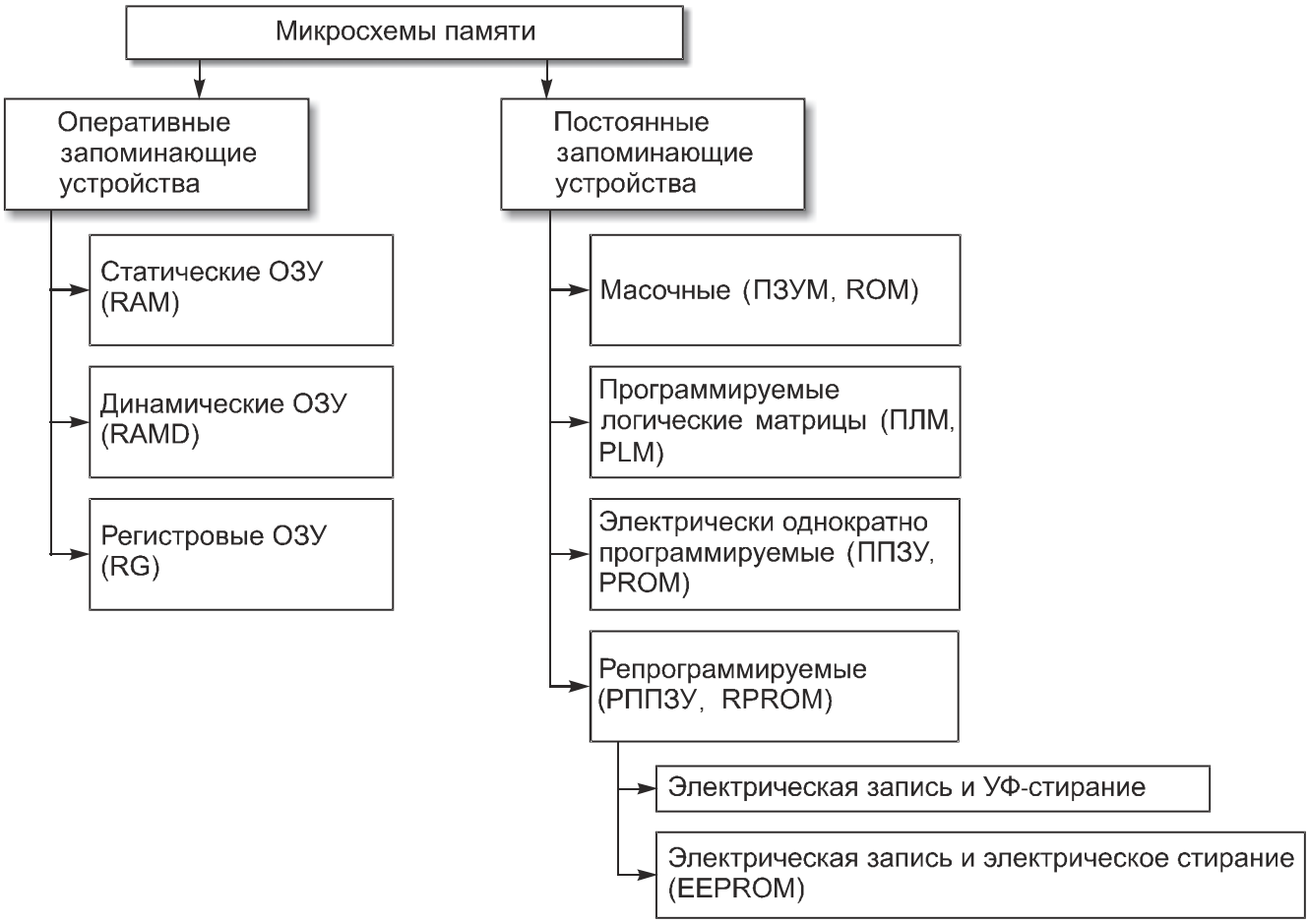


Рис. 21. Классификация микросхем памяти

ОЗУ предназначено для хранения переменной информации, оно допускает изменение своего содержимого в ходе выполнения процессором вычислительных операций с данными. Это значит, что процессор может выбрать из ОЗУ код команды и данные (режим считывания) и после обработки поместить в ОЗУ полученный результат (режим записи). Причем возможно размещение в ОЗУ новых данных на месте прежних, которые в этом случае перестают существовать. При этом различают статические и динамические ОЗУ.

В микросхемах *статических ОЗУ* информация хранится в виде устойчивого состояния триггера, который способен при наличии напряжения питания сохранять свое состояние неограниченное время. Достоинством таких ОЗУ является максимальное быстродействие, а недостатком – высокая стоимость и значительное энергопотребление.

В микросхемах *динамических ОЗУ* элементы памяти выполнены на основе конденсаторов, сформированных внутри полупроводникового кристалла. Такие элементы памяти не могут долгое время сохранять свое состояние, определяемое наличием или отсутствием электрического заряда, и поэтому нуждаются в периодическом обновлении (*регенерации*). Микросхемы динамических ОЗУ отличаются от статических гораздо большей информационной емкостью, что обусловлено меньшим числом компонентов в одном элементе памяти и, следовательно, более плотным их размещением в полупроводниковом кристалле. Однако динамические ОЗУ сложнее в применении, поскольку нуждаются в организации принудительной регенерации и в усложнении устройств управления. Динамическая память имеет среднее быстродействие и невысокую стоимость. Таким образом, ОЗУ может работать в режимах записи, считывания и хранения информации.

ПЗУ содержит информацию, которая не изменяется в ходе выполнения процессором программы и должна храниться при выключенном источнике питания. Такую информацию составляют стандартные подпрограммы, табличные данные, коды физических констант, постоянных коэффициентов и т.п. Эта информация заносится в ПЗУ предварительно, например, путем пережигания легкоплавких металлических перемычек в структуре ПЗУ, и в ходе работы, процессора может только считываться.

Существует разновидность ПЗУ, допускающая неоднократное (сотни тысяч циклов) перепрограммирование (репрограммирование). Элементом памяти в репрограммируемых ПЗУ (РПЗУ, PROM) является МДП-транзистор, обладающий свойством переходить в состояние проводимости под воздействием импульса программирующего напряжения и сохранять это состояние длительное время. Данный эффект обусловлен накоплением электрического заряда в подзатворном диэлектрике. Для стирания информации перед новым циклом программирования необходимо вытеснить накопленный под затвором заряд. В зависимости от способа выполнения этой операции микросхемы РПЗУ разделяют на два вида: со стиранием ультрафиолетовым светом (УФ РПЗУ) и со стиранием электрическим сигналом (ЭС РПЗУ, EEPROM, или *Flash-память*). Флэш-технология позволяет оснастить системную память уникальными свойствами. Подобно ОЗУ, флэш-память модифицируется электрически внутрисистемно, но, подобно ПЗУ, флэш энергонезависима и хранит данные даже после отключения питания. Однако в отличие от ОЗУ флэш нельзя переписывать побайтно: ее нужно стереть перед записью новых данных.

Микросхемы флэш-памяти в последнее время получили большое распространение ввиду высоких потребительских качеств – простоты

программирования, высокой скорости чтения и значительной емкости. Параметрические блоки флэш-памяти используются для хранения телефонных номеров, учета времени использования и идентификатора пользователя (SIM-карта) в сотовых телефонах. Производители автомобилей используют флэш-память в системах управления двигателями для хранения кодов ошибок и параметров оптимальных режимов работы. В каждом из подобных примеров изготовители экономят на расходах, связанных с необходимостью содержания складского запаса «прошитых» разными программами ПЗУ, используя флэш-память не только для хранения прикладных программ, но и параметров.

Следует отметить, что существует две разновидности флэш-памяти. Первая используется для хранения программ и имеет емкость порядка 1 Мбайт; вторая, – NAND EEPROM используется, в основном, в качестве мобильного носителя данных, имеет последовательный доступ к данным и емкость до 4 Гбайт. Таким образом, ПЗУ работает в режимах хранения и считывания.

Запоминающее устройство, реализующее функции основной памяти, размещают рядом с процессором в одном блоке, и такое ЗУ в этом смысле является внутрисистемным. Быстродействие внутреннего ЗУ должно быть соизмеримо с быстродействием процессора. Однако практически это требование не всегда удается выполнить: по временны**м** параметрам ОЗУ и ПЗУ отстают от процессора. Поэтому внутри ЭВМ обычно размещают еще и вспомогательную (буферную) память на быстродействующих регистрах, которая используется в качестве *сверхоперативного ЗУ* (СОЗУ или cash) с небольшой информационной емкостью для кратковременного хранения текущих команд, адресов и данных.

Важнейшими характеристиками ЗУ являются:

- емкость, удельная емкость;
- быстродействие;
- энергопотребление;
- способность сохранять информацию при отключении питания.

Информационная емкость определяет число единиц информации в битах или байтах, которое БИС памяти может хранить одновременно. Она выражается через число ячеек N с указанием разрядности n в виде $M=N*n$. *Удельная емкость* – отношение информационной емкости к ее физическому объему. *Быстродействие*, как правило, характеризуется двумя параметрами:

1. Время выборки ($t_в$) – представляет интервал времени между передачей сигнала «выборка кристалла» (CS) при считывании информации и появлением информации на шине данных,

2. Время цикла записи ($t_{цз}$), которое определяется минимально возможным временем с момента подачи сигнала CS при записи и повторном обращении к памяти.

В качестве характеристики быстродействия памяти выбирается максимальное из $t_в$ и $t_{цз}$. В табл. 3 приведены типичные скоростные параметры для разных классов памяти.

Таблица 3

Временные параметры основных типов памяти, мкс

Технология изготовления	**Статические ОЗУ SRAM**	**ППЗУ PROM**	**ПЗУ ROM**	**Динамические ОЗУ DRAM**
Биполярная	30–100	50–150	50–150	-
МОП	200–500	300	350–1800	500

5.2. Память как функциональный узел

Рассмотрим микросхему памяти как «черный ящик», обратив основное внимание на назначение ее выводов, внешние и внутренние характеристики. На рис. 22 приведены графические изображения ОЗУ и ПЗУ.

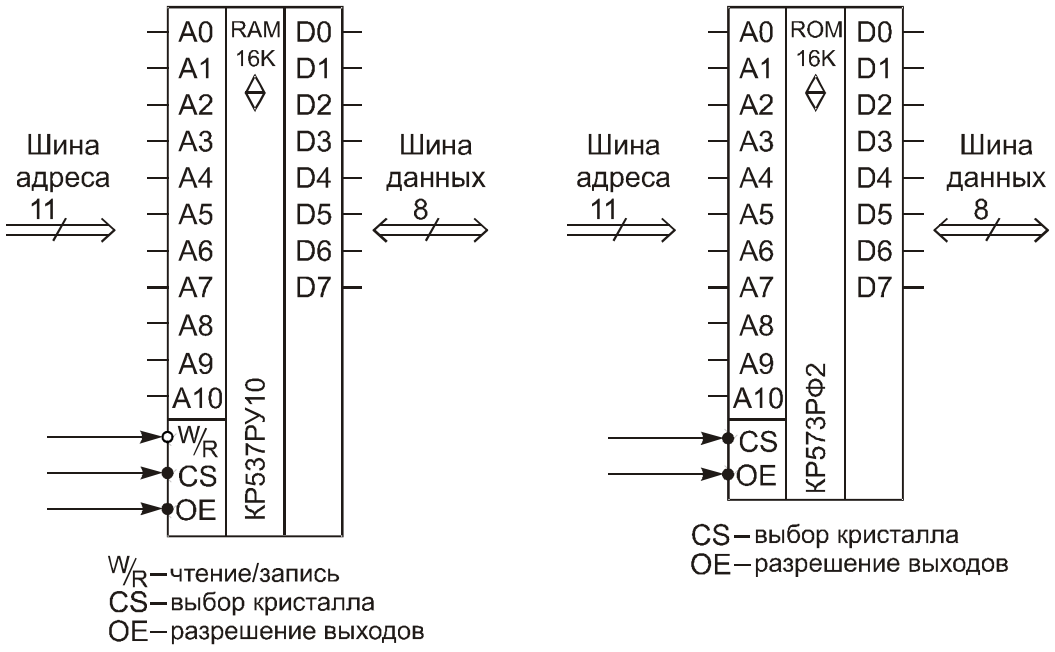


Рис. 22. Условные обозначения микросхем ОЗУ и ПЗУ

Сигналы и соответствующие выводы можно подразделить на *адресные*, *управляющие* и *информационные*. Число адресных входов A0 – A11 позволяет определить информационную емкость микросхемы: $2^{11} = 2048$ адресуемых ячеек памяти. Наличие восьми информационных выводов указывает на восьмиразрядную организацию каждой ячейки. Поэтому общий объем памяти составляет 2048×8 бит $= 2$ Кбайт. Для управления режимом работы предусмотрены три сигнала: W/R – чтение/запись (Write/Read), CS – выбор микросхемы (Chip Select) и OE – разрешение выходов (Output Enable). Для обращения к микросхеме для записи или считывания одного байта информации

необходимо подать сигнал CS с нулевым уровнем (разрешающий обращение) и сигнал W/R с соответствующим режиму уровнем: при записи – 1, при считывании – 0. Для упрощения дешифрации микросхемы памяти могут иметь несколько входов CS. Входы-выходы D совмещены, поэтому они обладают свойством двунаправленной проводимости. Отметим, что все операции чтения записи возможны только при низком активном уровне на входе OE. В противном случае шина данных переключается в высокоомное состояние, что равносильно ее отключению.

5.3. Многомодульная организация памяти

Механизм взаимодействия процессора с памятью основан на классической архитектуре вычислительной системы, состоящей из трех основных компонентов: процессора, памяти и устройств ввода/вывода, объединенных шинами данных, адреса и управления (рис. 23).

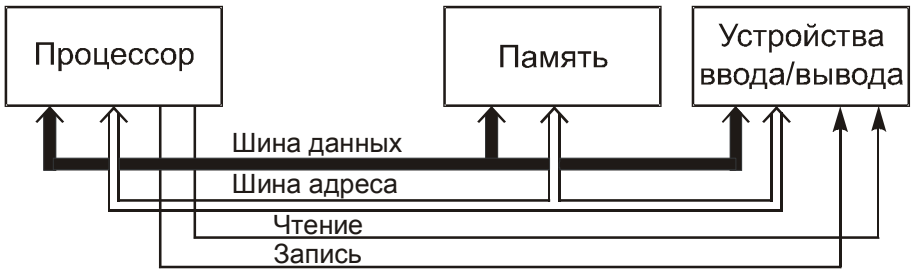


Рис. 23. Взаимодействие процессора и памяти

В простейшем случае в процессе работы программы МП выдает на адресную шину адрес требуемой ячейки памяти, сопровождаемый сигналом чтение/запись шины управления, а память извлекает или сохраняет информацию, находящуюся на шине данных. При обмене информацией с памятью ведущим является микропроцессор либо контроллер прямого доступа к памяти. При этом возникает проблема согласования быстродействия МП и памяти. Обычно процессор более быстродействующий, чем остальные компоненты, поэтому, чтобы согласовать временные параметры всех подсистем, применяются два способа:

1) синхронный, при котором между выдачей адреса и передачей данных производится фиксированная задержка (целое число тактов);

2) асинхронный, при котором после приема адреса и его дешифрации память отвечает процессору сигналом «Готовность».

На рис. 24 представлено построение и взаимодействие микропроцессора и памяти, состоящей из ОЗУ и ПЗУ одинаковой емкости и одинаковой организации 16 К×8. Известно, что процессор КР580ВМ80 с помощью шестнадцати адресных линий может адресовать 2^{16}= 64 Кбайт памяти. Организуем ее таким образом, чтобы ее первая половина (32 Кбайт) была отведена под ОЗУ, а вторая (32 Кбайт) – под ПЗУ. При этом все адресное пространство оказывается разделенным на четыре банка (две микросхемы ОЗУ и две микросхемы ПЗУ по 16 Кбайт каждая).

При обмене информацией процессор выставляет на адресной шине 16-разрядный адрес ячейки памяти, который сопровождается сигналом «чтение/запись». Четырнадцать младших разрядов адреса (A0-A13) непосредственно подключены к соответствующим входам микросхем памяти, а две старшие адресные линии A14 и A15 определяют номер банка. Микросхема-дешифратор посредством сигнала CS (выбор кристалла) позволяет выбрать положение микросхемы ЗУ в адресном пространстве. Для данного случая это адреса 0000h-7FFFh для ОЗУ и 8000h-FFFFh для ПЗУ. Напомним, что запись информации в ППЗУ возможна только вне микропроцессорной системы в специальном программаторе.

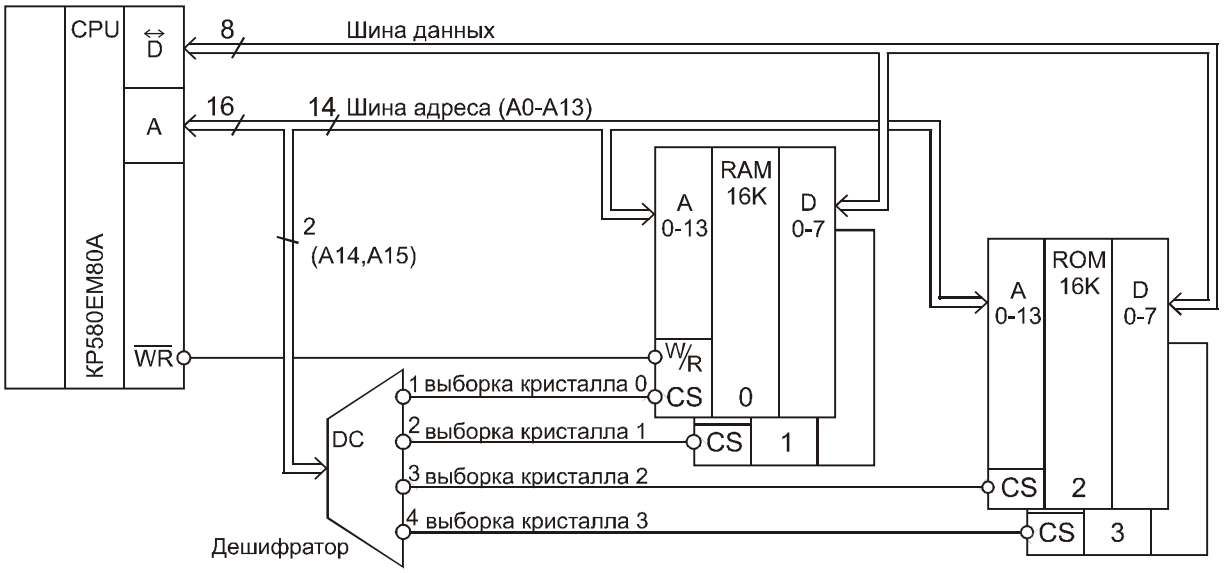


Рис. 24. Многомодульная организация памяти

В некоторых микропроцессорах (рис. 25) шина адреса и шина данных совмещается (мультиплексируется). В этом случае шина сначала используется для передачи адреса, а затем по ней передаются данные. При этом адрес запоминается во внешнем регистре, который стробируется специальным сигналом разрешения захвата адреса ALE.

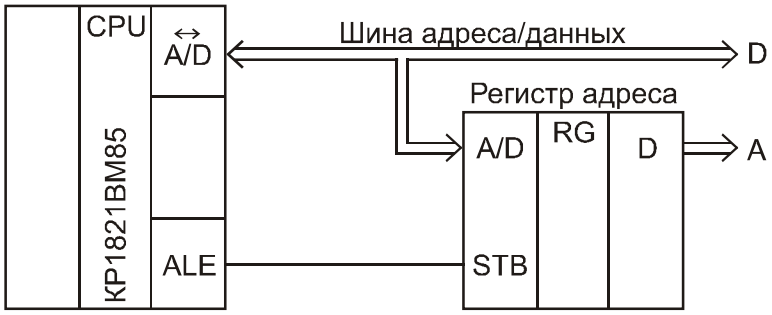


Рис. 25. Мультиплексирование шин

Аналогичный прием (совмещение шин адреса и данных) используется в большинстве современных микроконтроллеров.

5.4. Организация стековой памяти

Стеком называют безадресный способ организации памяти, доступ к которой организован по принципу: «*последним пришел, первым ушел*» (Last Input First Output – LIFO). Использование принципа доступа к памяти на основе механизма LIFO началось с больших ЭВМ. В малых ЭВМ она стала широко использоваться в связи с удобствами реализации процедур вызова подпрограмм и при обработке прерываний. В микропроцессорных системах стековый принцип доступа к памяти стал широко использоваться из-за короткой длины машинного слова. Стек позволяет сохранять адреса возврата и флаги при обработке прерываний и вызове подпрограмм, а также передавать параметры в подпрограммы.

Принцип работы стековой памяти состоит в следующем (рис. 26). Когда слово A помещается в стек, оно располагается в первой свободной ячейке памяти. Следующее записываемое слово перемещает предыдущее на одну ячейку вверх и занимает его место и т.д. Запись 8-го кода, после H, приводит к переполнению стека и потере кода A. Считывание слов из стека осуществляется в обратном порядке, начиная с кода H, который был записан последним. Заметим, что выборка, например кода E, невозможна до выборки кода F, что определяется механизмом обращения при записи и чтении типа LIFO. Для фиксации переполнения стека желательно формировать признак переполнения.

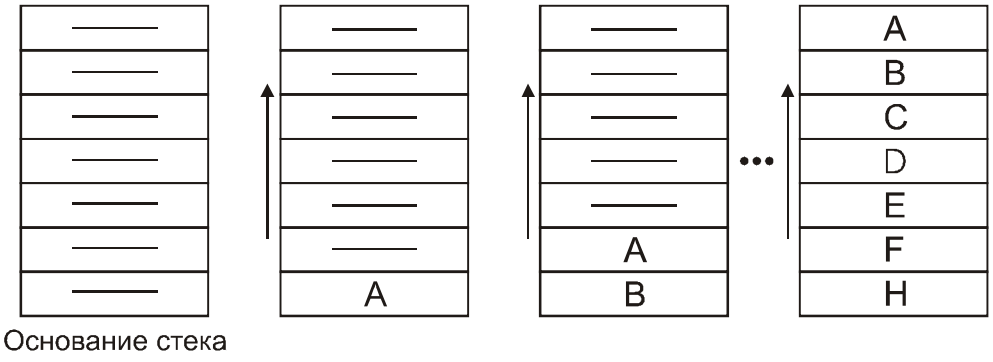


Рис. 26. Стек LIFO

Стек можно организовать двумя способами: на основе регистра сдвига и на основе ОЗУ и указателя стека (используется в процессоре КР580ВМ80) (рис. 27).

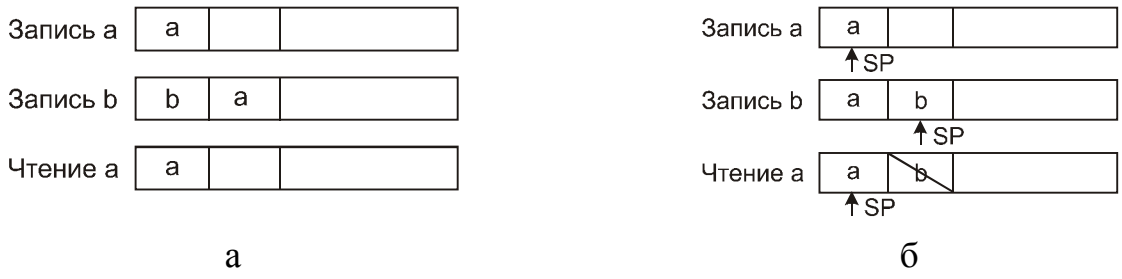


Рис. 27. Способы организации стека:
а – на основе регистра сдвига; б – на основе указателя стека и ОЗУ

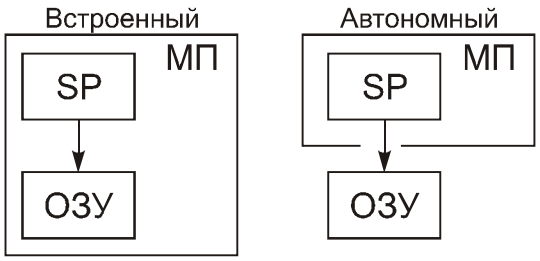


Рис. 28. Типы стека

Существует два типа стека (рис. 28):

• встроенный (указатель стека находится на кристалле МП);

• автономный (указатель стека находится на кристалле МП, а ОЗУ – на внешней микросхеме).

Встроенный стек имеет небольшой объем (несколько десятков килобайт), но обеспечивает максимальное быстродействие, так как не требует обращения к внешней памяти.

Чтение и запись в стек производится по следующему алгоритму:

PUSH a – запись в стек	**POP a** – чтение из стека
`SP:=SP+1`	`a:=M(SP)`
`M(SP):=a`	`SP:=SP-1`

В различных МП используются и другие алгоритмы записи/чтения. Так, стек может расти вниз, а не вверх, и его указатель может изменяться как до записи, так и после. Начальное значение указателя стека определяется программистом или операционной системой, а в некоторых МП, например MCS51, устанавливается при сбросе.

В сложных МП при вызове подпрограмм и прерываниях флаги сохраняются в стеке аппаратно. Если указатель попал в область, отведенную для векторов прерываний, команд или данных, то произойдет автоматическое переполнение стека. В простых МП за сохранением регистра флагов должен следить сам программист.

6. Организация ввода/вывода в микропроцессорной системе

В ЭВМ применяются три режима ввода/вывода:

- программно-управляемый ввод/вывод;
- ввод/вывод в режиме прерываний;
- ввод/вывод в режиме прямого доступа к памяти.

Первый из них характеризуется тем, что инициирование и управление осуществляется программой, выполняемой процессором, а внешние устройства играют сравнительно пассивную роль и только сигнализируют о своем состоянии, в частности, о готовности к операциям ввода/вывода. Во втором режиме ввод/вывод инициируется не процессором, а внешним устройством, генерирующим специальный сигнал прерывания. Реагируя на этот сигнал готовности устройства к передаче данных, процессор передает управление подпрограмме обслуживания устройства, вызвавшего прерывание. Действия, выполняемые этой подпрограммой, определяются пользователем, а непосредственными операциями ввода/вывода управляет процессор. Наконец, в режиме прямого доступа к памяти, который используется, когда пропускной способности процессора недостаточно, процессор приостанавливается, он отключается от системной шины и не участвует в передаче данных между основной памятью и быстродействующим ВУ.

6.1. Программно-управляемый ввод/вывод

Данный режим характеризуется тем, что все действия по вводу/выводу реализуются командами прикладной программы через порт (рис. 29). Наиболее простыми эти действия оказываются для «всегда готовых» внешних устройств, например индикатора на светодиодах. При необходимости ввода/вывода в соответствующем месте программы используются команды IN или OUT. Такая передача данных называется синхронным вводом/выводом (рис. 30, а).

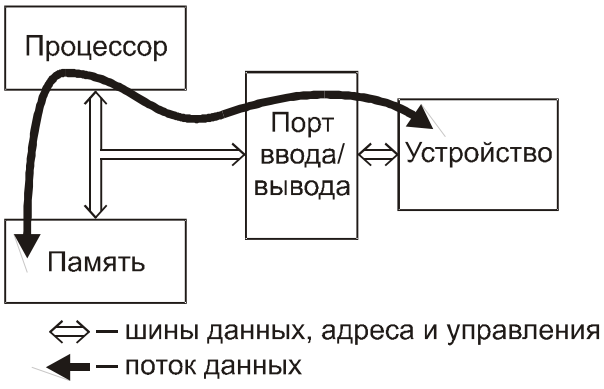


Рис. 29. Аппаратная реализация программно-управляемого ввода/вывода

Однако для большинства ВУ до выполнения операций ввода/вывода необходимо убедиться в их готовности к обмену (рис. 30, б). Такой режим ввода/вывода называют асинхронным. Общее состояние устройства

характеризуется флагом статуса READY, называемым также флагом готовности/занятости (READY/BUSY). Процессор проверяет готовность устройства с помощью команды `IN port READY`, выставляет флаги командой `ANA` и осуществляет проверку флагов командой `JP`. Если флаг READY установлен, то инициируются собственно ввод или вывод данных из порта DATA. Когда же флаг сброшен, процессор выполняет цикл из 2–3 команд с повторной проверкой флага READY до тех пор, пока устройство не будет готово к операциям ВВ. Данный цикл называется циклом ожидания готовности ВУ.

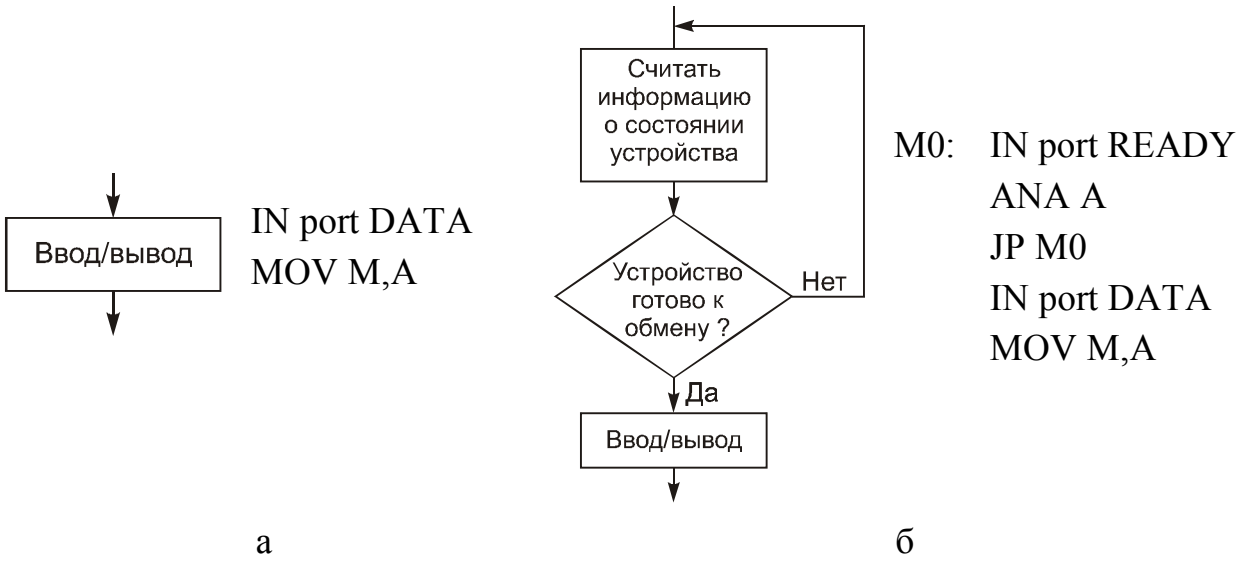


Рис. 30. Выполнение операций ввода/вывода: а – синхронный; б – асинхронный

Основной недостаток программного ввода/вывода связан с непроизводительными потерями времени процессора в циклах ожидания. К достоинствам следует отнести простоту его реализации, не требующей дополнительных аппаратных средств.

6.2. Ввод/вывод в режиме прерываний

При организации в ЭВМ системы прерываний непроизводительные потери времени процессора в циклах ожидания готовности резко сокращаются. Режим с прерыванием программы позволяет организовать обмен данными с внешними устройствами в произвольные моменты времени, не зависящие от программы (определяемые внешними устройствами).

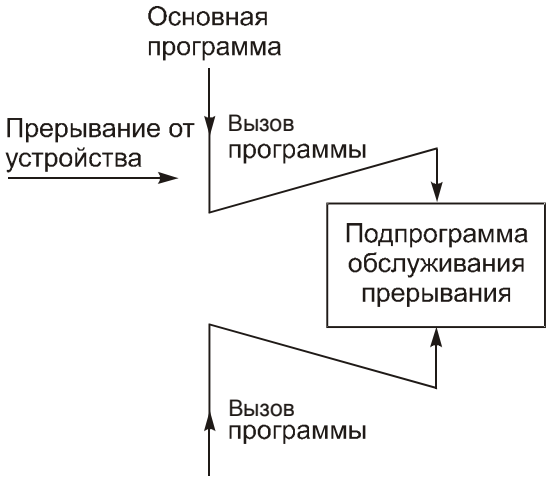
Главное отличие ввода/вывода в режиме с прерыванием от программного ввода/вывода состоит в том, что:

1) инициатором является внешний сигнал;

2) заранее не известно, в какой момент будет прервана фоновая программа.

При этом каждое периферийное устройство может посылать в процессор сигнал INT (INTerrupt) запроса прерывания, когда оно готово к операциям ввода/вывода. По существу, этот сигнал представляет собой выходной сигнал

триггера, фиксирующего флаг готовности READY. Сигнал INT появляется в произвольные моменты времени, асинхронно по отношению к действиям процессора, и управлять его появлением программа не может. Следовательно, заранее не известно, в какой точке программы и какие периферийные устройства инициируют прерывания, поэтому непосредственно в программе команды ввода/вывода использовать нельзя. Остается одно: реагируя на сигнал INT, процессор должен прервать, т.е. временно приостановить текущую программу, идентифицировать прерывающее устройство, перейти к подпрограмме обслуживания прерываний работы этого устройства, а после ее завершения возобновить выполнение прерванной программы. Подпрограмме обслуживания потребуются внутренние регистры процессора: аккумулятор, программный счетчик, некоторые РОН, и их текущее содержимое будет модифицировано. Но прерванная программа должна возобновиться так, как будто прерывания вообще не было (рис. 31).




```
PUSH PSW;  stack ← (A,F)
PUSH B
…; (операции ввода/вывода)

POP B
POP PSW
RET
```

Рис. 31. Реакция на прерывание

Факт обслуживания прерывания влияет на прерванную программу только увеличением времени ее выполнения. Следовательно, содержимое всех регистров, необходимых подпрограмме обслуживания прерывания, следует временно запоминать. В качестве такого временного «хранилища» удобно использовать стек (однако чем больше информации запоминается, тем больше время реакции на прерывание). Предпочтительными с точки зрения повышения производительности микроЭВМ являются уменьшение числа команд, обеспечивающих сохранение информации о прерванной программе, и реализация этих функций аппаратными средствами.

В микропроцессорном комплекте КР580 механизм управления прерываниями реализован в специальной БИС КР580ВН59 (рис. 32), однако общая последовательность реакции на сигнал прерывания примерно одинакова для всех микропроцессоров и содержит следующие шаги:

1. Периферийное устройство генерирует сигнал прерывания, который подается на вход INT процессора; на этой линии по схеме ИЛИ объединяются запросы всех устройств, работающих в режиме прерываний.

2. Процессор завершает текущую команду и, если прерывания разрешены (не замаскированы), формирует сигнал подтверждения прерывания INTA (INT Acknowledgement).

3. Запоминается содержимое счетчика команд PC и некоторых других внутренних регистров в стеке (в КР580 автоматически сохраняется только адрес возврата).

4. Процессор идентифицирует прерывающее устройство для перехода к соответствующей подпрограмме обслуживания.

5. Выполняется короткая (30–50 байт) подпрограмма обслуживания прерывания, в которой запрограммированы действия по передаче данных, модификации указателей, проверке окончания операций ввода/вывода.

6. Восстанавливается состояние прерванной программы, для чего сохраненное содержимое регистров извлекается из стека.

7. Возобновляется выполнение прерванной программы; это действие инициируется командой возврата из прерывания RET, являющейся последней командой подпрограммы обслуживания прерывания.

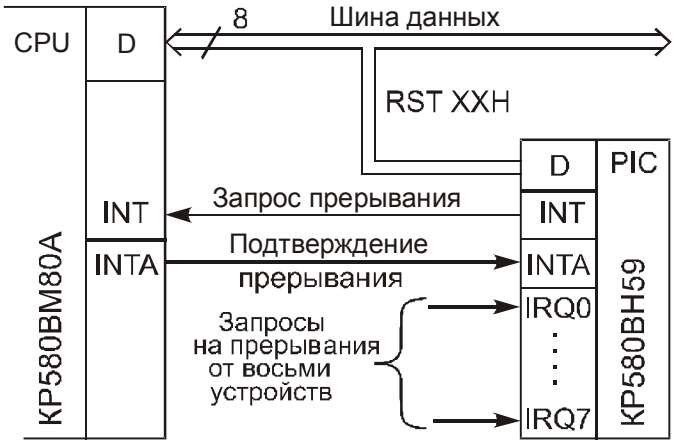


Рис. 32. Организация ввода/вывода в режиме прерываний

Следует отметить, что при выполнении этой последовательности в некоторых процессорах содержимое флагов и аккумулятора сохраняется в стеке автоматически, при этом для возврата используется специальная команда RTI (возврат из прерывания). Также существуют процессоры с несколькими наборами (банками) регистров. При обработке прерываний банки переключаются с основного на дополнительный, что позволяет избежать сохранения содержимого регистров в стеке.

В микроЭВМ обычно используется одноуровневая система прерываний, т.е. сигналы «Запрос прерывания» от всех ВУ поступают на один вход процессора. Поэтому возникает проблема идентификации ВУ, запросившего обслуживание, и реализации заданной очередности (приоритета) обслуживания ВУ при одновременном поступлении нескольких сигналов прерывания.

В контроллере КР580ВН59 реализован следующий механизм идентификации устройств. На этапе выполнения шага 4 приведенной выше последовательности, в

момент подтверждения процессором прерывания сигналом INTA, контроллер выставляет на шину данных команду RST, содержащую в себе номер источника запроса прерывания. Код этой команды имеет следующий вид:

$RST = 11NNN111_2$,

где NNN – это двоичный код устройства, запросившего прерывание. В результате дешифрации процессором команды RST формируется стартовый адрес (вектор) подпрограммы-обработчика прерывания:

Address = 0000 0000 00NN $N000_2$.

Например, прерывание от устройства №4 сформирует команду RST 4 (код $11\underline{100}111_2$), которой соответствует стартовый адрес 0020h (0000 0000 00NN $N000_2$). Число NNN также определяет максимальное количество внешних устройств, которое для КР580ВН59 равно восьми.

6.3. Ввод/вывод в режиме прямого доступа к памяти

Этот метод используется для скоростных внешних запоминающих устройств (ВЗУ), таких, например, как накопители на жестких магнитных дисках. В них обмен производится блоками фиксированного размера от 128 байт и более, причем этот обмен должен осуществляться в строгой последовательности без пропусков, так как пропуск хотя бы одного байта вызовет необходимость в повторном обмене. При этом время, отводимое на обмен одним байтом, строго ограничено скоростью перемещения магнитного носителя относительно магнитных головок и не превышает, как правило, нескольких микросекунд. Обеспечить обмен большими блоками данных с ВЗУ как при помощи программно-управляемого обмена, так и в режиме прерываний процессора невозможно, так как на обмен каждым байтом затрачивается несколько команд, суммарное время выполнения которых превышает допустимое время на обмен одним байтом с ВЗУ.

Поэтому высокоскоростной обмен производится в режиме прямого доступа к памяти (ПДП) без участия процессора. При этом ряд функций реализуется аппаратно, а время обмена одним байтом данных равно одному циклу обращения к памяти. Причем обменом управляет не программа, а специальный контроллер прямого доступа к памяти (микросхема КР580ВТ57 в комплекте КР580).

Существует две разновидности ПДП. В режиме «*прозрачного ПДП*» «передача данных выполняется без информирования процессора, для чего используются те интервалы машинных циклов, когда процессор не обращается к памяти, а выполняет внутренние преобразования данных. Такими интервалами в процессоре КР580 являются такты Т4 и Т5 (части цикла, используемые для внутренних операций). Процессор идентифицирует эти интервалы специальным сигналом, означающим доступность системной шины. Производительность процессора в таком режиме не уменьшается, но сами передачи носят нерегулярный характер, что ведет к уменьшению скорости передачи данных.

Другим способом является «*ПДП с приостановкой процессора*», при котором выполнение команд задерживается на несколько тактов. Как показано на рис. 33, контроллер непосредственно связан с памятью микроЭВМ через шины данных и адреса. При этом возникает проблема совместного использования шин процессором и контроллером ПДП. Для исключения конкуренции двух устройств за право владения шиной контроллер ПДП снабжен двумя управляющими линиями: «Захват шин» (HOLD) и «Подтверждение захвата» (HLDA).

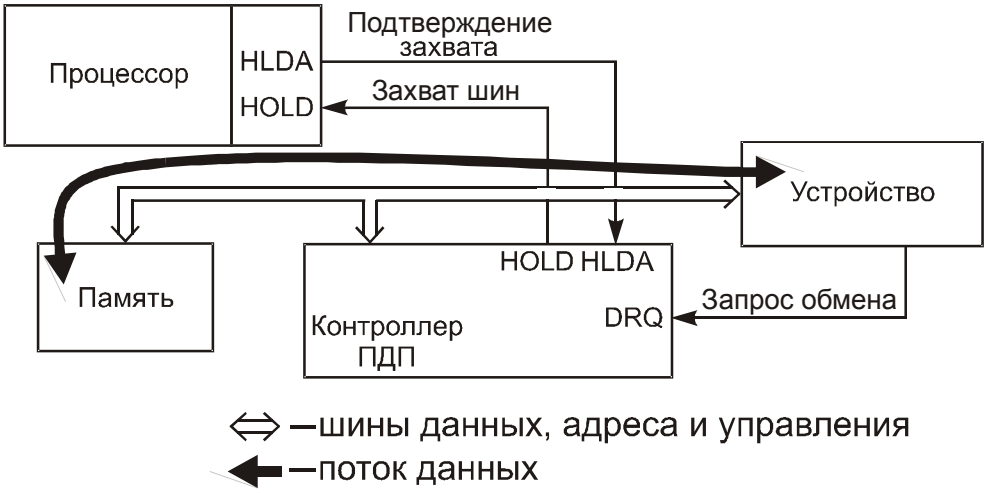


Рис. 33. Организация ввода/вывода в режиме ПДП

Если требуется обмен данными, то периферийное устройство адресует запрос обмена (сигнал DRQ) контроллеру ПДП. После получения от устройства такого запроса контроллер ПДП посылает на вход HOLD микропроцессора сигнал «Захват шин». Процессор, получив этот сигнал, приостанавливает выполнение очередной команды (закончив очередной машинный цикл), отключается от шин системного интерфейса (путем перехода шин в состояние высокого сопротивления) и выдает на системный интерфейс управляющий сигнал «Подтверждение захвата». С этого момента все шины управляются контроллером ПДП, который осуществляет обмен побайтно или блоками данных с памятью микроЭВМ и затем, сняв сигнал захвата, контроллер возвращает управление системным интерфейсом процессору. Как только контроллер ПДП будет готов к обмену следующим блоком данных, он вновь «захватывает» цикл процессора и т.д. В промежутках между сигналами подтверждения процессор может продолжать выполнение команд программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме программного ввода/вывода.

Применение в микроЭВМ режима ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т.е. число данных, подлежащих пересылке. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байт. Перед началом обмена в режиме ПДП необходимо выполнить подготовительные операции по записи в указанные регистры контроллера ПДП начального адреса выделенной внешнему устройству

памяти и ее объема в байтах или словах в зависимости от того, какими порциями информации ведется обмен.

6.4. Параллельная передача данных

Параллельная передача данных между интерфейсом и ВУ является наиболее простым способом обмена. При этом все биты данных передаются одновременно, за один такт. Очевидно, что такой способ обеспечивает высокую скорость обмена, но требует большого числа линий связи (нерационален, например, для подключения удаленных устройств). Однако в связи с сильными взаимными наводками между параллельными линиями, не удается передавать данные с большой частотой. Поэтому в современных вычислительных системах даже для соединения внутренних узлов все чаще используют последовательный интерфейс (например шина PCI Express).

В настоящее время параллельные интерфейсы стали обязательными компонентами практически всех микропроцессорных систем (рис. 34). Например, в персональном компьютере параллельным является порт LPT. Для организации параллельной передачи данных помимо шины данных, количество линий в которой равно числу одновременно передаваемых бит данных, используется минимальное количество управляющих сигналов.

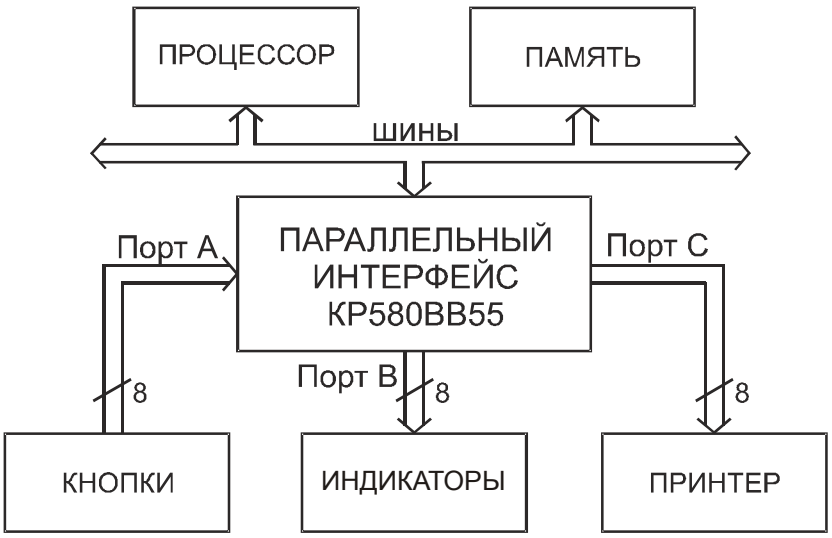


Рис. 34. Параллельный интерфейс в структуре микроЭВМ

Как пример типичного программируемого параллельного интерфейса рассмотрим микросхему КР580ВВ55 (рис. 35). Она предназначена для организации обмена информации между МП и внешними устройствами в параллельном 8-разрядном коде. Подключение периферийного оборудования производится через три двунаправленных параллельных 8-битных порта A, B и C.

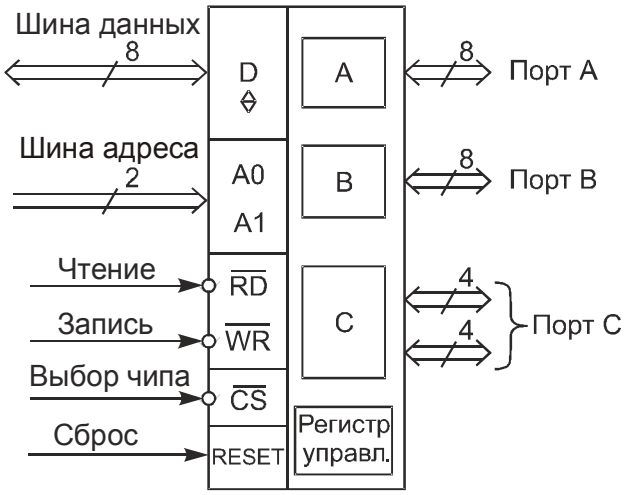


Рис. 35. Программная модель КР580ВВ55

Интерфейс подключается к системной шине с помощью 14 линий:

$D_{0\text{-}7}$ – двунаправленная шина данных с третьим состоянием;

A_0, A_1 – линии адреса, которые выбирают внутренний регистр интерфейса, подключаемый к шине данных: 00 – порт A, 01 – порт B, 10 – порт C и 11 – регистр управления;

RD – чтение информации из адресуемого линиями $A_{0,1}$ регистра на шину данных (активный уровень – низкий);

WR – запись информации с шины данных в адресуемый внутренний регистр интерфейса (активный уровень – низкий);

CS – выбор микросхемы (Chip Select) с низким активным уровнем;

RESET – сигнал сброса для приведения интерфейса в исходное состояние, при подаче которого регистр управления обнуляется, а все три порта переводятся в режим ввода.

При наличии системного контроллера программирование и обмен данными осуществляется командами ввода IN и вывода OUT. Приемником и источником данных является аккумулятор микропроцессора. При отсутствии системного контроллера порты оказываются в общем адресном пространстве памяти, и ввод/вывод через порты производится командами пересылок между ячейками памяти.

Входы $A_{0,1}$ интерфейса обычно подключаются к младшим линиям шины адреса, а вход CS используется, если в системе задействовано несколько микросхем портов. Тогда дополнительный дешифратор, подключенный к адресным линиям $A_{2,3,\ldots}$, определяет активную в текущий момент микросхему интерфейса. Напомним, что КР580ВМ80А способен адресовать до 256 портов ввода/вывода.

Программирование КР580ВВ55 заключается в записи в регистр управления управляющего слова, которое определяет один из трех режимов работы интерфейса:

Режим «0» *базового ввода/вывода* осуществляет простой ввод/вывод данных по трем 8-разрядным портам A, B, C и применяется для программно-управляемого обмена данными с медленнодействующими периферийными устройствами. При этом порт C может использоваться как два 4-разрядных канала, а необходимые управляющие сигналы (строб, готовность) могут формироваться программно по отдельным линиям;

Режим «1» *стробируемого ввода/вывода* предназначен для однонаправленных передач данных, инициируемых прерываниями. Информация передается через порты A и B, каждый из которых дополнен тремя управляющими сигналами (строб, готовность и прерывание), аппаратно-формируемыми на линиях порта C. Оставшиеся две линии порта C можно использовать как однобитные порты ввода/вывода;

Режим «2» *двунаправленный ввод/вывод*, при котором работает только порт A, а пять линий порта C выполняют функции квитирования и прерываний (две для готовности, две для строба и одна линия прерывания). При этом порт B может работать в режиме 0 или 1.

Сигнал прерывания может возникнуть по двум причинам: при заполнении и при освобождении входного буфера. Процессор различает эти два типа прерываний путем анализа содержимого порта C.

Существуют два формата управляющего слова (рис. 36), которые различаются старшим битом:

1XXX $XXXX_2$ – задание режима;

0XXX $XXXX_2$ – управление отдельными линиями порта C.

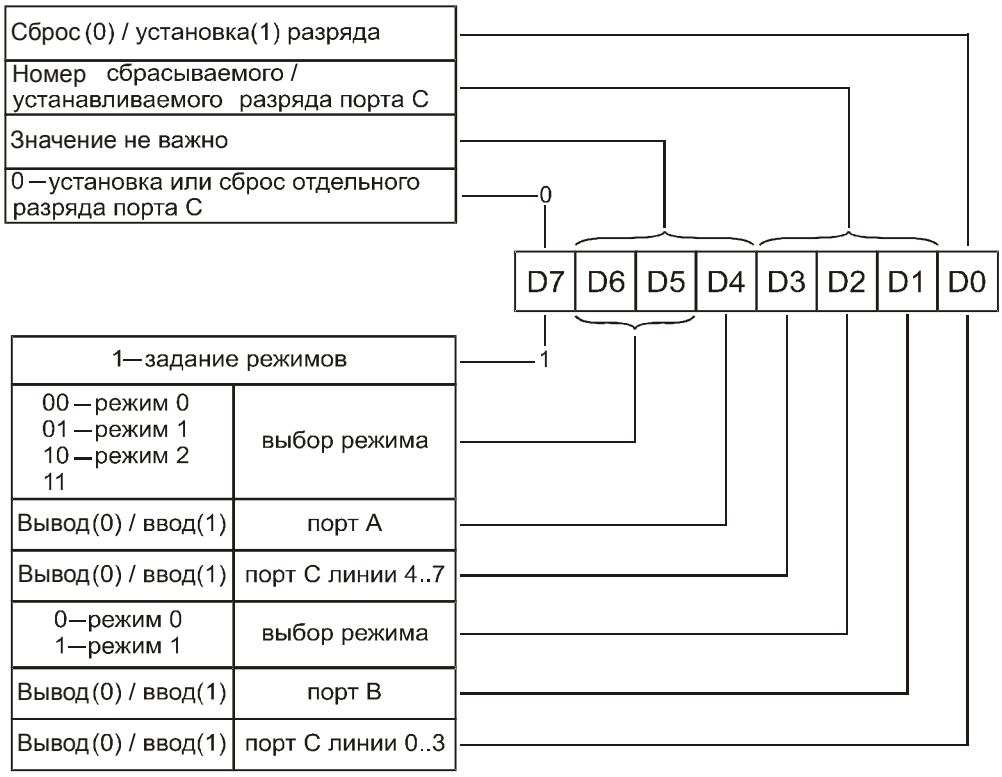


Рис. 36. Формат управляющего слова КР580ВВ55

Примечания:

• обычно программирование интерфейса осуществляется операционной системой или программой монитора, которая запускается по включению питания с адреса 0000H;

• операцию установки и сброса битов порта С можно выполнить и при помощи логических команд, однако этот способ нельзя назвать рациональным;

• управление отдельными линиями порта С требуется для формирования управляющих сигналов (стробирующих импульсов), а также для управления внешними устройствами типа реле (необходимо изменить состояние только одного бита).

6.5. Последовательная передача данных

Обмен данными с ВУ по последовательным линиям связи характерен для медленнодействующих или удаленных периферийных устройств. Применение последовательных линий связи обусловлено тем, что линии связи просты по своей организации, а функции автоматического преобразования из последовательного в параллельный формат реализуются аппаратно с помощью программируемого последовательного интерфейса. Для процессора такой интерфейс выглядит устройством параллельного ввода/вывода.

Передача данных по линии может происходить в соответствии со стандартом RS232, который описывает назначение и параметры 25 линий, однако на практике используют только часть из них – обычно 3–5 линий. Величиной скорости передачи является бод, который определяет количество переданных бит в секунду. Для повышения помехозащищенности при передаче данных на значительные расстояния уровни сигналов в линии связи не соответствуют уровням TTL, а имеют амплитуду от –12 до +12 В. Такие сигналы формируют канальные приемопередатчики серии К170.

Типичным представителем контроллера последовательного интерфейса является БИС КР580ВВ51. Она позволяет осуществить обмен данными как в программно-управляемом режиме, так и в режиме прерывания программы и предназначена для организации асинхронного либо синхронного обмена.

В *асинхронном режиме* используется стартстопный принцип, и передача может начаться в любой момент времени. При этом каждое передаваемое слово передатчик снабжает стартовым и стоповым битами, которыми осуществляется синхронизация запуска работы приемника. Затем в зависимости от применяемого кода передаются от 5 до 8 бит собственно символа. Передача символа завершается необязательным битом четного или нечетного паритета и одним, полутора или двумя стоповыми битами. После этого может быть начат цикл передачи следующего символа. Стандартный формат информационной последовательности при асинхронной связи приведен на рис. 37, а).

В *синхронном режиме* данные предаются целыми массивами слов. Для синхронизации запуска приемника используется не один бит, а одно или два слова синхронизации SYN1 и SYN2, после которых передаются 5–8-битные коды символов с необязательными битами четного или нечетного паритета (рис. 37, б). Приемник перед началом работы находится в режиме активного ожидания, принимая поступающую информацию, и осуществляет ее проверку на совпадение с кодом синхронизации. Совпадение свидетельствует о начале информационного массива данных.

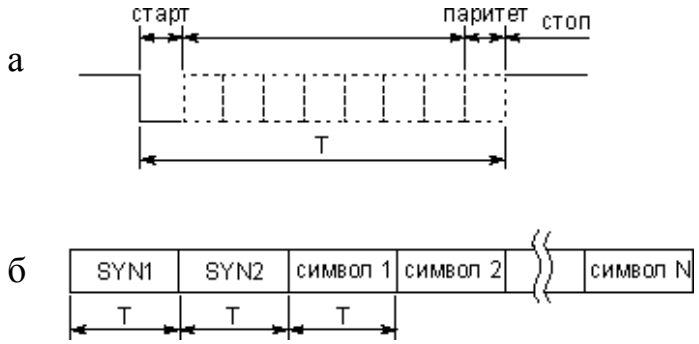


Рис. 37. Формат обмена: а – асинхронного; б – синхронного

На рис. 37, а видно, что асинхронному режиму свойственна значительная избыточность. Если, например, код символа содержит 5 бит, то вместе с ним могут передаваться до 4 служебных бит. Непроизводительное использование линии связи доходит до 44 %, поэтому асинхронный режим более выгоден с точки зрения передачи данных малой размерности, при этом скорость передачи составляет 9,6 Кбод. Для больших массивов целесообразно применение синхронного режима, в котором можно получить скорость 115,2 Кбод.

Рассмотрим основные элементы интерфейса КР580ВВ51 (рис. 38). Он имеет буферы приема/передачи данных и блок управления с регистрами, определяющими скорость обмена, длину символа, число стоповых бит, режим и условия контроля (четный или нечетный паритет). Связь с системной шиной и внешним устройством осуществляется по следующим линиям:

D – двунаправленная 8-разрядная шина данных с третьим состоянием;

C/D – команды/данные – линия идентификации передачи данных или управляющих слов;

RD – чтение информации из адресуемого линиями $A_{0,1}$ регистра на шину данных (активный уровень – низкий);

WR – запись информации с шины данных в адресуемый внутренний регистр интерфейса (активный уровень – низкий);

CLK – синхронизация – подключается к системному генератору;

CS – выбор микросхемы (Chip Select) с низким активным уровнем;

RESET – сигнал сброса для приведения интерфейса в исходное состояние, при подаче которого регистр управления обнуляется, а все три порта переводятся в режим ввода;

TxD – выход передатчика;

RxD – вход приемника;

TxC – управление скоростью передачи данных;

RxC – управление скоростью приема данных.

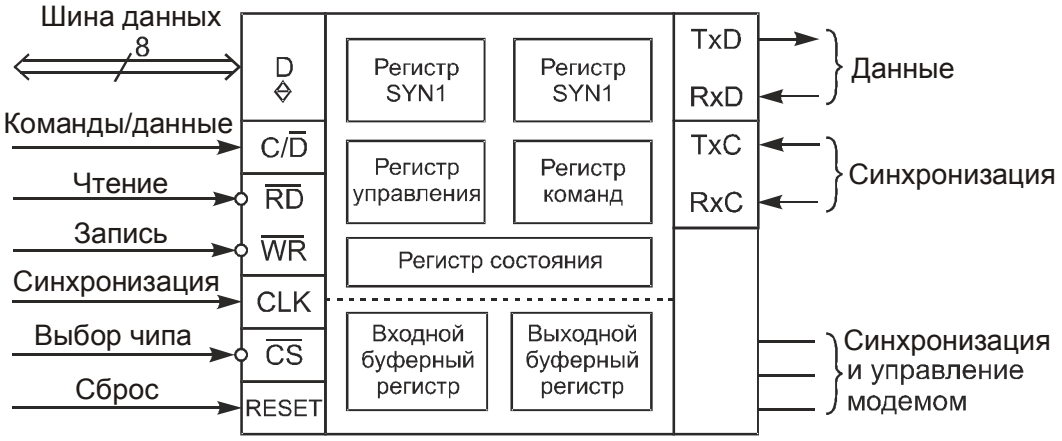


Рис. 38. Программная модель КР580ВВ51

Перед началом работы интерфейс устанавливается в исходное состояние подачей сигнала RESET. Затем необходимо произвести инициализацию управляющим словом (рис. 39).

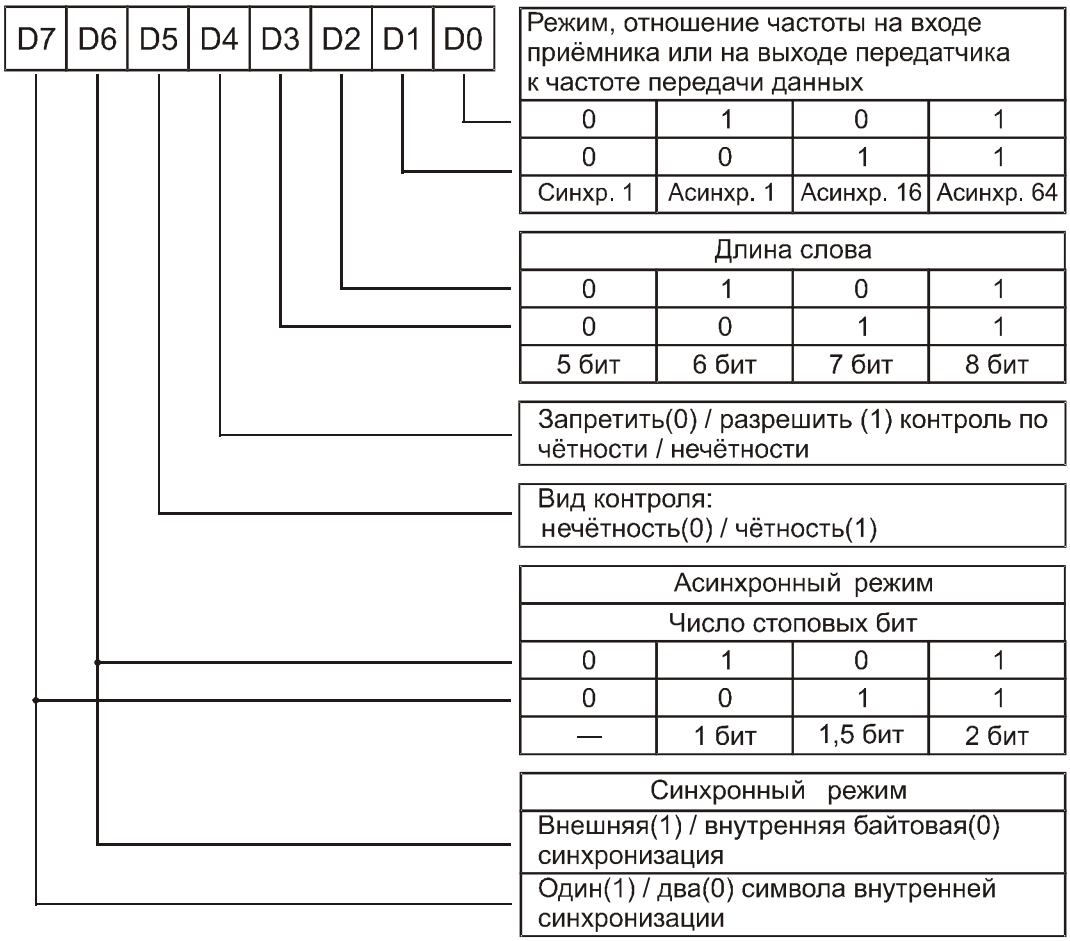


Рис. 39. Формат слова управления КР580ВВ51

Если задан синхронный обмен, то вслед за управляющим словом подаются коды одного или двух символов синхронизации. В процессе передачи данных на БИС подаются команды для оперативного указания конкретных операций: приема, передачи, перехода в режим активного ожидания (рис. 40).

Рис. 40. Формат слова команды КР580ВВ51

Для обеспечения программно-управляемого обмена предусмотрена возможность программного считывания слова-состояния (рис. 41), которое содержит информацию о текущем состоянии приемного и передающего буферов, наличии ошибок приема и передачи. Скорость обмена определяется как частотой входа CLK, так и программно и может составлять 1/16 или 1/64 этой частоты.

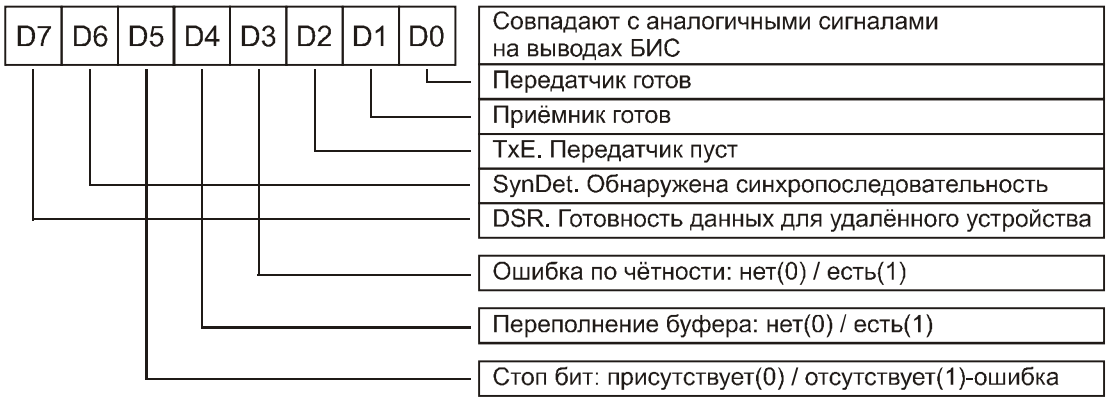


Рис. 41. Формат слова состояния КР580ВВ51

Следует отметить, что одна из команд выполняет сброс интерфейса КР580ВВ51, что эквивалентно подаче сигнала RESET.

7. Программная реализация алгоритмов управления

7.1. Программная реализация функций счета и временной задержки

При разработке программного обеспечения систем управления часто возникает необходимость выполнения определенных действий заданное количество раз. Для этого используются программные счетчики, которые реализуются двумя способами:

1. Путем инкремента содержимого регистра или ячейки памяти:

```
        MVI  B, 00H
M0:  Тело цикла
        INR  B
        MOV A, B
        CPI  30D
        JNZ  M0
```

2. Путем декремента содержимого регистра или ячейки памяти:

```
        MVI   B, 30D
M0:  Тело цикла
        DCR   B
        JNZ   M0
```

В этих примерах *тело цикла выполняется 30 раз*. Очевидно, что второй способ более удобен, так как позволяет реализовать число повторов от 1 до 256; если начальное значение B = 0, то цикл повторяется 256 раз. При необходимости в числе повторов более 255 можно использовать вложенные счетчики. Программный *счетчик с количеством повторений 1200* может выглядеть следующим образом:

```
        MVI  B, 30D      ; счетчик 30 циклов
M0:     MVI  C, 40D      ; счетчик 40 циклов
M1:  Тело цикла
        DCR  C
        JNZ  M1
        DCR  B
        JNZ  M0
```

Временные задержки реализуются путем повторения циклов с известным временем выполнения. Так как микропроцессор КР580ВМ80А работает на частоте 2 МГц, то время выполнения одного такта составляет T = 0,5 мкс. Зная количество тактов, необходимых для выполнения определенных команд, можно рассчитать время выполнения любого участка программного кода:

```
        MVI  B, 100D  ; 7 тактов, 100 повторений цикла
M0:     NOP           ; пустая операция 4 такта
        NOP           ; 4 такта
```

```
        DCR  B          ; 5 тактов
        JNZ  M0         ; 10 тактов
```

В этом примере число тактов $N = 7+(4+4+5+10)\cdot 100 = 2307$. Тогда время выполнения составляет $\tau = 2307\cdot 0{,}5 = 1153{,}5$ мкс.

Для задержки длительностью до 10 мс используются однократные циклы. Таким образом, нетрудно сформировать программную задержку на 1 мс:

```
        MVI  B, ...     ; … тактов, 100 повторений цикла
M0:     MVI  C, 86D     ; 7 тактов
M1:     NOP             ; 4 такта
        NOP             ; 4 такта
        DCR  C          ; 5 тактов
        JNZ  M1         ; 10 тактов
        DCR  B          ; 5 тактов
        JNZ  M0         ; 10 тактов
```

В этом примере число тактов $N = B\cdot(7+86\cdot(4+4+5+10)+4+5+10) = B\cdot 2000$. Для получения задержек на несколько секунд используют тройные циклы.

7.2. Программная генерация импульсов и функций времени

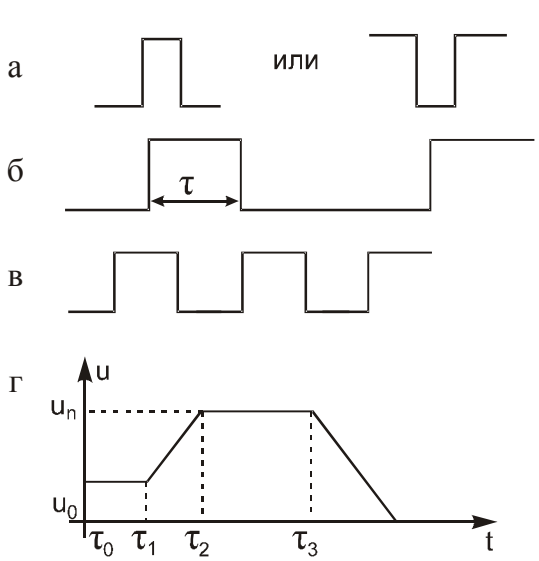


Рис. 42. Типы импульсных последовательностей и функция времени

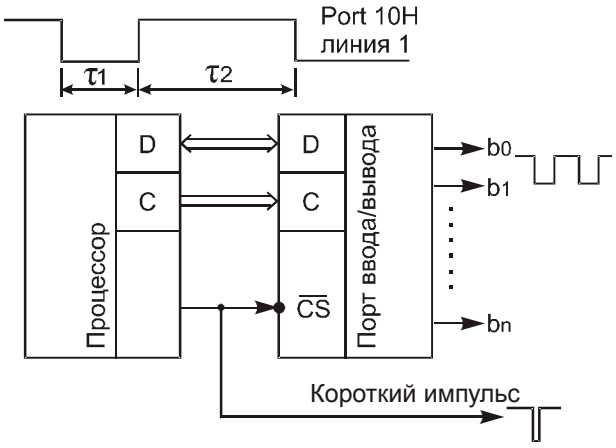
При реализации систем управления часто требуется формирование импульсов, которые могут быть трех различных типов:

1) без учета длительности (рис. 42, а);

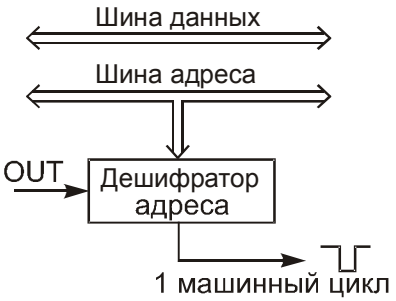
2) с заданной длительностью импульсов и длительностью пауз (рис. 42, б);

3) периодические (рис. 42, в).

Импульсы заданной длительности $\tau 2$ и периода $\tau 1$ (рис. 43, а) можно сформировать при помощи временн**о**й задержки и команд вывода OUT, выдавая их на одну из линий порта. При необходимости генерации коротких импульсов, длительность которых составляет один такт процессора, можно воспользоваться сигналом стробирования микросхемы порта либо дешифратором, подключенным к адресной шине (рис. 43, б).



а



б

Рис. 43. Реализация формирователя импульсов

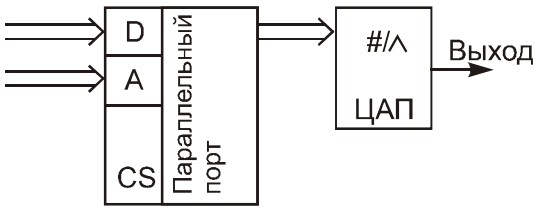


Рис. 44. Формирователь функции времени

```
M0:  MVI  A, 00000010B ; вкл. линии 1
     OUT  10H
     Задержка на τ1
     MVI  A, 00H       ; выкл. линии 1
     OUT  10H
     Задержка на τ2
     JMP  M0
```

Недостатки этой программы очевидны: она зациклена, а при выводе по команде OUT затрагивается не только заданная линия порта, но и другие линии, которые могут быть задействованы для других целей. Чтобы этого избежать, требуется иметь копию данных в памяти и изменять только один бит. Эту программу можно упросить, если необходимо обеспечить одинаковую длительность импульсов и пауз (такой сигнал называется меандром): $\tau 1 = \tau 2 = \tau$:

```
     MVI  A, 00000010B
M0:  CMA  A
     OUT  10H
     delay τ
     JMP  M0
```

Однако в этом случае ресурсы процессора задействованы полностью, поэтому более рационально использовать таймер. При управлении техпроцессами часто требуется формирование функций времени, т.е. сигналов сложной, непрямоугольной формы (рис. 42, г). Для их реализации используется ЦАП, подключаемый к порту вывода (рис. 44). Уровень напряжения на выходе ЦАП определяется кодом, формируемым на линиях порта. При этом максимальный уровень сигнала соответствует коду FFH, а минимальный – 00H. Форму функции времени удобно занести в массив $U_1, U_2, U_3, \ldots$ и ϕ_1, ϕ_2, ϕ_3. Таким образом, выдавая на линии порта определенные значения напряжения U_n через заданные промежутки времени ϕ_n, можно получить сигнал практически любой формы.

7.3. Программная реализация двухпозиционных регуляторов

Двухпозиционное (релейное) регулирование обеспечивает поддержание постоянного значения выходной величины за счет управляющих воздействий типа «включено-выключено» (например холодильника). Такое устройство (рис. 45) использует три порта: один – для чтения заданной температуры $t_{зад}$ (port 10H), второй – для чтение текущего состояния управляемого устройства $t_{тек}$ (port 11H), и третий – для управления устройством (port 12H).

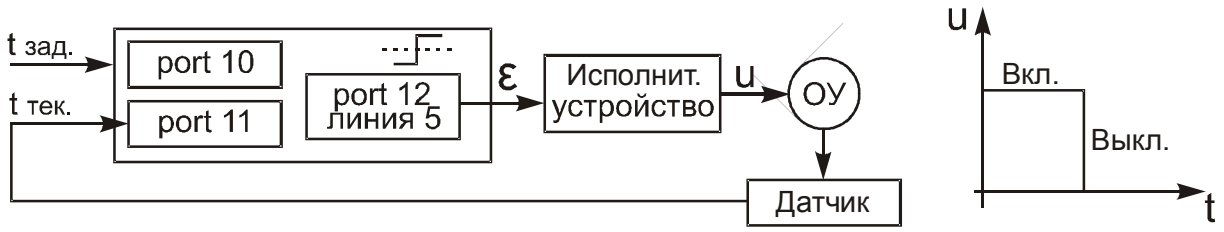


Рис. 45. Структурная схема двухпозиционного регулятора

Работу двухпозиционного регулятора можно представить следующим алгоритмом: if $t > t_{зад}$ then U = 0 else U = 1, а его программную реализацию:

```
M0: IN    10N
    MOV B, A          ; B ← t_зад
    IN    11N         ; A ← t_тек
    CMP B             ; определение знака (t_тек − t_зад)
    JP    M1
    MVI A, 00000000B  ; выключение линии 5 порта 12H
    JMP   M2
M1: MVI A, 00100000B  ; включение линии 5 порта 12H
M2: OUT 12H
    JMP   M0
```

Эта программа имеет ряд недостатков: она зациклена, причем выполняется очень быстро, поэтому ее целесообразно запускать по таймеру; ветви программы имеют разное время выполнения; частота переключения управляющего воздействия зависит от инерционности ОУ. Для малоинерционных объектов частота переключения может быть недопустимо большой или, наоборот, малой, а в рассмотренном примере выходная величина будет колебаться около заданного значения. При этом возможны два варианта:

1) амплитуда колебаний слишком велика, тогда вводят отрицательный гистерезис, т.е. упреждение (рис. 46, а);

2) амплитуда колебаний мала, но переключения происходят слишком часто, тогда вводят положительный гистерезис, т.е. пространственное запаздывание (рис. 46, б).

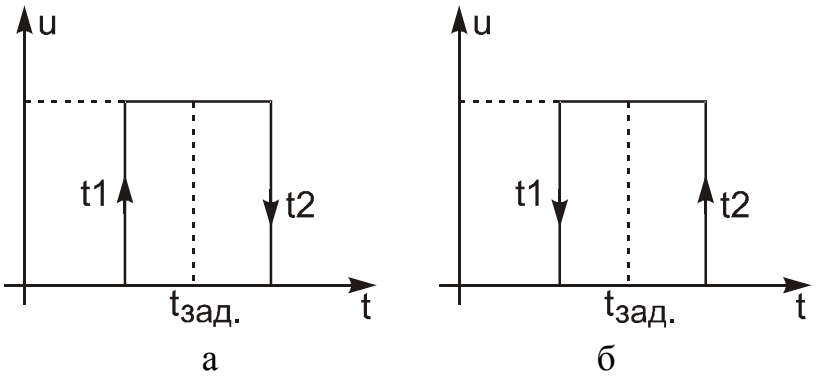


Рис. 46. Положительный (а) и отрицательный (б) гистерезис

Следует учитывать, что введение гистерезиса может отрицательно сказаться на точности, поэтому общим недостатком алгоритма двухпозиционного управления являются колебания выходной величины вокруг заданного значения, а достоинством – простота аппаратной и программной реализации.

7.4. Реализация алгоритмов пропорционального управления

Простейшим способом управления с непрерывным изменением управляющего воздействия является пропорциональное управление, которое реализуется на базе П, ПИ, ПИД регуляторов:

а) П-регулятор; $U = k \cdot \varepsilon$;

б) ПИ-регулятор: $U = k_1\varepsilon + k_2\int_0^t \varepsilon(t)dt$;

в) ПИД-регулятор: $U = k_1\varepsilon + k_2\int_0^t \varepsilon(t)dt + k_3\dot{\varepsilon}$,

где ε – разница температур. Из теории автоматического управления известно, что П-регулятор является простейшим, однако он имеет ошибку в установившемся режиме. Астатическую ошибку можно устранить с помощью ПИ-регулятора, но он обеспечивает слабую устойчивость. Поэтому для повышения устойчивости и качества переходных процессов используют пропорционально-интегрально-дифференциальное регулирование: ПИД-регулятор, который дополнительно имеет дифференциальную составляющую. В микропроцессорных системах ПИД-регуляторы реализуются при помощи рекуррентных соотношений. При этом принимают

$$\dot{\varepsilon}_k \approx \frac{\varepsilon_k - \varepsilon_{k-1}}{\Delta T}, \quad \int_0^t \varepsilon(t)dt \approx \sum_{i=1}^{k} \varepsilon_i \cdot \Delta t = \sum_{i=1}^{k-1} \varepsilon_i \Delta t + \varepsilon_k \Delta t .$$

Тогда алгоритмы, реализующие регуляторы, можно записать в виде

а) П-регулятор; $U_k = k \cdot \varepsilon$;

б) ПИ-регулятор: $U_k = A_1\varepsilon_k + A_2\varepsilon_{k-1}$;

в) ПИД-регулятор: $U_k = U_{k-1} + A_1\varepsilon_k + A_2\varepsilon_{k-1}$.

При программной реализации этого выражения возникают такие проблемы, как переполнение разрядной сетки при умножении; получение нулевого результата при вычитании близких чисел, особенно при вычислении скорости; интегральная составляющая зависит от периода квантования, откуда следует, что коэффициенты A_1, A_2 должны подстраиваться в соответствии с параметром Δt. Также следует учитывать, что программа зациклена, это приводит к необходимости ее вызова по таймеру. Проблему переполнения рассмотрим на примере П-регулятора (рис. 47). $U = k \cdot \varepsilon$, $k = 5$.

```
M0: IN   port_t
    MOV  B, A
    IN   port_G
    SUB  B          ; вычисляем ε
    MOV  B, A
    ADD  A          ; 2ε
    ADD  A          ; 4ε
    ADD  B          ; 5ε
    OUT  port_U
    JMP  M0
```

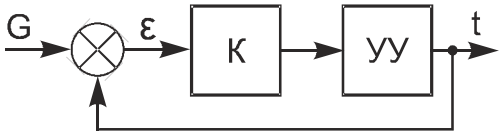


Рис. 47. П-регулятор

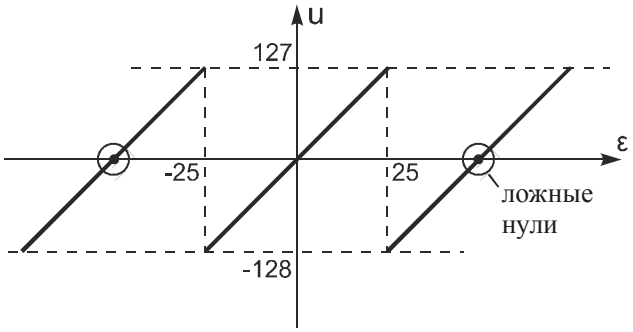


Рис. 48. Ложные нули

Работа этого алгоритма сопряжена с появлением ложных нулей в связи с переполнением разрядной сетки (рис. 48). Для исключения таких ситуаций вводят искусственное программное ограничение (рис. 49):

$$U = \begin{cases} +127, \varepsilon > 25 \\ 5 \cdot \varepsilon, \quad |\varepsilon| \le 25 \\ -127, \varepsilon < -25 \end{cases}$$

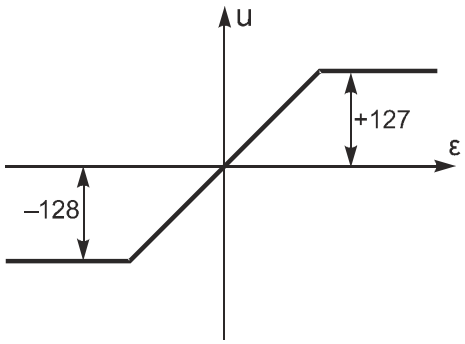


Рис. 49. Подавление ложных нулей

Аналогично реализуется ПИД-регулятор, однако при этом следует устранить переполнения при вычислении всех слагаемых и при суммировании. Необходимо учитывать также период квантования по времени, так как при малом периоде интегральная составляющая накапливается очень быстро. Выражение для ПИД-регуляторов удобно реализовать при помощи цифровых процессоров сигналов.

7.5. Позиционное и контурное управление

Рассматривая построение автоматизированных систем, можно выделить два метода формирования управляющего воздействия. В простейших случаях это может быть реализовано с помощью реле, которое формирует скачкообразное

управляющее воздействие, например включение/выключение нагревателя или привода какого-либо механизма, при этом выходом является одна координата. Такой закон управления называют позиционным. В более сложных случаях управляющее воздействие изменяется плавно, по нескольким координатам, с учетом возможностей системы. Это может быть, к примеру, управление пером графопостроителя (рис. 50). Такой закон управления называется контурным.

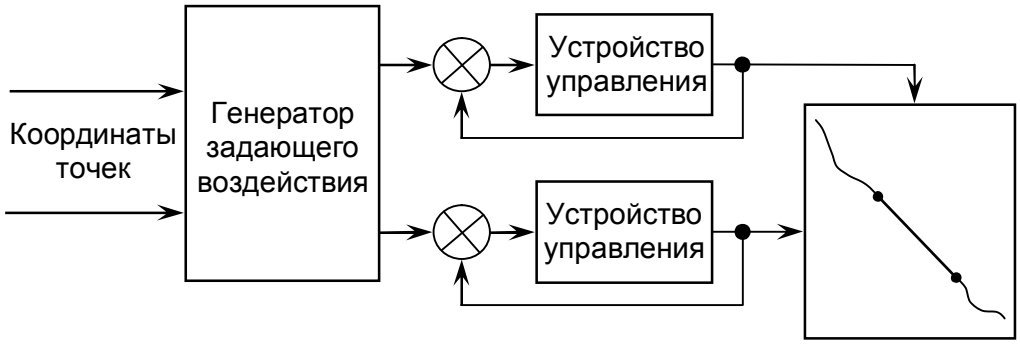


Рис. 50. Контурное управление

Позиционное и контурное управление имеет большое значение при управлении различными механическими системами, например роботами. При контурном управлении необходимо обеспечить не только переход в конечную точку, но и движение по заданной траектории.

7.6. Программная реализация алгоритмов линейной интерполяции

Алгоритмы линейной интерполяции используются для согласованного управления несколькими координатами (x, y, z) в станках с ЧПУ и роботах. Для этого используется два способа:

1) при помощи параметрического представления прямой;

2) на основе оценочной функции.

Рассмотрим *первый* способ (рис. 51). Для движения по прямой из точки (x_0, y_0) в точку (x_k, y_k) можно использовать следующий алгоритм:

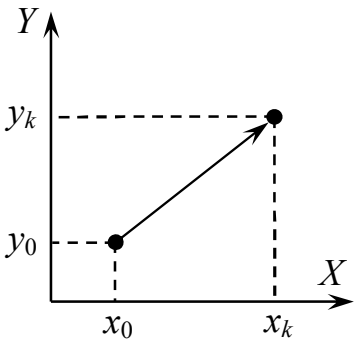


Рис. 51. Параметрическое представление прямой

for $t = 0 : \Delta t : T$

$x(t) = x_0 + V_x \cdot t$

$y(t) = y_0 + V_y \cdot t$

out (x, y)

end

К достоинству алгоритма можно отнести возможность изменения скорости в процессе движения, которая может иметь треугольный либо трапецеидальный профиль. Однако его программная реализация для микропроцессора КР580ВМ80А сопряжена с определенными трудностями, связанными с необходимостью умножения, а в этом МП такие команды отсутствуют.

Идея *второго* способа заключается в дискретном изменении координат X и Y и вычислении на каждом шаге оценочной функции f. За начальное положение примем начало координат. Пусть интерполируемый наклонный отрезок лежит в первом квадранте. Тогда его уравнение имеет вид $y = (\Delta y/\Delta x)\cdot x$. Оценочная функция $f = y - (\Delta y/\Delta x)\cdot x$. На интерполируемом отрезке прямой функция $f = 0$, выше этой прямой – $f > 0$, ниже – $f < 0$ (рис. 52), поэтому направление движения определяется знаком оценочной функции, которая пересчитывается на каждом шаге. Так как координаты x и y могут меняться только на дискретное значение, то будем измерять x и y в целых числах.

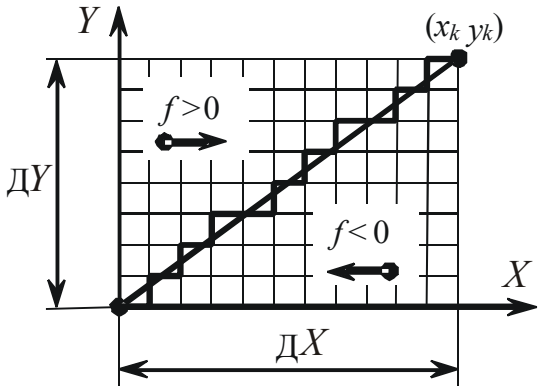


Рис. 52. Представление прямой методом оценочной функции

Алгоритм, основанный на вычислении оценочной функции, состоит в следующем:

• для текущих значений x_i и y_i вычисляется оценочная функция $f(x_i, y_i) = y_i - (\Delta y/\Delta x)\cdot x_i$;

• если функция $f(x_i, y_i) < 0$, то шаг делается по координате Y: $y_j + 1 = y_j + 1$;

• если функция $f(x_i, y_i) \ge 0$, то шаг делается по координате X: $x_j + 1 = x_j + 1$;

• вычисляется новое значение оценочной функции, и процесс повторяется. Вычисление этих операций заканчивается при попадании в точку Δx, Δy.

Поскольку в рассмотренном алгоритме важно не знание оценочной функции, а ее знак, то можно использовать функцию

$$f'(x_i, y_i) = y_i\Delta x - x_i\Delta y.$$

Пусть на j-м шаге алгоритма

$$f'(x_i, y_i) = y_j\Delta x - x_j\Delta y.$$

При шаге по оси Y

$$x_{j+1} = x_j;\ \ y_{j+1} = y_j + 1; \qquad (1)$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = (y_{j+1} + 1)\Delta x - x_j\Delta y = f'_j(x_j, y_j) + \Delta x.$$

При шаге по оси X

$$x_{j+1} = x_j + 1;\ \ y_{j+1} = y_j; \tag{2}$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = y_j \Delta x - (x_j + 1)\Delta y = f'_j(x_j, y_j) - \Delta y\,.$$

Формулы (1) и (2) позволяют вычислить оценочную функцию на каждом шаге алгоритма с использованием только операций сложения и вычитания. В программе предполагается, что отрезок прямой располагается в первом квадранте и имеет начальное значение $f(0, 0) = 0$. При интерполяции отрезков прямых, лежащих в других квадрантах, алгоритм остается тем же. Для его полной реализации необходимо учитывать номер квадранта, в котором лежит отрезок, увеличивая или уменьшая при этом соответствующие координаты x и y. Заметим, что интерполяция осуществляется в локальной системе координат. В этой же системе задаются координаты конечной точки, поэтому перед выводом необходимо преобразовать координаты из локальной системы в глобальную:

$X_{abs} = X_{abs} + X_{loc\,i}$

$Y_{abs} = Y_{abs} + Y_{loc\,i}$

Распределим регистры: B – x_i, C – y_i, D – f_i, E – x_k, H – y_k

```
        MVI  B, 00H    ; x0 = 0
        MOV  C, B      ; y0 = 0
        MOV  D, B      ; f0 = 0
M0:     MOV  A, D      ; A ← fi
        ANA  A         ;определение знака функции fi
        JM   M1
        INR  B         ; xi+1 = xi + 1
        SUB  H         ; fi+1 = fi - yk
        JMP  M2
M1:     INR  C         ; yi+1 = yi + 1
        ADD  E         ; fi+1 = fi + xk
M2:     MOV  D, A      ; D ← fi+1
        ЗАДЕРЖКА τ
        MOV  A, B
        OUT  port x
        MOV  A, C
        OUT  port y
        MOV  A, B
        ЗАДЕРЖКА τ     ; проверка окончания
```

```
CMP  E          ; определение знака (x_i − x_k)
JM  M0
MOV  A, C
CMP  H          ; определение знака (y_i − y_k)
JM  M0
```

7.7. Программная реализация алгоритмов круговой интерполяции

При интерполяции дуг окружностей можно использовать два способа:

1) при помощи параметрических уравнений;

2) метод оценочной функции.

Первый способ основан на математическом представлении дуги с помощью тригонометрических функций:

$$\begin{cases} X = x_0 + R \cdot \cos(\omega t), \\ Y = y_0 + R \cdot \sin(\omega t). \end{cases}$$

Программная реализация такого метода на КР580ВМ80А потребует не только использования операции умножения, отсутствующей в данном МП, но и вычисления функций *sin* и *cos*.

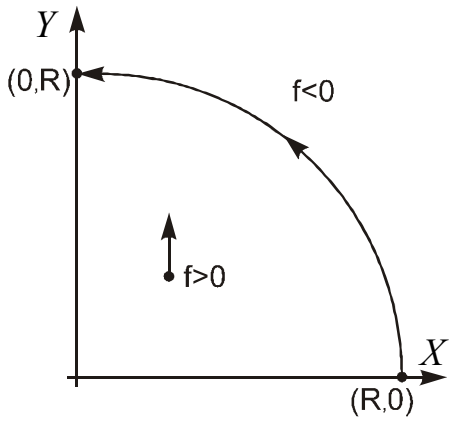


Рис. 53. Представление дуги методом оценочной функции

Метод оценочной функции (рис. 53) лишен этого недостатка. Этот метод использует оценочную функцию вида $f(x, y) = R^2 - (x^2 + y^2)$, где R – радиус дуги; x, y – расстояние от центра до текущей точки дуги вдоль осей X и Y. На линии окружности $f = 0$, внутри окружности $f < 0$, вне окружности – $f > 0$. Алгоритм интерполяции аналогичен алгоритму линейной интерполяции, изложенному выше, X и Y изменяются дискретно, и на каждом шаге вычисляется оценочная функция f; если $f > 0$ осуществляется шаг по X, если $f < 0$ – шаг по Y. При этом:

- если $f \geq 0$, то $y_{i+1} = y_i + 1$,

  $f_{i+1} = R^2 - x_i^2 - (y_i + 1)^2 = f_i - 2y_i - 1$;

- если $f < 0$, то $x_{i+1} = x_i - 1$

  $f_{i+1} = f_i + 2x_i - 1$.

Распределим регистры:

B – x_i, C – y_i, D – f_i, E – R

```
MOV  B, E       ; x_0 = R
MVI  C, 00H     ; y_0 = 0
```

```
        MVI  D, 00H     ; F0 = 0
M0:     MOV  A, D
        ANA  A          ; определение знака fi
        JM  M1
        SUB  C
        SUB  C
        DCR  A
        JNR  C
        JMP  M2
M1:     ADD  B
        ADD  B
        DCR  A
        DCR  B
M2:     MOV  D, A
        ЗАДЕРЖКА τ
        MOV  A, B
        OUT  port x
        MOV  A, C
        OUT  port y
        MOV  A, B
        CMP  E          ; flags(xi-R)
        JM  M0
        MOV  A, C
        CMP  E
        JM  M0
```

Примечание. В общем случае необходимо осуществлять интерполяцию в локальной системе координат, которая расположена в центре дуги окружности (не в начальной точке и/или в другом квадранте). Задача интерполяции для каждой дуги решается в локальной системе координат, а перед выводом необходимо перейти к глобальной системе.

Литература

1. Микропроцессорные системы. Учебное пособие для вузов. / Под ред. Д.В. Пузанкова. – СПб.: Политехника, 2002. – 935 с.
2. Бродин В.Б., Шагурин М.И. Микроконтроллеры: Архитектура, программирование, интерфейс. – М.: ЭКОМ, 1999. – 400 с.
3. Предко М. Руководство по микроконтроллерам: В 2 т. / Пер.с англ.; Под ред. И.И. Шагурина, С.Б. Лужанского. – М.: Постмаркет, 2001. – Т1 – 416 с., Т2 – 488 с.
4. Схемотехника электронных систем: Микропроцессоры и микроконтроллеры // Бойко В., Гуржий А., Жуйков В. и др.. – СПб.: BHV-Санкт-Петербург, 2004. – 464 с.
5. Бродин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. – М.: ЭКОМ, 2002. – 399 с.
6. Микропроцессорные системы и микроЭВМ в измерительной технике: Учеб. пособие для вузов. / Под ред. А.Г. Филиппова. – М.: Энергоатомиздат, 1995. – 368 с.
7. Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051. – М.: Горячая линия – Телеком, 2003. – 191 с.
8. Микропроцессоры в системах автоматического управления: INTEL 8XC196MC: Учеб. пособие / Н.Г. Бутырин, О.П. Кан, А.Л. Логинов, А.Н. Щербина. 1995. – 114 с.
9. Кузьминов А.Ю. Интерфейс RS232. Связь между компьютером и микроконтроллером. – М: Радио и связь, 2004. – 168 с.
10. Каратаев Н.В. Микроконтроллеры для бытовой аппаратуры. – М.: Додэка-XXI, 2001. – 207 с.
11. Козаченко В.Ф. Микроконтроллеры: Руководство по применению 16-разряд. микроконтроллеров Intel MCS-196/296 во встроенных системах управления. – М.: ЭКОМ, 1997. – 686 с.
12. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ.пособие. – М. : Радио и связь, 1998. – 556 с.
13. Тавернье К. PIC-микроконтроллеры: Практика применения: Пер.с фр. – М.: ДМК Пресс, 2002. – 272 с.
14. Яценков В.С. Микроконтроллеры Microchip. Практическое руководство. – М.: Горячая линия – Телеком, 2002. – 296 с.
15. Корнеев В.В., Киселев А.В. Современные микропроцессоры. – М.: Нолидж, 2000.

Св. план 2006, поз. 35

Учебное издание

Пашкевич Анатолий Павлович,
Чумаков Олег Анатольевич,
Лукьянец Степан Валерьянович

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Конспект лекций
для студентов специальностей
I – 53 01 07 «Информационные технологии и управление
в технических системах»
дневной формы обучения

В 2-х частях

Часть1

Редактор Т.Н. Крюкова
Корректор Н.В. Гриневич

Подписано в печать | Формат 60×84 1/16. | Бумага офсетная.
Гарнитура «Таймс». | Печать ризографическая. | Усл. печ. л. 4,3.
Уч.-изд.л. 3. | Тираж 200 экз. | Заказ .

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия на осуществление издательской деятельности №02330/0056964 от 01.04.2004.
Лицензия на осуществление полиграфической деятельности №02330/0131518 от 30.04.2004.
220013, Минск, П.Бровки, 6.