Министерство образования и науки Российской Федерации

Магнитогорский государственный технический
университет им. Г. И. Носова

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Утверждено Редакционно-издательским советом университета в качестве учебного пособия

Магнитогорск
2013

УДК 681.325.5(075)

Рецензенты:

Старший менеджер управления главного энергетика ОАО «ММК», кандидат технических наук
И.Л. Погорелов

Инженер сервисного центра ООО «Техноап-Инжиниринг» в г. Магнитогорске, кандидат технических наук
М.В. Коновалов

Лукьянов С.И.
Основы микропроцессорной техники: учеб. пособие / С.И. Лукьянов, Д.В. Швидченко, Е.С. Суспицын, Р.С. Пишнограев, Н.В. Швидченко, С.С. Красильников – Магнитогорск: Изд-во Магнитогорск. гос. техн. ун-та им. Г.И. Носова, 2012. 139 с.
ISBN 978-5-9967-0331-9

Излагаются основные принципы организации и функционирования центрального ядра ЭВМ, построенного на базе микропроцессора КР580ВМ80А. Приводится описание архитектуры микропроцессорной БИС. Рассматриваются особенности блоков управления, используемые способы адресации, системы команд и некоторые вопросы взаимодействия микропроцессоров с памятью и внешними устройствами. Пособие предназначено для студентов, обучающихся по специальности 210106 «Промышленная электроника» и направлению 210100 «Электроника и микроэлектроника» и в помощь студентам в закреплении знаний по курсам «Машинные языки программирования», «Основы микропроцессорной техники».

УДК 681.325.5(075)

ISBN 978-5-9967-0331-9

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современные микропроцессоры представляют собой весьма сложные по устройству изделия микроэлектроники. Многочисленные типы микропроцессоров характеризуются различными архитектурными решениями и функциональными возможностями. Микропроцессорная техника стремительно и многонаправленно развивается и совершенствуется, интегрируя новейшие достижения микроэлектроники и схемотехники.

Одним из важнейших направлений развития микропроцессорной техники является создание универсальных микропроцессоров, предназначенных для применения в вычислительных системах: персональных компьютерах, рабочих станциях, серверах, параллельных суперЭВМ.

Создание микроконтроллеров и цифровых сигнальных процессоров является также важнейшим направлением развития микропроцессорной техники.

Однокристальные микроконтроллеры выпускаются различных типов, отличаются назначением, сложностью, архитектурными решениями и т.д.

Изучение микропроцессорных средств, являясь достаточно сложной задачей, требует оптимального выбора тематики и временных затрат на их освоение. Значительный интерес для изучения представляют первые семейства 8-разрядных микропроцессорных БИС: 8080, 8085 (Intel), Z80 (Zilog), MC6800, MC6809 (Motorola), MCS6500 (MOS Technology). Эти микросхемы стали доступными и классическими примерами построения на их основе микропроцессорных систем. Особенно подробно описаны в учебно-методической и инженерно-технической литературе микропроцессорные БИС семейства 8080 и 8085 и их отечественные аналоги серий К580 и К1821. Поэтому начальные знания о микропроцессорах и микропроцессорных системах целесообразно получить, используя в качестве примеров семейства БИС серий К580, 8080. Это позволит эффективно освоить простейшие средства и далее двигаться к более сложным микропроцессорам и МП-системам.

1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Микропроцессор (**МП**) – это программно-управляемое электронное цифровое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки, выполненное на одной или нескольких интегральных схемах с высокой степенью интеграции электронных элементов.

Для организации законченных вычислительных, контрольно-измерительных или управляющих функций он должен объединяться в систему с другими элементами (память, каналы ввода-вывода, внешние устройства).

Микропроцессорный комплект (МПК) – совокупность микропроцессорных и других интегральных микросхем, совместимых по конструктивно-технологическому исполнению и предназначенных для совместного применения при построении МП, микроЭВМ и других средств вычислительной техники.

Микропроцессорная система (МП система) – информационная или управляющая система, построенная с применением микропроцессорных средств.

Архитектура МП системы – объединяет аппаратные средства и программное обеспечение, включает потребительские свойства системы. Архитектура отображает те аспекты системы, которые являются видимыми для пользователя: систему команд, режимы адресации, форматы и длину данных, набор регистров.

Физически реализованные в виде соответствующей аппаратуры функциональные блоки МП системы называются аппаратными средствами.

МикроЭВМ или микрокомпьютер – устройство обработки данных, содержащее: МП, постоянное и оперативное запоминающее устройства (ПЗУ и ОЗУ), устройства ввода-вывода информации (УВВ), периферийные устройства (ПУ) и некоторые другие схемы.

Микроконтроллер (МК) – микрокомпьютер, в основном, с небольшими вычислительными ресурсами и упрощенной системой команд, ориентированный не на производство вычислений, а на выполнение логического управления оборудованием, технологическими процессами.

Интерфейс – полная совокупность физических и логических соглашений о входных и выходных сигналах устройств с целью их сопряжения.

Совокупность аппаратных, программных и конструктивных средств, обеспечивающих взаимодействие функциональных устройств ЭВМ, называется внутренним системным интерфейсом (ШУ, ШД, ША).

Микропроцессорное устройство (МПУ) – совокупность МП средств, элементов и деталей, обладающих конструктивным и функциональным единством – ЭВМ, ПЭВМ, микроЭВМ, МК.

Контроллер – устройства ввода-вывода информации на периферийное (внешнее) устройство.

Адаптер – устройство для согласования физических параметров (входных, выходных сигналов) устройств с целью их сопряжения.

Логическая организация (архитектура) микропроцессоров (микропроцессорных средств) ориентирована на достижение универсальности применения, высокой производительности и технологичности.

Универсальность МП (микропроцессорных средств) определяется возможностью их разнообразного применения и обеспечивается программным управлением МП, позволяющим производить программную настройку МП на выполнение определенных функций магистрально-модульным принципом построения, а также специальными аппаратурно-логическими средствами: сверхоперативной регистровой памятью, многоуровневой системой прерывания, прямым доступом к памяти, программно-настраиваемыми схемами управления вводом/выводом и т.п.

Относительно высокая производительность МП достигается использованием для их построения быстродействующих больших и сверхбольших интегральных электронных схем и специальных архитектурных решений, таких, как стековая память, разнообразные способы адресации, гибкая система команд (или микрокоманд) и др.

Технологичность микропроцессорных средств обеспечивается модульным принципом конструирования, который предполагает реализацию этих средств в виде набора функционально законченных БИС, просто объединяемых в соответствующие вычислительные устройства, машины и комплексы.

2. КЛАССИФИКАЦИЯ МИКРОПРОЦЕССОРОВ

На рис. 1 приведена классификация микропроцессоров по важнейшим признакам.

1. ***По числу больших интегральных схем*** (БИС) в микропроцессорном комплекте различают **однокристальные**, **многокристальные** и **многокристальные секционные** микропроцессоры.

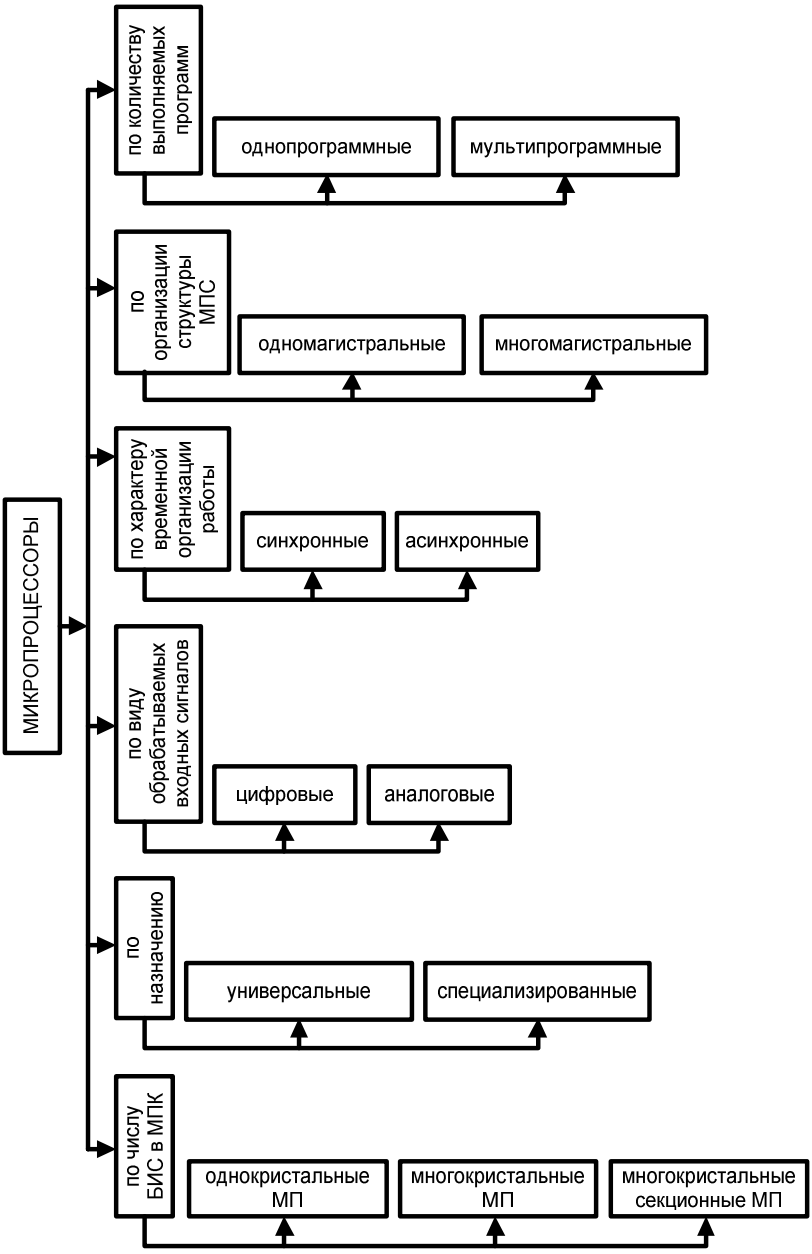


Рис. 1. Классификация микропроцессоров

Процессоры даже самых простых ЭВМ имеют сложную функциональную структуру, содержат большое количество электронных элементов и множество разветвленных связей. Изменять структуру процессора необходимо так, чтобы полная принципиальная схема или ее части имели количество элементов и связей, совместимое с возможностями БИС. При этом микропроцессоры приобретают внутреннюю магистральную архитектуру, т.е. в них к единой внутренней информационной магистрали подключаются все основные функциональные блоки (арифметико-логический, рабочих регистров, стека, прерываний, интерфейса, управления и синхронизации и др.).

Для обоснования классификации микропроцессоров по числу БИС надо распределить все аппаратные блоки процессора между основными тремя функциональными частями: операционной, управляющей и интерфейсной. Сложность операционной и управляющей частей процессора определяется их разрядностью, системой команд и требованиями к системе прерываний; сложность интерфейсной части разрядностью и возможностями подключения других устройств ЭВМ (памяти, внешних устройств, датчиков и исполнительных механизмов и др.). Интерфейс процессора содержит несколько десятков информационных шин данных (***ШД***), адресов (***ША***) и управления (***ШУ***).

Однокристальные микропроцессоры получаются при реализации всех аппаратных средств процессора в виде одной БИС или СБИС (сверхбольшой интегральной схемы). По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных микропроцессоров улучшаются. Однако возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса. Для получения многокристального микропроцессора необходимо провести разбиение его логической структуры на функционально законченные части и реализовать их в виде БИС (СБИС). Функциональная законченность БИС многокристального микропроцессора означает, что его части выполняют заранее определенные функции и могут работать автономно.

На рис. 2, а показано функциональное разбиение структуры процессора при создании трехкристального микропроцессора (пунктирные линии), содержащего БИС операционного (***ОП***), БИС управляющего (***УП***) и БИС интерфейсного (***ИП***) процессоров.

Операционный процессор служит для обработки данных, управляющий процессор выполняет функции выборки, декодирования и вычисления адресов операндов и также генерирует последовательности микрокоманд. Автономность работы и большое

быстродействие ***УП*** позволяет выбирать команды из памяти с большей скоростью, чем скорость их исполнения ***ОП***. При этом в ***УП*** образуется очередь еще не исполненных команд, а также заранее подготавливаются те данные, которые потребуются ***ОП*** в следующих циклах работы. Такая опережающая выборка команд экономит время ***ОП*** на ожидание операндов, необходимых для выполнения команд программ. Интерфейсный процессор позволяет подключить память и периферийные средства к микропроцессору; он, по существу, является сложным контроллером для устройств ввода/вывода информации. Интерфейсный процессор выполняет также функции канала прямого доступа к памяти.

Выбираемые из памяти команды распознаются и выполняются каждой частью микропроцессора автономно и поэтому может быть обеспечен режим одновременной работы всех интегральных схем МП, т.е. конвейерный поточный режим исполнения последовательности команд программы (выполнение последовательности с небольшим временным сдвигом). Такой режим работы значительно повышает производительность микропроцессора.

Многокристальные секционные микропроцессоры получаются в том случае, когда в виде БИС реализуются части (секции) логической структуры процессора при функциональном разбиении ее вертикальными плоскостями (рис. 2, б). Для построения многоразрядных микропроцессоров при параллельном включении секций БИС в них добавляются средства «стыковки».

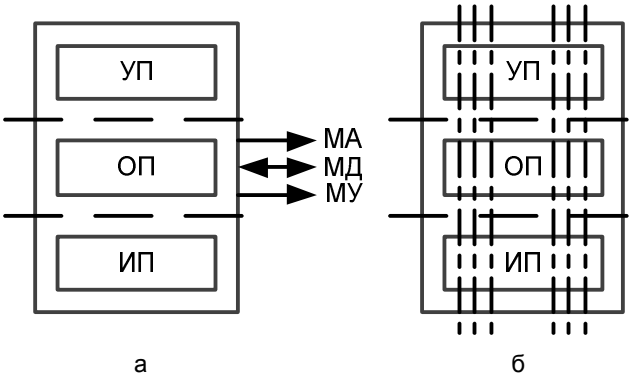


Рис. 2. Функциональная структура процессора (а) и её разбиение для реализации процессора в виде комплекта секционных БИС (б)

Для создания высокопроизводительных многоразрядных микропроцессоров требуется столь много аппаратных средств, не реализуемых в доступных БИС, что может возникнуть необходимость

еще и в функциональном разбиении структуры микропроцессора горизонтальными плоскостями. В результате рассмотренного функционального разделения структуры микропроцессора на функционально и конструктивно законченные части создаются условия реализации каждой из них в виде БИС. Все они образуют комплект секционных интегральных схем МП.

Таким образом, микропроцессорная секция – это БИС, предназначенная для обработки нескольких разрядов данных или выполнения определенных управляющих операций. Секционность интегральных схем МП определяет возможность «наращивания» разрядности обрабатываемых данных или усложнения устройств управления микропроцессора при «параллельном» включении большего числа БИС.

Однокристальные и трехкристальные интегральные схемы МП, как правило, изготовляют на основе микроэлектронных технологий униполярных полупроводниковых приборов, а многокристальные секционные БИС МП – на основе технологии биполярных полупроводниковых приборов. Использование многокристальных микропроцессорных высокоскоростных биполярных БИС, имеющих функциональную законченность при малой физической разрядности обрабатываемых данных и монтируемых в корпус с большим числом выводов, позволяет организовать разветвление связи в процессоре, а также осуществить конвейерные принципы обработки информации для повышения его производительности.

2. ***По назначению*** различают **универсальные** и **специализированные** микропроцессоры.

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Специализация МП, т.е. его проблемная ориентация на ускоренное выполнение определенных функций, позволяет резко увеличить эффективную производительность при решении только определенных задач.

Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения, МП для обработки данных в различных областях применений и т.д. С помощью специализированных МП можно эффективно решать новые сложные задачи параллельной обработки данных.

3. ***По виду обрабатываемых входных сигналов*** различают **цифровые** и **аналоговые** микропроцессоры.

Сами микропроцессоры – цифровые устройства, однако, могут иметь встроенные аналого-цифровые и цифроаналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход. С архитектурной точки зрения такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов и называются аналоговыми микропроцессорами. Они выполняют функции любой аналоговой схемы (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.). При этом применение аналогового микропроцессора значительно повышает точность обработки аналоговых сигналов и их воспроизводимость, а также расширяет функциональные возможности за счет программной «настройки» цифровой части микропроцессора на различные алгоритмы обработки сигналов.

Обычно в составе однокристальных аналоговых МП имеется несколько каналов аналого-цифрового и цифроаналогового преобразования. В аналоговом микропроцессоре разрядность обрабатываемых данных достигает 24 бит и более, большое значение уделяется увеличению скорости выполнения арифметических операций.

Отличительная черта аналоговых микропроцессоров – способность к переработке большого объема числовых данных, т.е. к выполнению операций сложения и умножения с большой скоростью при необходимости даже за счет отказа от операций прерываний и переходов. Аналоговый сигнал, преобразованный в цифровую форму, обрабатывается в реальном масштабе времени и передается на выход обычно в аналоговой форме через цифроаналоговый преобразователь. При этом, согласно теореме Котельникова, частота квантования аналогового сигнала должна вдвое превышать верхнюю частоту сигнала.

Сравнение цифровых микропроцессоров производится сопоставлением времени выполнения ими списков операций. Сравнение же аналоговых микропроцессоров производится по количеству эквивалентных звеньев аналого-цифровых фильтров рекурсивных фильтров второго порядка. Производительность аналогового микропроцессора определяется его способностью быстро выполнять операции умножения: чем быстрее осуществляется умножение, тем больше эквивалентное количество звеньев фильтра в аналоговом преобразователе и тем более сложный алгоритм преобразования цифровых сигналов можно задавать в микропроцессоре.

Одним из направлений дальнейшего совершенствования аналоговых микропроцессоров является повышение их универсальности и гибкости. Поэтому вместе с повышением скорости обработки большого объема цифровых данных будут развиваться средства обеспечения развитых вычислительных процессов обработки цифровой информации за счет реализации аппаратных блоков прерывания программ и программных переходов.

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

Синхронные микропроцессоры – микропроцессоры, в которых 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.

5. ***По организации структуры микропроцессорных систем*** различают **одно-** и **многомагистральные** микроЭВМ.

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. 3).

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 бу-

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

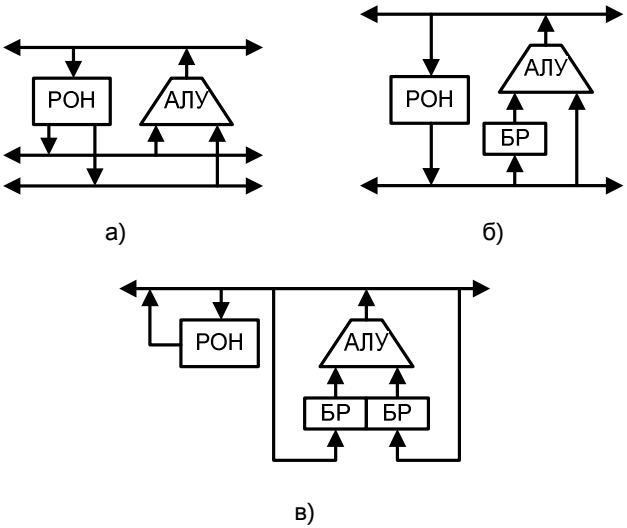


Рис. 3. Топология: а – трехшинная; б – двухшинная; в – одношинная

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*.

В одномагистральных микроЭВМ все устройства имеют одинаковый интерфейс и подключены к единой информационной магистрали, по которой передаются коды данных, адресов и управляющих сигналов.

В многомагистральных микроЭВМ устройства группами подключаются к своей информационной магистрали. Это позволяет осуществить одновременную передачу информационных сигналов по нескольким (или всем) магистралям. Такая организация систем усложняет их конструкцию, однако увеличивает производительность.

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 существуют 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. 4), по которой в разные моменты времени передаются сигналы как адреса, так и данных.

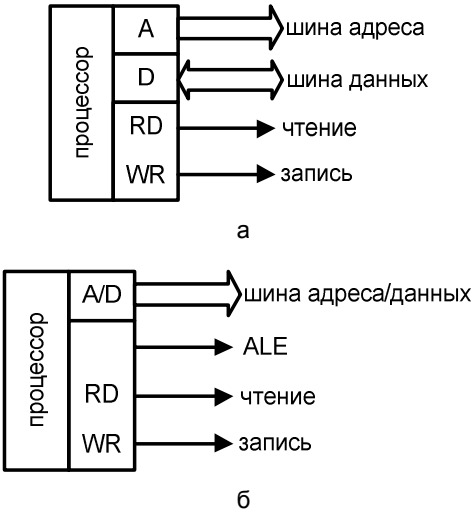


Рис. 4. Микропроцессор: а – с раздельными шинами; б – с совмещёнными шинами

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. 3, в. С каждым проводником общей шины связаны три входа (*РОН*, два буферных регистра) и два выхода (*РОН*, *АЛУ*). Известны три способа решения этой задачи (рис. 5): логическим объединением, объединение с помощью схем с открытым коллектором и объединение с использованием схем с тремя устойчивыми состояниями.

Логическое объединение (рис. 5, а) выполняется с помощью схемы ИЛИ, на входы которой при подаче управляющего сигнала Y_1 на общей шине появляется выходной сигнал *АЛУ*, а при подаче сигнала Y_2 – выходной сигнал *РОН*. Этот способ может использоваться при создании внутренней шины (на кристалле).

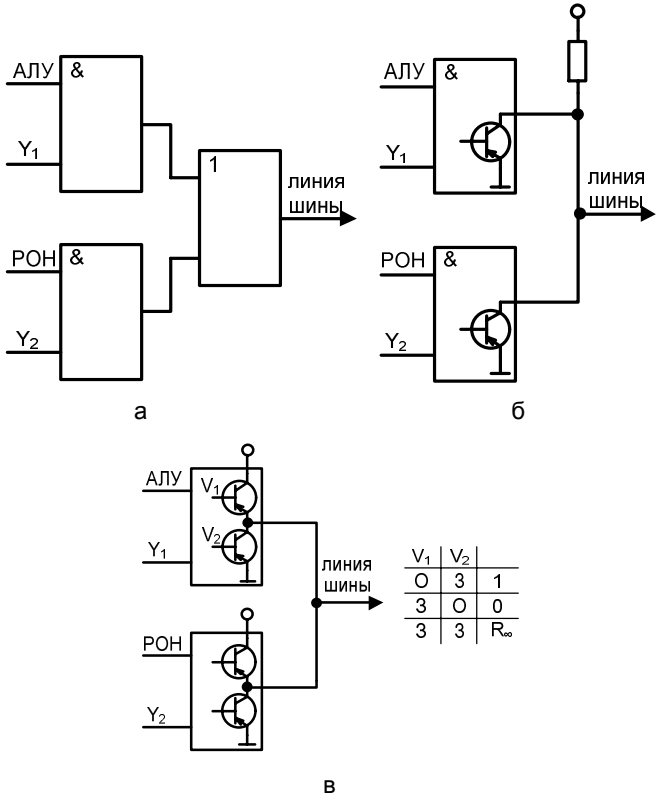


Рис. 5. Подключение нескольких источников к одной шине:
а – логическим объединением, б – «монтажным ИЛИ», в – объединением на базе схем с тремя устойчивыми состояниями

Объединение с помощью схем с открытым коллектором характеризуется электрическим соединением выходов нескольких логических элементов. Поэтому этот способ часто называют «монтажным ИЛИ» (рис. 5, 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 (рис. 5, в) 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.

6. ***По количеству выполняемых программ*** различают **одно-** и **многопрограммные** микропроцессоры.

В однопрограммных микропроцессорах выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы.

В много- или мультипрограммных микропроцессорах одновременно выполняется несколько (обычно несколько десятков) программ. Организация мультипрограммной работы микропроцессорных управляющих систем позволяет осуществить контроль за состоянием и управлением большого числа источников или приемников информации.

7. ***По быстродействию:*** **малые** (до 100 тыс. операций в секунду), **средние** (до 500 тыс.), **большие** (до 1,5 млн), **сверхбольшие** (свыше 1,5 млн операций в секунду).

Здесь указано быстродействие центрального процессора. Реальное быстродействие ЭВМ существенно ниже за счет «медленных» устройств ввода-вывода.

3. СТРУКТУРА ТИПИЧНОЙ МИКРОЭВМ

Структура микроЭВМ является магистрально-модульной. В такой структуре имеется группа магистралей (шин), к которым подключаются различные модули (блоки), обменивающиеся между

собой информацией по одним и тем же шинам поочередно. На рис. 6 показана структура типичной трехмагистральной микроЭВМ.

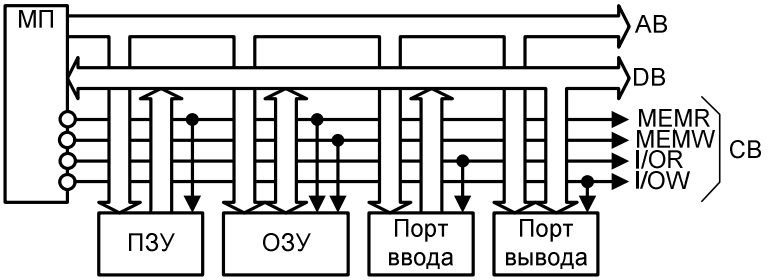


Рис. 6. Структура типичной микроЭВМ

Данная структура сильно упрощена, поскольку МПС (микроЭВМ) в своем составе содержит такие устройства, как шинные формирователи, параллельные и последовательные адаптеры, контроллер прерываний, контроллер прямого доступа к памяти, таймеры и т.д.

3.1. **Система шин микроЭВМ**

Шина представляет собой набор электрических проводников, объединенных функционально и часто физически. Шины объединяют все функциональные блоки микроЭВМ и обеспечивают обмен данными. На рис. 7 показано условное изображение шины, где ***n*** – разрядность шины (число проводников).

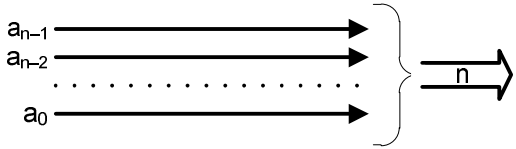


Рис. 7. Условное обозначение шины

В составе типичной микроЭВМ выделяют три типа шин:

AB (*Address Bus*) – шина адреса;

DB (*Data Bus*) – шина данных;

CB (*Control Bus*) – шина управления.

Шина адреса (*AB*) предназначена для однозначного определения элемента микроЭВМ (например, ячейки памяти или устройства ввода/вывода) и является однонаправленной. Шина данных (*DB*) служит для обмена данными между элементами микроЭВМ (двунаправленная). Шина управления (*CB*) предназначена для согласования работы элементов микроЭВМ. В структуре трехмагистральной микроЭВМ (см. рис. 6) приведены 4 сигнала шины управления:

MEMR (*Memory Read*) – сигнал чтения из памяти;
MEMW (*Memory Write*) – сигнал записи в память;
I/OR (*Input/Output Read*) – сигнал чтения из внешнего устройства;
I/OW (*Input/Output Write*) – сигнал записи во внешнее устройство.

Это далеко не весь список управляющих сигналов в МПС.

3.2. **Микропроцессор**

Как уже отмечалось ранее, микропроцессор – функционально законченное устройство, работающее по программе, подаваемой на ее вход. МП в микроЭВМ является центральным узлом по обработке информации. Он вырабатывает большинство управляющих сигналов МПС, выдает адреса ячеек памяти, адреса устройств ввода/вывода, из которых следует считать или записать данные. Микропроцессор ведет обмен данными практически со всеми узлами микроЭВМ и осуществляет за ними контроль.

3.3. **Память**

Память микроЭВМ представляет собой совокупность регистров (ячеек), предназначенных для хранения информации в двоичной форме. Каждая ячейка имеет уникальный адрес, что обеспечивает возможность доступа к ней. Адрес представляет собой двоичное слово, длина которого определяет количество ячеек, которое может быть адресовано. Совокупность всех адресов образует адресное пространство микроЭВМ.

Если в *АВ* ***m*** разрядов, то объем адресного пространства

$$M = 2^m.$$

Память имеет две классические разновидности: постоянное запоминающее устройство (*ПЗУ*) и оперативное запоминающее устройство (*ОЗУ*).

ПЗУ хранит фиксированные программы и данные, оно является энергонезависимым и при выключении питания информацию не теряет. *ОЗУ* хранит оперативные данные (изменяемые программы, промежуточные результаты вычислений и пр.) и теряет свое содержимое вместе с потерей питания. Но в *ОЗУ,* в отличие от *ПЗУ,* можно записывать данные, а не только читать в процессе работы.

3.4. **Порты**

Порты ввода/вывода играют роль посредника между микроЭВМ и внешними устройствами. Как и ячейки памяти, порты имеют адрес, что позволяет иметь множество портов в составе одной микроЭВМ. Роль портов могут выполнять микросхемы буферных

регистров, микросхема программируемого параллельного интерфейса (*ППИ*) и др.

3.5. **Работа микроЭВМ**

В качестве примера, иллюстрирующего работу микроЭВМ, рассмотрим процедуру, для реализации которой нужно выполнить следующую последовательность элементарных операций:

1. Нажать клавишу с буквой «А» на клавиатуре.
2. Поместить букву «А» в память микроЭВМ.
3. Вывести букву «А» на экран дисплея.

Это типичная процедура ввода-сохранения-вывода, рассмотрение которой дает возможность пояснить принципы использования некоторых устройств, входящих в микроЭВМ.

На рис. 8 приведена подробная диаграмма выполнения процедуры ввода-сохранения-вывода. Команды уже загружены в первые шесть ячеек памяти. Хранимая программа содержит следующую цепочку команд:

1. Ввести данные из порта ввода 1 (клавиатура).
2. Запомнить данные в ячейке памяти 200.
3. Переслать данные в порт вывода 10 (дисплей).

Рассмотрим прохождение команд и данных внутри микроЭВМ с помощью занумерованных кружков на диаграмме.

1. МП выдает адрес *100* на шину адреса (*ША*). По шине управления (*ШУ*) поступает сигнал, устанавливающий память программ (конкретную микросхему) в режим считывания.

2. ЗУ программ пересылает первую команду («Ввести данные») по шине данных (*ШД*) и МП получает это закодированное сообщение. Команда помещается в регистр команд. МП декодирует (интерпретирует) полученную команду и определяет, что для команды нужен операнд.

3. МП выдает адрес *101* на *ША*; *ШУ* используется для перевода памяти программ в режим считывания.

4. Из памяти программ на *ШД* пересылается операнд «Порт 1». Этот операнд находится в программной памяти в ячейке *101*. Код операнда (содержащий адрес порта 1) передается по *ШД* к МП и направляется в регистр команд. МП теперь декодирует полную команду («Ввести данные из порта 1»).

5. МП, используя *ША* и *ШУ*, связывающие его с устройством ввода, открывает порт 1. Цифровой код буквы «А» передается в аккумулятор внутри МП и запоминается. Важно отметить, что при обработке каждой программной команды МП действует согласно микропроцедуре выборки-декодирования-исполнения.

6. МП обращается к ячейке *102* по *ША*. *ШУ* используется для перевода памяти программ в режим считывания.

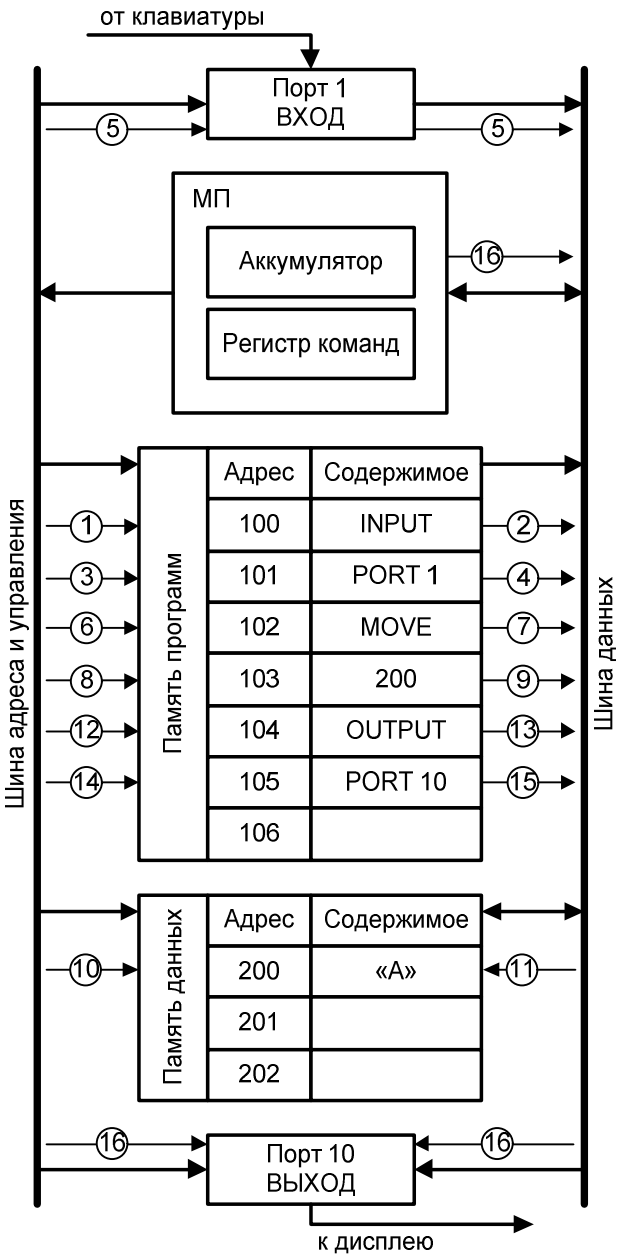


Рис. 8. Диаграмма выполнения процедуры ввода-сохранения-вывода

7. Код команды «Сохранить данные» подается на *ШД* и пересылается в МП, где помещается в регистр команд.

8. МП дешифрирует эту команду и определяет, что для нее нужен операнд. МП обращается к ячейке памяти *103* и приводит в активное состояние вход считывания микросхем памяти программ.

9. Из памяти программ на *ШД* пересылается код сообщения «Адрес 200». МП воспринимает этот операнд и помещает его в регистр команд. Полная команда «Сохранить данные в ячейке памяти 200» выбрана из памяти программ и декодирована.

10. Теперь начинается процесс выполнения команды. МП пересылает адрес *200* на *ША* и активизирует вход записи, относящийся к памяти данных.

11. МП направляет хранящуюся в аккумуляторе информацию в память данных. Код буквы «А» передается по *ШД* и записывается в ячейку *200* этой памяти. Выполнена вторая команда. Процесс запоминания не изменяет содержимое аккумулятора. В нем попрежнему находится код буквы «А».

12. МП обращается к ячейке памяти *104* для выбора очередной команды и переводит память программ в режим считывания.

13. Код команды вывода данных пересылается по *ШД* к МП, который помещает ее в регистр команд, дешифрирует и определяет, что нужен операнд.

14. МП выдает адрес *105* на *ША* и устанавливает память программ в режим считывания.

15. Из памяти программ по *ШД* к МП поступает код операнда «Порт 10», который далее помещается в регистр команд.

16. МП дешифрирует полную команду «Вывести данные в порт 10». С помощью *ША* и *ШУ*, связывающих его с устройством вывода, МП открывает порт 10, пересылает код буквы «А» по *ШД*. Буква «А» выводится через порт 10 на экран дисплея.

4. АРХИТЕКТУРА МИКРОПРОЦЕССОРОВ

При создании микропроцессоров используются все виды архитектуры, созданные за время их развития: регистровая, стековая, ориентированная на оперативную память.

Регистровая архитектура микропроцессора (архитектура типа «регистр – регистр») определяет наличие достаточно большого регистрового файла внутри БИС микропроцессора. Этот файл образует поле памяти с произвольной записью и выборкой информации. Микропроцессоры с регистровой архитектурой имеют высокую эффективность решения научно-технических задач, поскольку высокая скорость работы сверхбыстрой ОЗУ (СОЗУ) позволяет эффективно

использовать скоростные возможности арифметико-логического блока. Однако при переходе к решению задач управления эффективность таких микропроцессоров падает, так как при переключениях программ необходимо разгружать и загружать регистры СОЗУ.

Стековая архитектура микропроцессора дает возможность создать поле памяти с упорядоченной последовательностью записи и выборки информации. Эта архитектура эффективна для организации работы с подпрограммами, что необходимо для решения сложных задач управления, или при работе с языками высокого уровня. Хранение адресов возврата позволяет организовать в стеке эффективную обработку последовательностей вложенных подпрограмм. Однако стек на кристалле микропроцессора с малой информационной ёмкостью быстро переполняется, а стек большой ёмкости требует значительных ресурсов. Реализация стека в ОЗУ решает эти проблемы.

Архитектура микропроцессора, ориентированная на оперативную память (архитектура типа «память – память»), обеспечивает высокую скорость работы и большую информационную ёмкость рабочих регистров и стека при их организации в ОЗУ. Эта архитектура отнесена к типу «память – память», поскольку в МП с такой архитектурой все обрабатываемые числа после операции в микропроцессоре вновь возвращаются в память, а не хранятся в рабочих регистрах.

При оценке быстродействия ЭВМ необходимо учитывать физическую реализацию как элементов, так и связей между ними. Высокая скорость срабатывания логических элементов интегральных схем не всегда может обеспечить высокую скорость работы ЭВМ, поскольку большие значения индуктивно-емкостных параметров связей на печатных платах не позволяют передавать высокоскоростные сигналы без искажения. Применение БИС существенно уменьшило размеры ЭВМ, снизило паразитные параметры связей. Поэтому стало возможным физически отделить регистровый блок регистров и стек от арифметико-логического блока и обеспечить при этом их высокоскоростную совместную работу. При создании ЭВМ в одном кристалле регистровые СОЗУ и ОЗУ имеют практически одни и те же параметры. Повышение скорости работы ОЗУ позволяет удалить регистровый файл и стек из кристалла микропроцессора и использовать освободившиеся ресурсы для развития системы команд, средств прерывания, многоразрядной обработки. Организация рабочих регистров и стека в ОЗУ приводит к уменьшению скорости передачи информации, однако при этом повышается общая эффективность такого решения за счет большой информационной ёмкости полей регистровой и стековой памяти, а также развития систем команд и прерываний.

Архитектура микропроцессора, ориентированная на оперативную память, обеспечивает экономию площади кристалла мик-

ропроцессора. В этом случае на кристалле размещается только регистр-указатель начального файла регистров. Адресация остальных регистров осуществляется указанием в команде кода смещения. Доступ к рабочим регистрам в этом случае замедляется, поскольку приходится совершать сопряженное с затратами времени кольцевое «путешествие» из процессора во внекристальную память, где размещаются рабочие регистры. Однако контекстное переключение в микропроцессоре с такой архитектурой происходит быстро, поскольку при прерывании необходимо только изменить значение содержимого регистра-указателя рабочей области.

Другая отличительная особенность архитектуры микропроцессора, ориентированная на оперативную память – двухадресный формат команд. В этих микропроцессорах нет специального накапливающего регистра, выполняющего функции подразумеваемой ячейки результата для всех двухоперандных команд.

В отличие от микропроцессоров с архитектурой, ориентированной на оперативную память, в микропроцессорах с регистровой архитектурой рабочие области регистров размещаются в логических частях процессоров. Однако малая плотность логических схем по сравнению с плотностью схем памяти ограничивает возможность регистровой архитектуры. В свою очередь, микропроцессоры с архитектурой, ориентированной на память, обеспечивают быстрое подключение к рабочим областям, когда необходимо заменять контексты. Смена контекстов осуществляется изменением векторов трех регистров – счетчика команд, регистра состояния и указателя рабочей области. Достоинство этой архитектуры в отношении смены контекстов связано с выполнением только одной команды для передачи полного вектора контекста. Микропроцессоры с регистровой архитектурой требуют больших и довольно медленных последовательностей команд или дополнительных логических схем для передачи данных от каждого из регистров к памяти, организованной вне БИС микропроцессора. Использование возможностей быстрой смены контекстов и фактически неограниченной рабочей области в микропроцессорах с архитектурой, ориентированной на оперативную память, позволяет контроллерам легко находить применение в 16-разрядных системах. Особенно это относится к системам, работающим в реальном масштабе времени.

При разработке системы обработки данных, ориентированных на использование в системах управления, важное значение для определения характеристик системы, ее габаритов и стоимости имеет выбор архитектуры процессора. Поэтому в настоящее время наиболее распространенными архитектурами программируемых контроллеров, которые лежат в основе таких систем, являются архитектуры, ориентированные на память. Еще одной статьей

затрат, где видно различие между рассматриваемыми архитектурами, являются дополнительные логические схемы, требуемые для осуществления таких важных операций, как обработка многократных прерываний. Объем дополнительных логических схем возрастает по мере роста числа операций по смене контекстов. Но еще больше объем дополнительной логики возрастает для архитектур, ориентированных на регистры, за счет повышения их емкости. В результате существенно снижается быстродействие процессора.

К достоинствам архитектуры микропроцессора, ориентированной на оперативную память, относится возможность развития системы, позволяющая снизить время разработки программного обеспечения. Под развитием понимается способность систем внедрять в виде функциональных модулей программные, программно-аппаратурные и даже аппаратурные средства, которые можно использовать в системе по мере совершенствования аппаратурных средств и накопления опыта.

Распределенные системы управления часто требуют полуавтономных контроллеров, которые должны вписываться в определенные иерархические структуры. При этом архитектура микропроцессора, ориентированная на память, обеспечивает естественный и эффективный интерфейс между контроллерами, расположенными на одном иерархическом уровне, и процессами управления, расположенными на более высоком иерархическом уровне, а структура связей между контроллерами может быть обеспечена за счет развитых информационных магистралей.

4.1. **Типовая структура микропроцессора**

Под архитектурой микропроцессора понимаются логическая структура отдельных блоков МП, их взаимосвязь, система команд, способы адресации, состав, назначение, принципы взаимодействия технических средств и программного обеспечения.

Микропроцессор реализует программное управление вычислительным процессом. Необходимую для этого управляющую информацию он получает в виде машинных команд, хранимых в оперативной памяти. ***Команда*** – это управляющее слово, в закодированном виде обозначающее одну из операций над ее операндами. Кроме команд, в оперативном запоминающем устройстве (*ОЗУ*) хранятся данные, используемые операционным блоком при выполнении команд. В процессе решения задачи микропроцессор исполняет последовательности команд в соответствии с программой, представляющей полное описание алгоритма решения задачи.

В самом общем случае функциональную схему любого микропроцессора можно представить в виде композиции трех функциональных блоков: операционного блока (***ОБ***), блока управления (***БУ***)

и интерфейсного блока (***ИБ***). Кроме них в состав микропроцессора могут входить и некоторые другие блоки, участвующие в организации вычислительного процесса, например, блок прерывания, блок защиты памяти, блоки контроля, диагностики и другие. Типовая структурная схема микропроцессора приведена на рис. 9.

Операционный блок предназначен для выполнения некоторого функционально полного набора логических и арифметических операций. В его состав входят арифметико-логическое устройство *АЛУ* и блок регистров общего назначения. Основой *АЛУ* являются двоичный сумматор и набор логических схем. В *АЛУ* выполняются несколько простейших арифметических (сложение, вычитание) и поразрядных логических (И, ИЛИ, НЕ и др.) операций. Операции по обработке данных, для которых в операционном блоке отсутствуют аппаратные средства, выполняют программно с помощью процедур, реализуемых в виде последовательности простых операций операционного блока.

Кроме универсального *АЛУ*, операционный блок МП может содержать одно или несколько специализированных *АЛУ*. В качестве последних обычно используют блоки аппаратного умножения и деления, а также блоки для выполнения операций с плавающей точкой. Наличие специализированных *АЛУ* естественно увеличивает сложность СБИС МП, но за счет быстрого выполнения дополнительных операций, а также возможности параллельного выполнения некоторых операций на различных *АЛУ*, производительность МП повышается.

Важной составляющей операционного блока современных МП является блок внутренней памяти, реализованный в виде набора программно доступных регистров, называемых регистрами общего назначения (*РОН*). Время обращения к *РОН* меньше, чем к любым другим устройствам памяти, поэтому память на *РОН* называется сверхоперативной (*СОЗУ*). Число *РОН* в МП невелико (6–16), но при их использовании существенно ускоряется выполнение операций. При наличии блока *РОН* операнды команд могут размещаться в одной из двух запоминающих сред – в *ОЗУ* (основной оперативной памяти) или в *СОЗУ*. Использование *СОЗУ* позволяет исключить значительную часть обращений МП к основной памяти через общую системную шину. С одной стороны, это повышает производительность за счет более быстрого обращения к *СОЗУ*, с другой стороны, появляется возможность параллельно с работой МП использовать системную шину для обмена информацией между другими устройствами ВМ. Используя специальные команды, пользователь может записывать информацию в *РОН*, считывать ее из *РОН* и использовать эту информацию в различных арифметических и логических операциях.

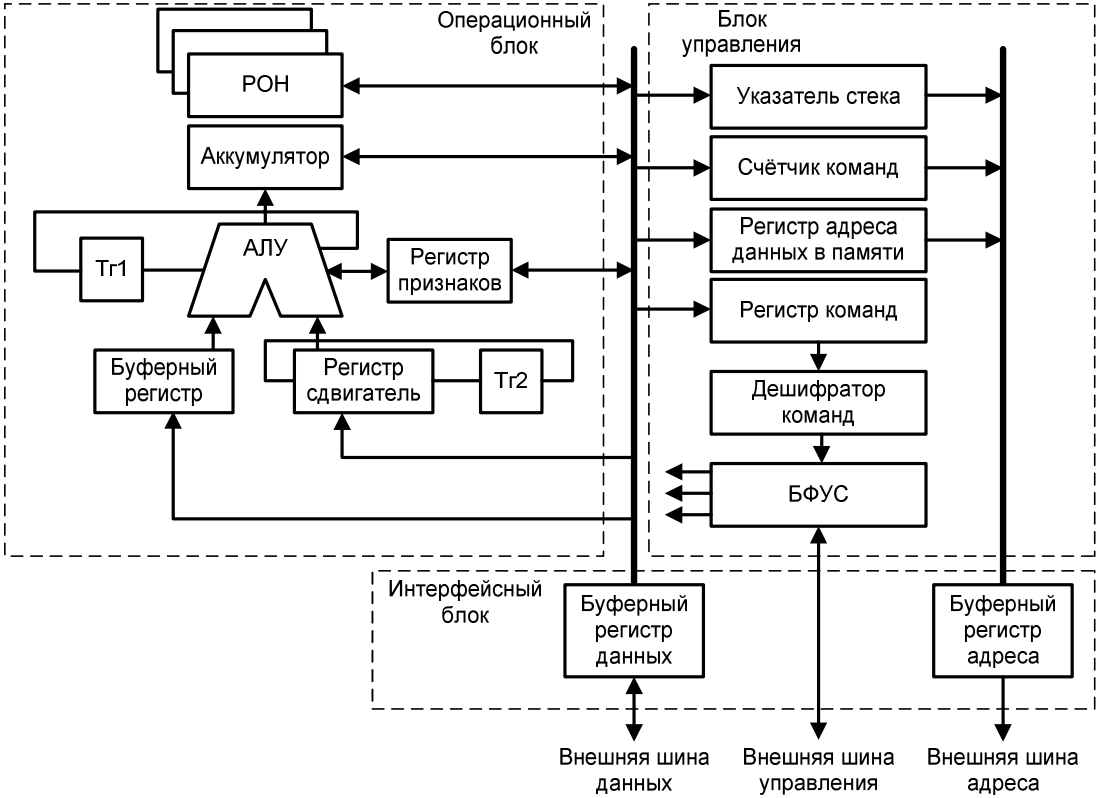


Рис. 9. Типовая структура микропроцессора

В большинстве ранних моделей МП один из общих регистров выделялся в качестве главного регистра. Наделение главного регистра, называемого аккумулятором или регистром результата, особыми функциями, позволяло реализовать *ОБ* в виде одноадресного устройства. В таком *ОБ* один из исходных операндов арифметических и логических операций обязательно должен размещаться в аккумуляторе и в него же помещается результат. Другой операнд названных операций может находиться в памяти или *РОН*. Входные данные поступают в аккумулятор с внутренней шины МП, при этом сам аккумулятор тоже может посылать свои данные на внутреннюю шину.

Функциональные возможности *ОБ*, содержащего аккумулятор (см. рис. 9), достаточно широки. С его помощью можно реализовывать различные операции (команды).

Например, содержимое любого *РОН* или ячейки памяти по внутренней шине данных может быть передано через аккумулятор в буферный регистр или напрямую в регистр-сдвигатель.

АЛУ обеспечивает выполнение арифметических и логических операций над содержимым регистра-сдвигателя и буферного регистра с записью результата в аккумулятор, откуда он может быть помещен в любой *РОН* или ячейку памяти. В частности, операция сдвига содержимого любого *РОН* выполняется последовательно путем передачи слова из *РОН* в сдвигающий регистр, сдвига этого слова и последующей записи преобразованного слова в тот же регистр *РОН*.

Необходимо отметить, что «ранние» модели однокристальных МП из-за несовершенства микроэлектронных технологий содержали относительно небольшое число транзисторов на кристалле. По этой причине такие МП имели малую разрядность *ОБ*, и для обработки операндов повышенной разрядности ее необходимо было выполнять программно путем последовательной обработки отдельных частей многоразрядных слов. Для обеспечения возможности обработки данных с разрядностью, превышающей разрядность *АЛУ* и регистров, в структуре *ОБ* (см. рис. 9) предусмотрены два дополнительных триггера *Тг1* и *Тг2*. С их помощью осуществляется запоминание сигналов арифметического переноса из *АЛУ* и выходного бита переноса регистра сдвига. Например, с помощью 8-разрядного МП сравнительно просто осуществляется арифметическая обработка 24-разрядных слов. Для этого выполняют 3 цикла обработки 8-разрядных частей этих слов.

При выполнении операций *АЛУ* формируются признаки результата (флаги). Типичными признаками являются: нулевой результат, наличие переноса, переполнение, четность, знак и некоторые другие.

Флаг нулевого результата ***ZF*** (*Zero Flag*) устанавливается в 1 при нулевом значении результата. При ненулевом результате ***ZF*** = 0.

Флаг переноса ***CF*** (*Carry Flag*) запоминает значение переноса (заёма) при сложении (вычитании) операндов. В некоторых МП флаг ***CF*** также используется для запоминания выдвигаемого бита при сдвиге операнда.

Флаг переполнения ***OF*** (*Overflow Flag*) фиксирует переполнение разрядной сетки результата при выполнении операций со знаковыми числами.

Переполнение происходит только тогда, когда коды слагаемых имеют одинаковые значения знаковых разрядов, а код суммы имеет другое значение знакового разряда. Для определения переполнения *OVR* используют логическую операцию Исключающее ИЛИ над значениями переносов в знаковый разряд C_{s-1} и из знакового разряда C_s ($OVR = C_{s-1} \oplus C_s$). В соответствии с логикой этой операции переполнение возникает, если перенос в знаковый разряд C_{s-1} не совпадает со значением переноса из знакового разряда C_s.

Флаг четности или паритета ***PF*** (*Parity Flag*) фиксирует наличие четного числа единичных разрядов в младшем байте результата операции. Флаг четности может быть использован, например, для контроля правильности передачи данных.

Флаг знака ***SF*** (*Sign Flag*) дублирует значение старшего бита результата. При выполнении операций с использованием дополнительных кодов флаг *SF* соответствует знаку числа.

Признаки результата операции в *АЛУ* (флаги) запоминаются в одноименных разрядах регистра признаков. В большинстве случаев они используются для программного управления последовательностями выполняемых команд при ветвлениях и циклах.

Блок управления в процессе выполнения программы координирует работу всех блоков МП и микропроцессорной системы в целом. С помощью этого блока формируются управляющие сигналы, необходимые для организации обмена информацией с внешними устройствами, и обеспечивается выборка команд программы из памяти. В целом блок управления выполняет следующие действия:

– считывает и запоминает текущую команду;

– управляет ее выполнением;

– формирует адрес следующей команды;

– управляет обменом информацией с внешними устройствами по системной шине.

Блок управления состоит из регистра команд (*РгК*), дешифратора команд (*ДшК*) и блока формирования управляющих сигналов (*БФУС*). В состав блока управления также включают программно доступные счетчик команд ***PC*** (*Program Counter*) и указатель сте-

ка ***SP*** (*Stack Pointer*). Большинство элементов этого блока, кроме счетчика ***PC*** и указателя стека ***SP***, являются программно недоступными.

Блок управления обеспечивает выполнение любой команды в виде последовательности трех фаз: выборка, декодирование и выполнение. При выборке осуществляется считывание очередной команды из памяти и пересылка ее в МП. Адрес считываемой команды определяется содержимым программного счетчика ***PC***. Любая команда всегда содержит всю необходимую информацию о выполняемой операции и об ее операндах. Для указания этой информации команды имеют определенную структуру, называемую форматом или структурой команды. Форматы команд у различных типов МП в деталях отличаются. Общим является то, что структура команды состоит из двух частей: кода операции и адресной части. Код операции однозначно определяет тип выполняемой операции. Адресная часть указывает на ячейки памяти, к которым надо обратиться, выполняя команду. В ней содержится информация об адресах операндов и результата. В зависимости от типа команда может состоять из одного или нескольких байт, при этом код операции всегда размещается в первом байте команды. Код операции текущей команды запоминается в *РгК* на время ее выполнения.

В фазе декодирования содержимое *РгК* с помощью *ДшК* преобразуется в управляющее слово. Схема синхронизации, используя это слово, вырабатывает совокупность сигналов, управляющих внутренними операциями МП и обменом информацией между МП и внешними устройствами.

После выборки и дешифрирования команды *ОБ* в декодированном виде получает информацию о том, какую операцию он должен выполнить, где расположены данные, куда следует направить результат операции и какая команда будет выполняться следующей. Далее следует фаза выполнения, в которой *БФУС* вырабатывает последовательности управляющих сигналов и передает их в *ОБ*.

Переход от языка машинных команд к выполнению элементарных операций в операционном блоке МП реализуется методом интерпретации. Интерпретация означает, что команда, помещенная в регистр команд МП, при исполнении разворачивается в последовательность элементарных операций, из которых складывается обработка данных. Процесс функционирования операционного блока МП осуществляется синхронно с тактовыми импульсами, поступающими от генератора, синхронизирующего работу всех блоков. Частота этих импульсов (тактовая частота) характеризует быстродействие МП. Элементарное действие, выполняемое в одном из узлов операционного блока в течение одного такта, называется ***микрооперацией***. Отдельные узлы *ОБ* за один такт могут

выполнять только одну простейшую операцию. Для регистра – это запись кода или выдача хранимого кода на выходы. Для счетчика – запись кода в счетчик, прибавление или вычитание единицы, выдача хранимого кода. Для мультиплексора – передача на выход кодового сигнала со входа, определяемого управляющим кодом. Для *АЛУ* – простейшими операциями являются: сложение, вычитание, сдвиг, реализация одной из поразрядных логических операций. В некоторые такты в различных узлах операционного блока (регистрах, мультиплексорах, счетчиках и др.) одновременно могут выполняться несколько микроопераций. Микрооперация или совокупность микроопераций, выполняемых одновременно в течение одного такта, соответствует ***микрокоманде***. Настройка операционного блока на выполнение требуемой микрокоманды осуществляется с помощью сигналов, поступающих на его управляющие входы. Набор этих сигналов (вектор сигналов управления) вырабатывается блоком формирования управляющих сигналов БФУС. Команда в общем случае реализуется как последовательность отдельных микрокоманд и обычно выполняется за несколько тактов. Последовательность микрокоманд, обеспечивающая выполнение команды, называется **микропрограммой команды**.

Интервалы времени, кратные машинному такту и связанные с выполнением микрооперации и команды, характеризуют специальными понятиями:

– **машинный цикл** – интервал времени, в течение которого происходит один внешний обмен данными между МП и внешним устройством (памятью или периферийным устройством);

– **командный цикл** – интервал времени, в течение которого выполняется команда. Командный цикл может состоять из одного или нескольких машинных циклов.

Алгоритм выполнения команды обычно формализуют и представляют блок-схемой алгоритма. При реализации алгоритма набор управляющих сигналов формируется с помощью управляющего автомата. Поскольку выполнение любой команды по тактам осуществляется последовательностью микрокоманд, то управляющий автомат называют **микропрограммным автоматом** (***МПА***). При построении управляющего автомата используют два основных способа управления исполнением команд программы – **схемное** или **аппаратное управление** и **микропрограммное управление**. Микропроцессоры, реализующие названные способы управления, называют соответственно микропроцессорами с **жесткой логикой** управления и микропроцессорами с **гибкой логикой** управления.

Алгоритм команды в микропроцессорах с *МПА* с жесткой логикой управления задан жестко соединениями схемы. Список опе-

раций (система команд) такого МП является неизменным, и после его изготовления какие-либо изменения в системе команд МП невозможны. *МПА* с жесткой логикой управления чаще всего строится на основе программируемой логической матрицы и представляет собой достаточно сложную схему. Его основным достоинством является высокое быстродействие. Устройства управления однокристальных микропроцессоров обычно реализуют в виде подобного *МПА* с жесткой логикой управления.

МПА с мягкой логикой управления выполняют по типовой структуре автомата с блоком памяти микропрограмм, имеющим линейно-адресную организацию. В блоке микропрограммного управления (*БМУ*) каждой выполняемой операции (команде) ставится в соответствие совокупность хранимых в памяти слов – микрокоманд, содержащих информацию о микрооперациях, подлежащих выполнению в течение одного машинного такта, и указание, какая должна быть выбрана из памяти следующая микрокоманда. Простейший *БМУ* (рис. 10) состоит из схемы формирования адресов микрокоманд и микропрограммной памяти (*МПП*) с выходным регистром микрокоманд. Основным назначением *БМУ* является преобразование кода команды в последовательность микрокоманд, точнее, в последовательность адресов микропрограммной памяти, в которой расположены микрокоманды, образующие микропрограмму данной команды. В микропроцессорах с мягкой логикой управления жестко заданным является не список команд, из которых обычно составляется программа решения задачи, а список элементарных однотактовых микроопераций. Разработка управляющего автомата *МПА* с мягкой логикой управления в значительной мере определяется разработкой микропрограмм команд.

Из заданного набора микроопераций пользователь путем микропрограммирования может создавать произвольные системы команд, ориентированные на эффективное решение определенных задач, или эмулировать требуемые системы команд. **Эмуляция** – это создание системы команд для процессора, имитирующего команды другого процессора. Эмуляция позволяет использовать ранее разработанное программное обеспечение на более совершенном процессоре.

До появления технологии СБИС *МПА* с мягкой логикой управления имели преимущественное распространение. В основном это было связано с тем, что при ограниченных возможностях технологии производства интегральных схем и средств автоматизации проектирования использование блоков памяти с регулярной структурой для хранения микропрограмм команд существенно облегчало внесение изменений в алгоритмы управления команд. Отметим, что

использование такого подхода внутри однокристального микропроцессора не является предпочтительным, так как приводит к снижению быстродействия. При необходимости изменения команд таких микропроцессоров модифицируют схемные реализации алгоритмов команд. При развитых средствах автоматизированного проектирования это не представляет труда. В зависимости от системы команд и сложности алгоритмов микропрограмм команд при построении *МПА* в современных микропроцессорах используются оба подхода.

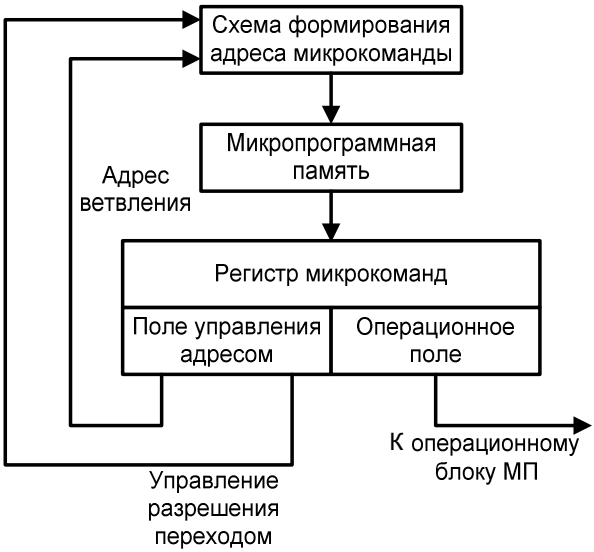


Рис. 10. Функциональная схема блока микропрограммного управления

На начальном этапе развития микропроцессоров возможности микроэлектронных технологий ограничивали сложность реализации структур МП. По этой причине основные блоки микропроцессоров в ряде случаев выполняли в виде отдельных СБИС. Микропроцессоры подобной реализации получили название многокристальных.

При разделении схем основных блоков процессора на секции, предназначенные для обработки нескольких разрядов данных или выполнения определенных управляющих операций, получается структура многокристального секционного МП. Отличительной особенностью секционных МП с микропрограммным управлением является возможность наращивания разрядности обрабатываемых данных или усложнения управляющего блока. МП данного типа реализуется из нескольких БИС, каждая из которых имеет структуру ***n***-разрядного операционного, управляющего или другого функ-

ционального блока микропроцессора. В большинстве случаев в виде секций реализуют операционный блок процессора. При последовательном включении ***n***-разрядных секций операционного блока, объединенных общей шиной микропрограммного управления, получают процессор требуемой разрядности.

Функциональная законченность БИС многокристального МП обеспечивает возможность автономной работы его блоков и упрощает конвейеризацию исполнения последовательностей команд программы. В ряде случаев благодаря этому можно обеспечить повышение производительности МП.

Кратко охарактеризуем назначение счетчика команд ***PC*** и указателя стека ***SP***. Счетчик команд ***PC*** предназначен для адресации команд программы. После выборки из памяти очередной команды в ***PC*** формируется адрес следующей по порядку команды. В командах условных и безусловных переходов, вызова подпрограмм и возврата из подпрограмм в ***PC*** непосредственно загружается адрес перехода. Введение счетчика команд в структуру МП позволило упростить формат адресной части команды, исключив из него поле, содержащее адрес следующей команды.

Стек – это память с линейно упорядоченными ячейками и специальным механизмом доступа, исключающим необходимость указания адреса при записи и чтении. В зависимости от используемого правила доступа, называемого **дисциплиной**, различают два типа организации стековой памяти: **очередь** и **стек**. Дисциплина определяется применительно к входным (записываемым) и выходным (читаемым) последовательностям слов. Очередь реализует дисциплину ***FIFO*** (*First-In-First-Out* – первый поступивший извлекается первым). Стек, в отличие от очереди, организован в соответствии с дисциплиной ***LIFO*** (*Last-In-First-Out* – последний поступивший извлекается первым), т.е. информация из стека выбирается в обратном, по отношению к записи, порядке.

Физически стековая память МП или просто стек представляет собой набор регистров (аппаратурный стек) или ячеек оперативной памяти, снабженный указателем стека ***SP***. Указатель ***SP*** всегда адресует «вершину стека», под которой понимается ячейка стека, доступная для чтения. По мере записи и считывания данных из стека содержимое ***SP*** меняется: при записи или загрузке в стек, например, при исполнении команд *PUSH*, значение ***SP*** уменьшается – стек растет в сторону младших адресов, а при чтении или выталкивании данных из стека, в частности, при исполнении команд *POP*, значение ***SP*** увеличивается. Указанное правило при обращении к стеку реализуется автоматически, и поэтому при операциях со стеком возможно безадресное задание операнда. В качестве

указателя стека обычно используют реверсивный счетчик. Стековая память является безадресной. Принцип работы стека и способ адресации его вершины иллюстрирует рис. 11.

Важнейшей характеристикой стека является его размер. МП может содержать относительно небольшой по размеру аппаратный стек (число внутренних регистров стековой памяти, как правило, не превышает 8–16) или не содержать такового совсем. Более распространена архитектура МП, использующая практически неограниченный внешний стек, моделируемый в основной памяти с произвольным доступом.

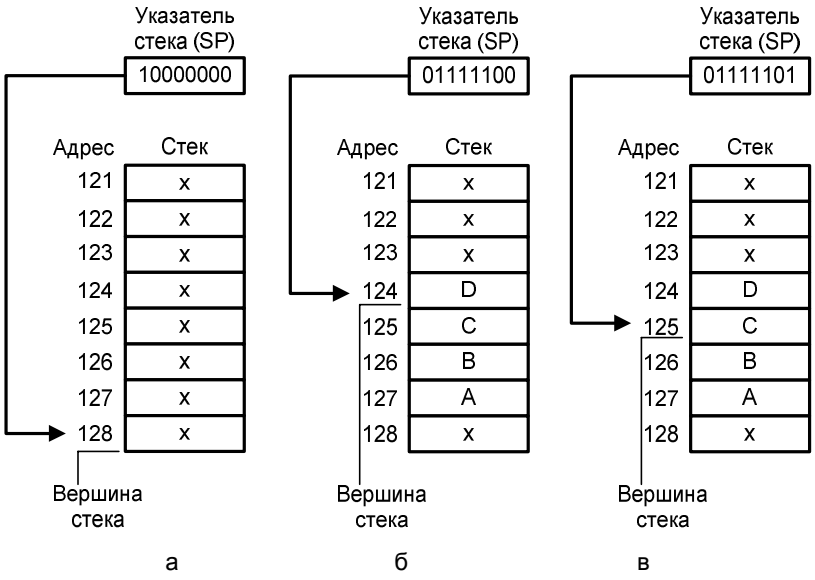


Рис. 11. Принцип работы стека и способ адресации его вершины с использованием указателя (SP):
а – стек после инициализации (содержит недействительные данные);
б – стек после записи в него командами *PUSH* данных A, B, C, D;
в – стек после извлечения из него элемента данных D (данные D стеку не принадлежат)

В МП стек применяется в качестве средства сохранения адресов возврата и состояния данных при работе с подпрограммами. Его использование приводит к существенным упрощениям при организации вложенных подпрограмм, когда одна программа вызывает другую, которая, в свою очередь, может вызвать третью и т.д. В таких случаях при каждом вызове адрес возврата текущей программы и другая необходимая информация (содержимое *РОН*) загружаются в стек. При возврате информация в обратном порядке

выбирается из стека. Заметим, что при организации стека, моделируемого в памяти с произвольным доступом, время обращения к элементам данных стека равно времени обращения к памяти. Однако стек, наряду с отмеченными особенностями его использования, эффективнее обычной памяти. Во-первых, используемые при обращении к стеку команды *PUSH* и *POP* короче стандартных команд обращения к памяти, так как в них один из операндов неявно адресуется через регистр ***SP***, и, во-вторых, инкремент или декремент указателя ***SP*** с образованием нового адреса производится автоматически. Из других применений стека можно выделить его использование для временного хранения данных, когда для них нет смысла выделять фиксированные места в памяти, для организации прерываний, для передачи и возврата параметров при вызовах процедур возврата из них. Кроме этого, стековая память является важнейшей компонентой процессоров со стековой архитектурой.

Интерфейсный блок МП предназначен для организации взаимодействия МП с памятью и устройствами ввода/вывода, расположенными на системной шине процессора, а также для обмена данными между операционным блоком и внутренними устройствами МП. Непосредственное подсоединение устройств ввода/вывода к МП осуществляется с помощью специальных схем сопряжения, которые называются интерфейсом ввода/вывода. В общем случае интерфейсный блок МП должен выполнять следующие функции:

– формировать выходные сигналы на шинах адреса, данных и управления в режиме вывода;

– формировать выходные сигналы на шинах адреса и управления и считывать (воспринимать) сигналы с шины данных в режиме ввода;

– синхронизировать процессы внутри МП и на системной шине;

– реализовывать стандартный для системной шины протокол обмена.

Протокол обмена информацией по системной шине, объединяющей сигналы шин данных, адреса и управления, определяет последовательности сигналов (временную диаграмму сигналов), обеспечивающих правильную передачу информации между устройствами микропроцессорной системы. Электрические спецификации сигналов на линиях системной шины определяют гарантированные уровни электрических напряжений, идентифицирующих логические состояния *0* и *1*, и их нагрузочную способность. Одной из особенностей компонентов микропроцессорных систем, обеспечивающей возможность их электрического сопряжения друг с другом и разнообразными периферийными устройствами, явля-

ется TTL-совместимость. Выбор такого стандарта для интерфейса определяется очень широким распространением разнообразных устройств на TTL-схемах. Выходные сигналы МП являются маломощными (их нагрузочная способность эквивалентна одному входу TTL-схем). Для согласования выходных сигналов МП с нагрузочными входами внешних устройств используют специальные усилители, в частности, шинные формирователи.

Взаимодействие по шине (регистровая пересылка между одним из регистров операционного блока МП и ячейкой основной памяти либо портом периферийного устройства) осуществляется за цикл шины. Длительность цикла шины может изменяться в зависимости от быстродействия внешних устройств. Для согласования по быстродействию взаимодействующих при обмене по шине устройств используется принцип квитирования. Суть его в том, что процессор, управляющий обменом, в каждом цикле ждет уведомления (квитанции) о том, что устройство на шине выполнило операции, связанные с обменом, т.е. выставило на шину данные при вводе либо восприняло данные с шины при выводе. Для уведомления используется сигнал «Готовность» (*READY*), передаваемый от внешнего устройства в процессор по одной из линий шины управления.

4.2. **Система команд**

Система команд является одной из основных архитектурных характеристик МП. Система команд определяет совокупность операций, реализуемых МП. В понятие система команд входят:

– список команд, их функциональное назначение;

– форматы команд и обрабатываемых данных;

– способы адресации данных.

Команды, реализуемые любым МП, можно подразделить на следующие функциональные группы:

– пересылки данных и ввода-вывода;

– арифметических и поразрядных логических операций;

– передачи управления.

Команды пересылок данных обеспечивают как внутренний обмен информацией между регистрами внутри МП, так и внешние обмены данными при их передаче в МП из памяти или устройства ввода и из МП в память или устройство вывода. Обозначения команд пересылок, используемые в языках Ассемблера, имеют вид: *MOVE* (переслать), *LOAD* (загрузить), *STORE* (запомнить), *EXCHANGE* (обменять). В командах этой группы обычно указывается направление передачи, источник и/или приемник данных. Так как большинство современных МП различают адресные пространства ввода/вывода и памяти, то для обращения к портам ввода/вывода,

выделенным в отдельное адресное пространство, используют специальные команды пересылок с обозначениями *INPUT* для ввода и *OUTPUT* для вывода. В группу команд пересылок часто включают команды загрузки в стек *PUSH* и извлечения *POP* из стека.

Команды арифметических и поразрядных логических операций. В большинстве случаев в число команд этой группы входят команды простейших арифметических операций: сложить (*ADD*), вычесть (*SUBTRACT – SUB*) и команды поразрядных логических операций И (*AND*), ИЛИ (*OR*), Исключающее ИЛИ (*EXCLUSIVE OR – XOR*). К командам арифметических операций часто относят команды сдвигов (арифметических и логических), а к командам логических операций – команды сравнения *COMPARE* (неразрушающего вычитания). Логические сдвиги отличаются от арифметических тем, что в них участвуют все разряды чисел, включая знаковые.

Команды сложных арифметических операций типа умножения и деления содержатся в системах команд не у всех МП. В «ранних» моделях МП таких команд нет, поэтому названные операции необходимо выполнять программным путем, что требует больших затрат времени и памяти.

В некоторых МП система команд ориентирована только на обработку двоичных чисел с фиксированной запятой. Операции с другими формами представления двоичных чисел чаще всего выполняются программно. Большинство МП также обеспечивают обработку данных, представленных в двоично-десятичном коде. Для коррекции результатов арифметических операций над такими данными в группу команд арифметических операций включают специальные команды десятичной коррекции типа *DECIMAL ADJUST*.

В число команд этой группы могут входить команды обработки чисел в формате с плавающей точкой, а также команды мультимедийной обработки (*SIMD*-команды).

Команды передачи управления используются для изменения последовательности выполнения команд при наличии программных ветвлений *JUMP*, обращении к подпрограммам *CALL* и выхода из них *RETURN*. В зависимости от результата выполнения текущей команды с помощью команд условной передачи управления, например, команды условного перехода при нулевом значении сигнала переноса *JUMP ON CARRY ZERO*, МП может выбрать одну из возможных ветвей продолжения программы. Обычно в системе команд имеется несколько команд условных переходов по прямому и инверсному значениям различных признаков результата.

Системы команд современных МП, наряду с традиционными командами пересылок, арифметических и логических операций, командами передачи управления, содержат в своем составе груп-

пы команд, значительно расширяющие функциональные возможности МП по обработке информации, управлению его работой, а также обеспечивающие реализацию многозадачного защищенного режима работы. В частности, команды МП Pentium можно разделить на следующие функциональные группы:

– команды операций над целыми числами. В их число включают команды пересылки данных и ввода-вывода, команды арифметических и поразрядных логических операций, в том числе команды сдвига, команды операций со строками символов, команды битовых операций;

– команды операций над числами с плавающей точкой;

– команды передачи управления;

– команды расширений ***MMX*** (*MultiMedia Extensions*) и ***SSE*** (*Streaming SIMD Extensions*), поддерживающие технологию ***SIMD*** (*Single Instruction – Multiple Data*) над целыми числами (*MMX*) и числами с плавающей точкой (*SSE*). Команды данной группы выполняют однотипные действия сразу над всеми числами в упакованных форматах;

– команды поддержки языков высокого уровня;

– системные команды поддержки функций ОС по управлению памятью, средствами защиты и переключению задач;

– команды управления МП.

В системы команд конкретных МП могут входить команды, не вписывающиеся в предложенную классификацию. Подобные команды не отражают общих принципов построения программ и рассматриваются как дополнительные.

4.3. **Структура команд**

Выше отмечено, что команда МП в общем случае содержит операционную и адресные части. Соглашение о распределении разрядов между названными частями команды и о способе кодирования информации определяет структуру команды или ее формат (рис. 12). В операционной части команды, состоящей из ***n – k*** двоичных разрядов, содержится код операции, обеспечивающий кодирование 2n–k операций и определяющий, какие при этом будут задействованы устройства в МП или вне его. В ***k***-разрядной адресной части команды содержится информация об адресах операндов, участвующих в выполнении операции. В общем случае адресная часть команды должна содержать четыре адресных поля ***A1***,…, ***A4***. Они предназначены для задания кодов адресов операндов (***A1***, ***A2***), адреса результата (***A3***) и адреса следующей команды (***A4***). В качестве адресов ***A1***,…, ***A3*** могут использоваться адреса ячеек оперативной памяти и адреса регистров внутренней памяти МП, а в качестве адреса ***A4*** – только адреса ячеек оперативной памяти.

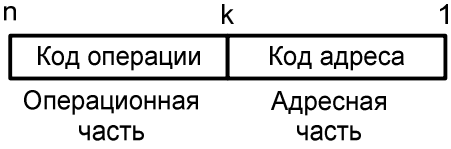


Рис. 12. Обобщенная структура команды

При использовании полного набора адресов в адресной части команды ее формат оказывается громоздким. Из-за ограниченного числа разрядов в машинном слове, разрядность которого обычно равна разрядности МП и ячеек оперативной памяти, актуальны способы повышения экономичности представления информации в команде. Прежде всего, было отмечено, что не для всех операций необходим полный набор адресов ***A1*** – ***A4***. В адресной части большинства команд указывается меньшее число адресов. В зависимости от указываемого числа адресов команды подразделяются на безадресные (нульадресные), одно-, двух-, трех- и четырехадресные. Практически во всех МП в адресном поле команды исключен адрес ***A4***. Это обусловлено тем, что большинство команд относятся к линейным участкам алгоритмов и такие команды могут быть размещены в ячейках оперативной памяти с последовательно возрастающими адресами. В этом случае для получения адреса следующей команды к адресу текущей достаточно прибавить единицу, что удобно реализовать путем инкремента счетчика команд. Такой способ адресации команд называют **естественным**. Соответственно МП, реализующие естественный способ адресации команд, называются МП с **естественным способом адресации команд**. При нарушении естественного порядка следования команд (ветвлениях, объединениях, циклах) используют специальные команды передачи управления, в которых содержится адрес перехода, но не используются адреса операндов. В отличие от МП с естественным способом адресации команд, МП, в адресном поле команд которых используется адрес перехода ***A4***, называют МП с **принудительным способом адресации команд**.

Использование адреса результата ***A3*** в адресном поле команды во многих случаях также оказывается избыточным. Это можно объяснить тем, что результат арифметических и логических операций над двумя операндами часто помещают на место одного из операндов (после вычисления результата операнды чаще всего больше не используются). При этом достигается не только сокращение разрядности команды, но и повышается производительность МП, поскольку исключаются лишние пересылки. Следует отметить, что при переходе к двухадресным командам в их адресное поле необходимо вводить дополнительные разряды, кодиру-

ющие назначение адресуемых операндов: кто из них является источником, а кто – приемником информации. По схеме двухадресного вычислителя реализованы МП x86 компании Intel, большинство процессоров фирмы Motorola и др.

В МП аккумуляторной архитектуры число адресов в адресной части команды уменьшено до одного. В них один из операндов, размещенный в аккумуляторе, неявно задается кодом команды и результат помещается в аккумулятор. По схеме одноадресного вычислителя реализованы МП 8080, MCS-51 компании Intel и ряд других.

Наконец, существует относительно небольшая группа безадресных команд, в которых осуществляется безадресное (неявное) задание операнда. К безадресным командам относятся команды управления процессором, например, пуска и останова. Безадресными также являются команды, реализующие операции со стеком. В них операнд, адресуемый указателем ***SP***, неявно задается кодом команды. Безадресные команды за счет исключения адресного поля имеют предельно сокращенный формат, но они не могут образовать функционально полную систему команд и применяются только вместе с адресными.

Число адресов, указываемых в команде, влияет на время решения задач, затраты памяти, сложность процессора и зависит от класса решаемых задач. В частности, для научно-технических расчетов, в которых большой объем занимают многошаговые вычисления, более эффективными оказываются одноадресные команды, а в некоторых случаях (при использовании стекового процессора) и безадресные команды. Для задач управления, в которых используется большое число пересылок и логических операций, эффективнее двухадресные команды. В современных микропроцессорах используют только безадресные, одноадресные и двухадресные команды. Трехадресные команды в системах команд МП присутствуют очень редко, а четырехадресные отсутствуют.

Рассмотренные способы указания адресов операндов иногда используют для классификации МП по числу адресуемых в команде операндов. В соответствии с этим классификационным признаком различают безадресные, одно-, двух- и трехадресные архитектуры.

4.4. **Способы адресации операндов и команд**

Рассматривая формат команды, мы предполагали, что адреса, содержащиеся в адресном поле команды, полностью идентифицируют адрес операнда в памяти. Такой способ адресации называют прямой адресацией. Прямая адресация неэкономична с точки зрения затрачиваемых разрядов для указания адреса (при увеличении размера памяти разрядность адреса также растет).

Имеются и другие недостатки прямой адресации. Например, в перемещаемых программах, где адреса операндов вычисляются в процессе выполнения самой программы, использование прямой адресации просто невозможно. Поэтому, наряду с прямой адресацией, разработаны и широко используются другие способы адресации. Им соответствуют различные механизмы формирования исполнительных адресов операндов в памяти. Для указания конкретного способа вычисления исполнительного адреса адресное поле команды дополняется специальным полем признака адресации.

Основными способами адресации являются **прямая**, **непосредственная**, **неявная**, **косвенная** и **относительная** адресации. Встречаются и другие способы, представляющие собой разновидности и комбинации перечисленных способов. При описании способов адресации будем использовать следующие обозначения. Адрес, указываемый в команде, обозначим A_K, а адрес физической ячейки памяти, к которой происходит обращение, назовем исполнительным адресом $A_И$.

Адресация данных

Прямая адресация – адрес операнда содержится в коде команды и обычно следует за кодом операции (рис. 13). Прямая адресация используется при работе с простыми переменными и константами, местоположение которых в памяти не меняется в процессе выполнения задачи. При прямой адресации

$$\mathrm{A_К} = \mathrm{A_И}.$$

Прямая адресация операндов, размещенных в регистрах МП, имеет специальное название **прямая регистровая** адресация или просто **регистровая** адресация. При использовании регистровой адресации в адресном поле команды указывается адрес (код) регистра. Команды, содержащие только регистровые операнды, являются наиболее компактными. В связи с тем, что все операции с регистровыми операндами реализуются операционным блоком процессора без обращения к основной оперативной памяти, команды с регистровыми операндами выполняются быстрее других типов команд.

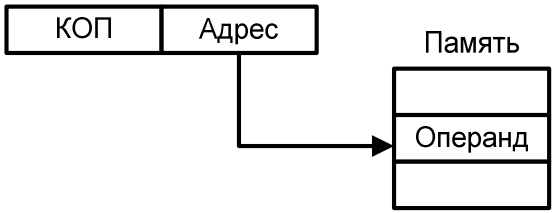


Рис. 13. Прямая адресация

Непосредственная адресация позволяет задавать фиксированные значения операнда (Оп) непосредственно в адресной части команды (рис. 14). Собственно адресация при этом отсутствует, т.е.

$$\text{Оп} = \text{А}_\text{К},$$

и при обращении к операнду нет необходимости обращаться к памяти. Непосредственная адресация удобна при работе с константами. Естественно, что непосредственный операнд может быть задан только как операнд-источник и его формат не может превышать разрядность операционного блока МП. Недостатком непосредственной адресации является необходимость расширения формата команд за счет указания самого операнда в адресном поле команды.

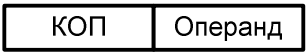


Рис. 14. Непосредственная адресация

Неявная адресация – способ адресации, при котором в команде не содержатся явные указания об адресе операнда, но этот адрес подразумевается. Фактически неявная адресация позволяет задавать адрес операнда по коду операции без указания этого адреса в адресной части команды. Неявно адресуемыми операндами могут быть аккумулятор, индексный и базовый регистры, указатель стека, отдельные биты регистра признаков и некоторые другие.

Косвенная адресация – эффективный и важный способ адресации, при котором адрес, указываемый в команде, является указателем ячейки, содержащей исполнительный адрес операнда в памяти. Фактически при косвенной адресации в команде указывается адрес адреса. Для обозначения косвенной адресации используется запись вида

$$\text{А}_\text{И} = (\text{А}_\text{К}).$$

Частным случаем косвенной адресации является **регистровая косвенная адресация**, при которой адрес, указываемый в команде, определяет общий регистр процессора с размещенным в нем адресом операнда в памяти. Способ обращения к операнду с использованием косвенной адресации показан на рис. 15. С точки зрения затрачиваемых разрядов для представления адреса, регистровая косвенная адресация оказывается много эффективнее прямой адресации, поскольку при ее использовании в адресном поле команды указывается только адрес общего регистра, а он много короче полного адреса операнда в памяти. Неслучайно кос-

венная адресация широко применяется в ВМ с коротким машинным словом. Однако регистровая косвенная адресация требует предварительной загрузки регистра косвенным адресом памяти и на это расходуется дополнительное время.

Косвенную адресацию удобно использовать при обработке списков и массивов данных, размещенных в памяти, а также при решении задач, когда, оставляя неизменным адрес в команде, можно изменять содержимое ячейки с этим адресом.

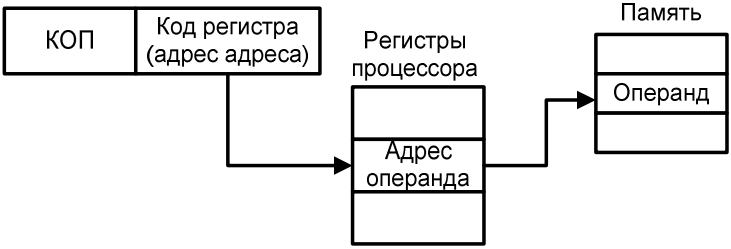


Рис. 15. Косвенная адресация

При использовании косвенной адресации заметно упрощается построение циклических программ. Способность выполнять циклы является одним из фундаментальных свойств ВМ. Благодаря циклам можно пользоваться командами по несколько раз, не дублируя их в памяти. В качестве примера организации цикла рассмотрим процедуру вычисления суммы элементов массива, расположенных в памяти в порядке возрастания их номеров (адресов).

$$y = \sum_{i=1}^{i=n} x_i.$$

Нетрудно заметить, что алгоритм вычисления суммы может быть представлен в виде циклической процедуры, повторяемой ***n*** раз. В каждом цикле, наряду с выполнением операции суммирования содержимого какого-либо регистра, например аккумулятора, с последовательно выбираемыми из памяти элементами массива, необходимо осуществлять модификацию адреса для выбора следующего элемента массива и контролировать окончание цикла. Блок-схема алгоритма циклической обработки приведена на рис. 16.

В соответствии с представленным алгоритмом в циклических программах выделяют три группы команд: инициализации параметров цикла, рабочей части цикла, контроля окончания цикла и модификации его параметров. Модификацию адресов операндов при циклической обработке удобно выполнять с помощью команд

инкрементирования или декрементирования содержимого регистра с адресом текущего элемента массива. При инкрементировании к значению регистра прибавляется число, определяемое размером (количеством байт) операнда, а при декрементировании – из регистра вычитается соответствующее число. При наличии средств автоматической модификации адреса косвенная адресация называется **автоинкрементной** или **автодекрементной**. При автоинкрементной адресации содержимое адресуемого регистра в каждом цикле сначала используется как адрес операнда, а затем получает приращение, равное числу байт в элементе массива. Проверку окончания цикла чаще всего осуществляют путем анализа содержимого счетчика циклов на соответствие заданному значению.

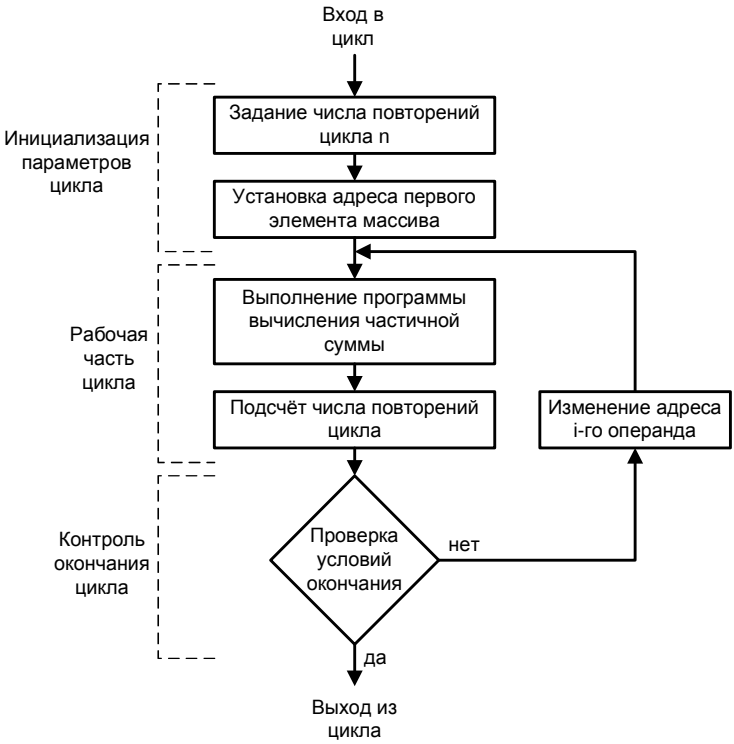


Рис. 16. Алгоритм циклической обработки

Развитием и модификацией метода косвенной адресации является **относительная адресация** или **базирование**. Это обобщенное название ряда методов адресации, обеспечивающих вычисление исполнительного адреса операнда в памяти в виде суммы

базового значения адреса и «смещения» *DISP*, указываемого в команде (рис. 17). Поскольку вычисление исполнительного адреса $A_И$ связано с потерей времени, часто для определения адреса $A_И$ операцию суммирования заменяют операцией конкатенации (приписывания разрядов). В этом случае базовый адрес содержит старшие, а смещение – младшие разряды исполнительного адреса. При использовании конкатенации, в отличие от суммирования, базовый адрес может задавать не любую ячейку памяти, а только ячейки, адреса которых содержат нули в младших разрядах.

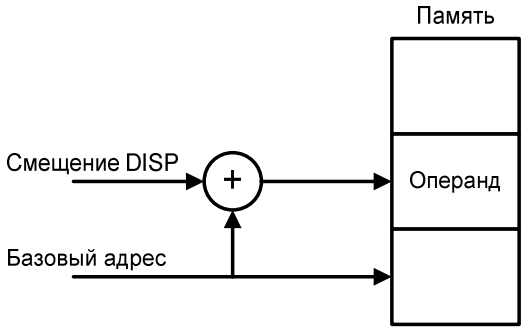


Рис. 17. Формирование исполнительного адреса при относительной адресации

Базирование (относительная адресация) широко применяется для адресации памяти, представленной в виде блоков фиксированного или произвольного размера. Блоки фиксированного размера называют страницами, а произвольного – сегментами. Соответственно различают память со **страничной организацией** и **сегментированную память**. Полная информация, необходимая для определения физического адреса произвольной ячейки памяти с подобной организацией, содержится в указателе адреса, который включает в себя идентификатор базового адреса блока и смещение внутри блока. Для определения базового адреса блока (сегмента или страницы) используют различные способы идентификации. Чаще всего базовые адреса блоков хранятся в специальных таблицах (сегментных или страничных), и идентификатор в указателе адреса служит индексом (номером) строки такой таблицы. Разрядность базового адреса в общем случае определяет максимальное число адресуемых блоков памяти, а число бит в смещении задает максимальный размер блока. Исполнительный (физический) адрес операнда образуется в результате суммирования базового адреса блока и смещении внутри блока.

Важной особенностью базирования (относительной адресации) является то, что при изменении базовых адресов блоков со-

держимое блоков не меняется и блоки можно свободно перемещать в пределах всего адресного пространства памяти. Благодаря этому свойству, базирование обеспечивает очень важную функцию операционных систем – так называемую перемещаемость программ. Перемещаемость программы предполагает неизменяемость адресных ссылок в программе при ее перемещении внутри доступного процессору пространства памяти. Базовые адреса исполняемых программ определяются операционной системой непосредственно при загрузке программы в *ОП*.

Собственно название относительная адресация закрепилось за способом адресации команд программной памяти, при котором базовый адрес размещается в счетчике команд ***PC***, а смещение *DISP*, указываемое в команде, определяет адрес перехода относительно текущего значения счетчика команд.

$$\mathrm{A_И} = (\mathrm{PC}) + DISP.$$

При использовании сравнительно коротких кодов смещения *DISP* относительная адресация операндов программной памяти позволяет при меньшем формате команды обеспечить доступ к требуемой ячейке памяти в ограниченном диапазоне адресов. В большинстве случаев относительную адресацию используют в командах условной и безусловной передачи управления для реализации коротких переходов.

Относительную адресацию также часто используют для адресации специальных структур данных, таких, как массивы однотипных переменных, элементы записей и других. Базовые адреса операндов при использовании относительной адресации обычно размещаются в регистрах процессора (специальных или общего назначения). Так как одна из составляющих исполнительного адреса находится в адресуемом в команде регистре, относительную адресацию данных иногда называют модифицированным способом косвенной адресации или косвенной адресацией со смещением. В зависимости от способа использования адресуемого в команде регистра различают два вида косвенной адресации со смещением – **базовую адресацию** и **индексную адресацию**. Регистры, адресуемые в команде, которые соответствуют этим способам адресации, называют базовыми ***B*** и индексными ***I***.

Индексная адресация обычно применяется для обработки упорядоченных массивов значений переменных, каждое из которых определяется собственным номером. При индексной адресации базовый адрес массива задается смещением *DISP*, указываемым в команде, а значение индекса (номер элемента массива) определяется содержимым индексного регистра. Исполнительный

адрес при индексной адресации определяется путем суммирования смещения с содержимым индексного регистра.

$$\text{А}_\text{И} = (I) + DISP.$$

Индексная адресация удобна, если необходимо записать или считать список данных из последовательных ячеек памяти не подряд, а с некоторым шагом, указанным в индексе, например, когда элементы матрицы, записанные по строкам, необходимо прочитать по столбцам. Данный способ адресации особенно эффективен, если в процессоре используется механизм автоматического приращения или уменьшения содержимого индексного регистра при каждом обращении к нему. В целом индексная адресация является развитием метода косвенной адресации.

Для доступа к структурам данных переменной длины применяют базовую адресацию. Базовой адресацией называется способ адресации, при котором базовый адрес, определяющий начало набора элементов, хранится в базовом регистре, а смещение в команде определяет расстояние до определенного элемента.

$$\text{А}_\text{И} = (\text{В}) + DISP.$$

Базовую адресацию удобно использовать для адресации элементов записи или структуры, под которыми понимают набор именованных элементов данных, возможно различных типов. Примером записи является следующий набор элементов данных:

Ф И О / Год поступления / Курс / Группа и т.д.

Запись бывает **простой** и **указываемой**. В простой записи положение любого элемента зафиксировано, и к нему можно обратиться, используя прямую адресацию. По существу простая запись совпадает с простой переменной. В указываемой записи положение элемента записи зависит от указателя (содержимого адресуемого в команде базового регистра).

Режимы адресации операнда для указываемой записи и для поиска элемента массива очень похожи. В обоих случаях используется режим косвенной адресации со смещением. Однако составляющие исполнительного адреса для указываемой записи и массива трактуются по-разному. В случае массива элементов смещение в команде соответствует началу массива, а значение адресуемого регистра – расстоянию в массиве. При указываемой записи содержимое адресуемого регистра соответствует началу записи, а смещение в команде – расстоянию в записи.

Для адресации элемента в указываемом массиве, т.е. массиве, адресуемом указателем, используют **базово-индексную адресацию**. При этом способе адресации базовый адрес массива задается

указателем базы (базовым регистром), а номер элемента массива определяется значением индексного регистра. Базово-индексная адресация удобна при работе со сложными структурами данных.

Базово-индексную адресацию со смещением применяют для адресации элементов в указываемом массиве записей, т.е. в массиве, каждый элемент которого является записью. Базовый адрес массива задается указателем базы, номер записи (элемента массива) определяется значением индексного регистра, а смещение в команде указывает расстояние в записи.

Адресация команд

При адресации команд программы также используют основные способы адресации, но в силу специфики их назначения они имеют специальные названия: **абсолютная** и **относительная адресация**.

Абсолютная адресация – это способ адресации команд, при котором в адресном поле команды указывается адрес следующей команды. При выполнении команды с абсолютной адресацией осуществляется загрузка счетчика команд процессора непосредственными данными. **Относительная** адресация команд рассмотрена выше.

Основные способы адресации, представленные в данном разделе, в тех или иных сочетаниях используются практически во всех реализуемых системах команд, расширяя или сокращая список команд реального МП. Тип адресации указывается либо неявно кодом операции, либо в явной форме в адресном поле команды.

Завершая рассмотрение способов адресации операндов в памяти микропроцессорных систем подчеркнем, что выбор режима адресации определяется конкретной задачей и во многих случаях очевиден. Однако нередки ситуации, когда для обращения к одним и тем же элементам данных допускается использование различных режимов адресации. В конечном итоге только пользователь осуществляет выбор конкретного режима адресации, и в этом ему предоставлены большие возможности.

5. ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРОВ С ФИКСИРОВАННЫМИ РАЗРЯДНОСТЬЮ И СПИСКОМ КОМАНД

5.1. Структура микропроцессора КР580ВМ80А

Особенности микропроцессоров этого класса рассмотрим на примере однокристального 8-разрядного МП КР580ВМ80А с фиксированной системой команд без возможности аппаратного наращивания разрядности.

Однокристальный микропроцессор КР580ВМ80А предназначен для параллельной обработки 8-разрядной цифровой информации. По назначению относится к классу универсальных МП и применяется в различных областях техники – от одноплатных контроллеров технологических процессов до персональных ЭВМ средней производительности. Кристалл МП выполнен по n-МОП технологии и содержит 4800 транзисторов. Быстродействие МП достигает 500 тыс.оп/с простых операций типа «регистр – регистр» при длительности цикла 250 нс. Конструктивно МП выполнен в пластмассовом корпусе (буква Р в аббревиатуре названия МП) с 40 выводами.

Основные технические характеристики процессора:

1) потребляемая мощность – 1,25 Вт;

2) напряжение питания –5 В, +5 В, +12 В;

3) допустимое отклонение напряжения питания ± 5%;

4) нагрузочная способность каждого вывода БИС – один вход элемента ТТЛ;

5) уровень лог. «1» 2,4 – 5 В; лог. «0» 0 – 0,4 В;

6) температурный диапазон –10…+70°С;

7) время спада и нарастания входных напряжений на выводах БИС – 30 нс.

Для расширения функциональных возможностей разработаны микросхемы поддержки, которые составляют микропроцессорный комплект К580, состоящий из 18 БИС (табл. 1).

Таблица 1

Состав микропроцессорного комплекта К580

Наименование БИС	Назначение	Краткое описание
КР580ВМ80А	Центральный микропроцессор	Параллельный прием, выдача и обработка 8-разрядной информации
КР580ВК28	Системный контроллер и буферный регистр данных	Хранение и дешифрация байта состояния МП, формирование шины управления системы
КР580ВК38		
КР580ВВ55А	Программируемый параллельный интерфейс	Сопряжение ШД системы для осуществления ввода-вывода параллельной информации
КР580ВВ51А	Программируемый последовательный интерфейс	Приемно-передающее устройство для обмена информацией: интерфейс: а) с МП – параллельным кодом; б) с ВУ – последовательным кодом
КР580ВН59	Программируемый контроллер прерывания	Все необходимые операции по обслуживанию до 8 запросов на прерывания от ВУ с возможностью расширения

Окончание табл. 1

Наименование БИС	Назначение	Краткое описание
КР580ВИ53	Программируемый таймер	Формирование программно-управляемых временных задержек для синхронизации управляемых объектов
КР580ВТ57	Программируемый 4-канальный контроллер прямого доступа к памяти	Высокоскоростной обмен информацией между памятью и ВУ по четырем каналам
КР580ВГ75	Программируемый контроллер видеотерминала	Вывод информации из памяти системы на экран растровых сканирующих дисплеев
КР580ГФ24	Генератор тактовых импульсов	Генерирование двух последовательностей тактовых импульсов, необходимых для работы МП
КР580ВВ79	Программируемый интерфейс клавиатуры и дисплея	Контроллер ввода-вывода для клавиатуры и цифрового дисплея
КР580ВА86	Шинный формирователь	Двунаправленный 8-разрядный шинный формирователь с высокой нагрузочной способностью
КР580ВА87		То же с инвертирующим выходом
КР580ИР82	Буферный регистр	Восьмиразрядный буферный регистр
К580ИР83		То же с инвертирующим выходом
К580ВК91А	Интерфейс микропроцессор – канала общего пользования	Сопряжение микропроцессоров и однокристальных микроЭВМ с линией коллективного пользования информационно-измерительной системы
КР580ВА93	Приёмо-передатчик МП – канал общего пользования	Программируемый приемопередатчик, предназначенный для использования в устройствах вычислительной техники и измерительной аппаратуре с цифровой обработкой информации
К580ВГ18	Контроллер шин	Реализует арбитраж, управление временной диаграммой и подключение управляющих сигналов к линиям магистрали

МП БИС имеет однонаправленную 16-разрядную адресную магистраль (*МА*), обеспечивающую адресацию к любой из 2^{16} 8-разрядной ячейке памяти или внешнего устройства (ВУ); двунаправленную 8-разрядную магистраль данных (*МД*) и 12 сигналов управления (шесть входных и шесть выходных).

Условное обозначение МП БИС К580ВМ80А приведено на рис. 18.

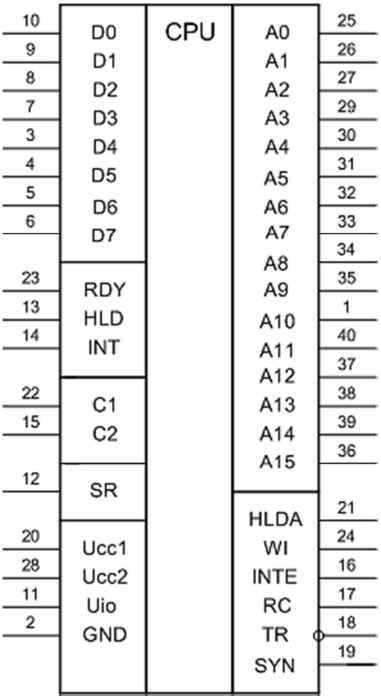


Рис. 18. Условное обозначение МП БИС КР580ВМ80А

Функциональное назначение внешних выводов МП БИС КР580ВМ80А:

A0 – A15 – выводы шины адреса.

D0 – D7 – выводы шины данных.

SYN (*СИНХР*, *SYNC*) – на этом выходе МП БИС формируется сигнал СИНХРОНИЗАЦИЯ вначале каждого машинного цикла.

RC (*П*, *DBIN*) – сигнал ПРИЕМ на этом выходе указывает на готовность МП БИС к приему данных.

WI (*ОЖД*, *WAIT*) – сигнал ОЖИДАНИЕ на этом выходе указывает, что МП находится в состоянии ожидания.

TR (*ЗП*, *WR*) – на этом выходе МП БИС сигнал ЗАПИСЬ указывает, что данные выданы МП БИС и установлены на *МД* (магистраль данных) и могут быть записаны во ВУ.

HLDA (*П.ЗХ*, *Р.ЗХ*) – на этом выходе МП БИС сигнал ПОДТВЕРЖДЕНИЕ ЗАХВАТА появляется в ответ на сигнал *З.ЗХ* (ЗАПРОС ЗАХВАТА) и указывает, что *МД* и *МА* находятся в состоянии высокого омического сопротивления.

INTE (*Р.ПР*) – на этом выходе сигнал РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ указывает на состояние внутреннего триггера разрешения прерывания МП БИС. Состояние триггера может быть установлено программно с помощью команд *EI*, *DI*. При уровне «0» на выходе *INTE* прием запросов прерывания МП БИС невозможен.

RDY (*Г*, *READY*) – сигнал ГОТОВНОСТЬ на этом входе информирует о готовности ВУ к обмену информацией с МП БИС. При уровне «0» МП БИС будет находиться в состоянии ОЖИДАНИЕ.

HLD (*З.ЗХ*, *HOLD*) – вход, используемый для подачи сигнала ЗАПРОС ЗАХВАТА на переход МП БИС в состояние ЗАХВАТ, в котором *МА* и *МД* переходят в третье состояние. Обычно это состояние используется для организации обмена информацией по каналу прямого доступа к памяти.

INT (*З.ПР*) – вход, используемый для подачи сигнала ЗАПРОС ПРЕРЫВАНИЯ. Сигнал поступает от внешнего источника на прерывание выполнения основной программы и переход на выполнение подпрограмм обслуживания прерывания. Сигнал запроса прерывания не воспринимается МП БИС при работе его в режимах ЗАХВАТ, ОЖИДАНИЕ или нулевом состоянии внутреннего триггера разрешения прерывания.

SR (*RESET*, *R*) – вход, по которому поступает сигнал на начальную установку МП БИС, при этом обнуляются его программный счетчик, внутренние триггеры, формирующие сигналы РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ и ПОДТВЕРЖДЕНИЕ ЗАХВАТА.

C1, *C2* (*CLK1*, *CLK2*, *Ф1*, *Ф2*) – входы для подачи тактовых сигналов. Эти сигналы являются не пересекающимися во времени сигналами, определяющими тактовую частоту работы МП БИС.

Структурная схема МП КР580ВМ80А (рис. 19) состоит из двух частей: операционной (*ОП*) и управляющей (*УП*). Обе части расположены на одном кристалле. Управляющая часть содержит недоступную для пользователя управляющую память, в которую в процессе изготовления БИС записаны операции, определяющие состав команд МП.

Структурная схема МП содержит следующие функциональные блоки: блок *АЛУ*, блок регистров *РОН* со схемой выборки регистра и выходным мультиплексором, блок синхронизации и управления (*БСУ*), буферы адресов (*БА*) и данных (*БД*).

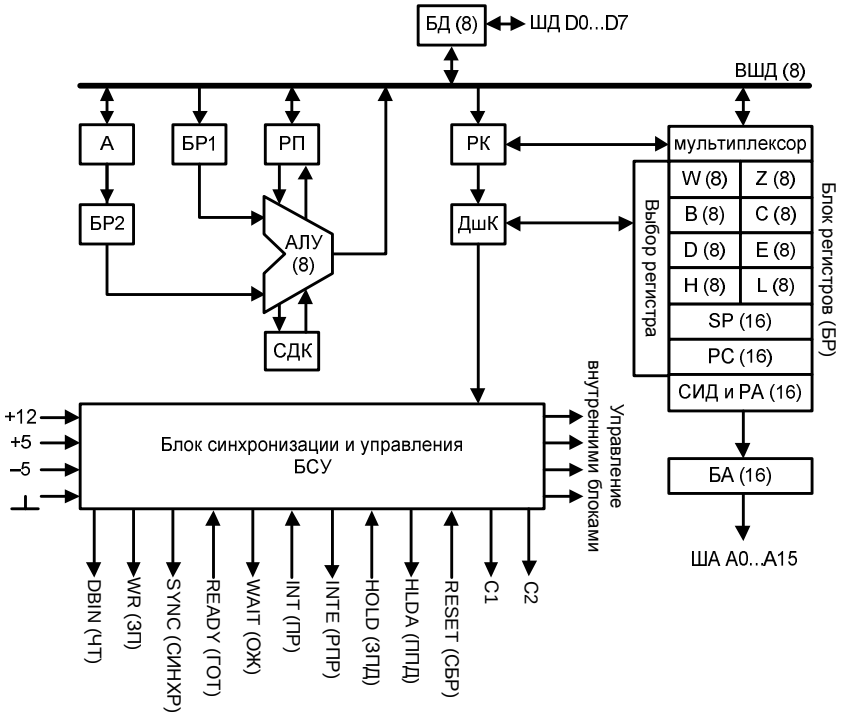


Рис. 19. Структурная схема микропроцессора КР580ВМ80А

В состав блока *АЛУ* входят 8-разядное *АЛУ*, регистр результата – аккумулятор (***А***), 8-разрядные буферные регистры (***БР1*** и ***БР2***), регистр признаков (флагов) ***РП*** и схема двоично-десятичной коррекции (***СДК***). Вход регистра ***БР1*** соединен с внутренней магистралью МП, а вход регистра-защелки ***БР2*** – с аккумулятором ***А***, выполняющим функции регистра-накопителя.

АЛУ МП выполняет арифметические, поразрядные логические операции, а также операции циклического сдвига над 8-разрядными двоичными числами. Базовой операцией АЛУ является операция сложения двоичных чисел. Все арифметические, логические и сдвиговые операции выполняются при участии аккумулятора. Результат операции размещается в аккумуляторе. Обмен информацией МП с ВУ возможен только через аккумулятор. Наряду с операциями над 8-разрядными двоичными числами, МП допускает выполнение арифметической операции сложения над операндами в формате двоично-десятичных чисел. В этом формате байт содержит две десятичные цифры, представленные двоичным кодом с весами разря-

дов *8421*. При выполнении сложения чисел в двоично-десятичном коде на двоичном сумматоре в общем случае получается результат, не соответствующий двоично-десятичному представлению. Коррекция результата осуществляется с помощью схемы *СДК*.

Признаки результата арифметических и поразрядных логических операций фиксируются в регистре признаков, содержащем флаги нуля ***Z***, знака ***S***, переноса ***C***, паритета ***P*** и вспомогательного (межтетрадного) переноса ***AC***. Формат регистра признаков МП КР580ВМ80А приведён на рис. 20. Пять битов признаков устанавливаются в зависимости от результатов выполнения операций следующим образом:

Бит знака (***S***) – устанавливается в соответствии с 7-м битом (***D7***) результата.

Бит нуля (***Z***) – устанавливается в «1», если результат равен 0; в противном случае сбрасывается в «0».

Бит четности (***P***) – устанавливается в «1», если число единиц в результате четно; в противном случае сбрасывается в «0».

Бит переноса (***C***) – устанавливается в «1», если в результате выполнения операции сложения возникает перенос из старшего разряда или при выполнении операции вычитания перенос не возникает (т.е. происходит заём); в противном случае обнуляется.

Бит дополнительного переноса (***AC***) – устанавливается в «1» при появлении переноса (заёма) из 3-го (***D3***) в 4-й (***D4***) бит результата при выполнении операций сложения (вычитания).

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	0	AC	0	P	1	C

Рис. 20. Формат регистра признаков МП КР580ВМ80А

Установку признаков рассмотрим на примере арифметических операций (сложения и вычитания) с 8-разрядными операндами (рис. 21).

Арифметическое-логическое устройство реализует простейшие арифметические и логические операции (сложение, вычитание, сдвиги, сравнение, логическое умножение и т.п.). Все более сложные операции (умножение, деление, вычисление элементарных функций и др.) выполняются по специальным подпрограммам.

Блок регистров предназначен для приема, хранения и выдачи различной информации, используемой в выполнении команд. В состав этого блока входят шесть 16-битовых регистров: три пары 8-разрядных регистров общего назначения, счетчик команд ***PC***, указатель стека ***SP*** и регистр временного хранения ***WZ***.

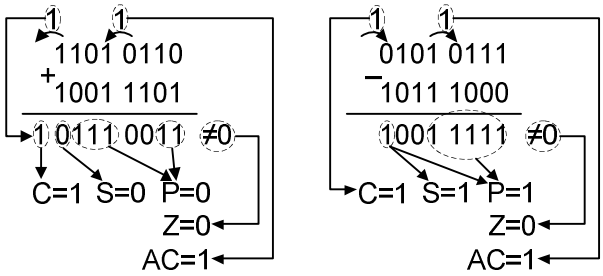


Рис. 21. Установка признаков при выполнении арифметических операций

Для выполнения операций инкремента/декремента содержимого регистров блок регистров дополнен схемой инкремента/декремента ***СИД***. Шесть регистров общего назначения ***B***, ***C***, ***D***, ***E***, ***H***, ***L,*** наряду с их непосредственным использованием в 8-разрядных операциях, могут объединяться в регистровые пары ***BC***, ***DE***, ***HL***. В командах регистровые пары обозначаются по имени старшего регистра в паре ***B***, ***D***, ***H***. Они могут хранить 16-битные операнды или использоваться в качестве указателей памяти. Все регистры имеют 3-разрядные кодовые обозначения. Например, регистр ***D*** имеет кодовое обозначение *010*. Такое же кодовое обозначение имеет и регистровая пара ***DE***.

16-разрядный регистр ***PA*** предназначен для сохранения адреса операнда при обращении к памяти на время машинного цикла. Выход регистр ***PA*** соединен с буферным регистром адреса ***БА***.

Счетчик команд ***PC*** хранит адрес текущей ячейки программной памяти. После выбора очередного байта любой команды содержимое ***PC*** увеличивается на единицу.

Указатель стека ***SP*** адресует вершину стека. В микропроцессорных системах с МП КР580ВМ80А стек моделируется в оперативной памяти. В этих МП содержимое ***SP*** увеличивается при выборке данных из стека и уменьшается при загрузке данных в стек, при этом обмен данными между МП и стеком осуществляется 16-разрядными словами путем последовательной передачи по шине старшего и младшего байтов слова.

Программно недоступный регистр ***WZ*** используется для временного хранения второго и третьего байтов многобайтных команд.

Блок управления МП содержит регистр команд ***РК***, дешифратор команд (*ДшК*) и схемы синхронизации и управления (*БСУ*). С помощью этого блока обеспечивается формирование сигналов, настраивающих операционный блок на выполнение операций, определяемых кодом команды, и сигналов, осуществляющих внешние обмены между МП и внешними устройствами (ВУ). Обме-

ны информацией между внутренними блоками микропроцессора выполняются по 8-разрядной внутренней шине данных (*ВШД*). Внешние обмены информацией между МП и ВУ осуществляются по системной шине, объединяющей линии данных, адреса и управления. Каждый внешний обмен реализуется в течение одного машинного цикла. Протокол обмена информацией по системной шине включает правила организации последовательностей сигналов, обеспечивающих правильную передачу информации между компонентами микропроцессорной системы. Сигналы системной шины, формируемые МП КР580ВМ80А, показаны на рис. 19.

Шина данных объединяет 8 двунаправленных тристабильных линий ***D7***–***D0***. По этой шине осуществляется обмен любой информацией в системе: по ней передаются команды, операнды, результаты операций, вводимые и выводимые данные. Направление передачи определяется сигналами *DBIN* и *WR*, которые генерирует МП в каждом машинном цикле.

Однонаправленная шина адреса A15–A0 предназначена для передачи адресной информации из МП в память и в устройства ввода/вывода (УВВ). Адресуемое пространство памяти, определяемое разрядностью шины адреса, составляет 64 Кбайт. Адресуемое пространство устройств ввода/вывода составляет 256 устройств ввода и 256 устройств вывода. При обращении к УВВ 8-разрядный адрес порта дублируется на линиях шины адреса. Он одновременно выдается на линии ***A15***–***A8*** и ***A7***–***A0***. Порты УВВ можно подключать как к линиям ***A7***–***A0***, так и к линиям ***A15***–***A8***. Такое решение обеспечивает возможность выравнивания нагрузки на линиях шины адреса.

Шина управления состоит из десяти линий, по которым передаются управляющие сигналы, определяющие характер и порядок функционирования компонентов микропроцессорной системы. Сигналы управления имеют следующее назначение.

Входной сигнал сброса *RESET* инициализирует счетчик команд ***PC*** нулевым значением, определяя начало выполнения программы с команды, размещенной в нулевой ячейке памяти.

Входной сигнал готовности *READY*, формируемый внешними устройствами при их готовности к обмену, позволяет организовать асинхронный обмен данными. Неактивный сигнал *READY* приостанавливает обмен данными по шине. С помощью этого сигнала внешние устройства управляют скоростью обмена информацией с МП.

Выходной сигнал ожидания *WAIT* формируется микропроцессором, когда его работа приостановлена.

Выходные сигналы *DBIN* и *WR* определяют направление передачи по шине данных относительно микропроцессора. Сигнал

DBIN формируется при передаче данных из внешнего устройства в микропроцессор, а сигнал *WR* – при обменах, в которых информация передается от микропроцессора во внешнее устройство.

Входной сигнал запроса прерывания *INT* формируется периферийным устройством (ПУ) при его готовности к обмену информацией по прерыванию. Реагируя на этот сигнал, МП прерывает выполнение текущей программы, временно запоминает ее состояние, выполняет программу обработки запроса (осуществляет обмен данными с устройством), после чего восстанавливает прежнее состояние прерванной программы и продолжает ее выполнение.

Выходной сигнал разрешения прерываний *INTE* разрешает или запрещает обслуживание запросов прерываний от периферийных устройств. Сигнал *INTE* формируется внутренним триггером разрешения прерывания. Этот триггер управляется программно. Команда *EI* разрешает прерывания, команда *DI* – запрещает.

Входной сигнал запроса прямого доступа к памяти *HOLD* информирует МП о необходимости обмена данными между быстродействующим ПУ и памятью без участия МП. Реагируя на этот сигнал, МП приостанавливает выполнение текущей программы, переводит буферные регистры шин адреса и данных в состояние высокого сопротивления (отключается от шин) и формирует выходной сигнал *HLDA*, разрешающий ПУ, инициирующему прямой доступ к памяти, распоряжаться системной шиной.

Выходной сигнал синхронизации *SYNC* идентифицирует начало каждого машинного цикла, в течение которого осуществляется обмен информацией между МП и внешним устройством.

5.2. **Синхронизация работы микропроцессорной системы на базе МП КР**580**ВМ**80**А**

МП тактируется двумя последовательностями импульсов *C1* и *C2*, вырабатываемых ИС тактового генератора КР580ГФ24. Импульсные последовательности имеют амплитуду 12 В, частоту 0,5–3,0 МГц и не перекрываются во времени. Кроме того, генератор тактовых импульсов ГТИ формирует положительный импульс стандартного ТТЛ-уровня и отрицательный импульс «строб состояния», который далее для краткости будем обозначать *STB* длительность не менее

$$\frac{T_{оп}}{(9-15)нс},$$

где $T_{оп}$ – период тактовых сигналов опорной частоты.

Формирование всех этих импульсов происходит с частотой повторения, равной девяти периодам колебаний задающего квар-

цевого резонатора, подключаемого к выводам ГТИ. Следовательно, для получения частоты следования тактовых импульсов, равной 2 МГц, потребуется кварцевый резонатор с частотой, равной 18 МГц. Одновременно ГТИ используется для формирования сигналов *ГОТОВНОСТЬ* и *СБРОС*. Функциональная схема ГТИ и его подключение к выводам МП показаны на рис. 22, 23.

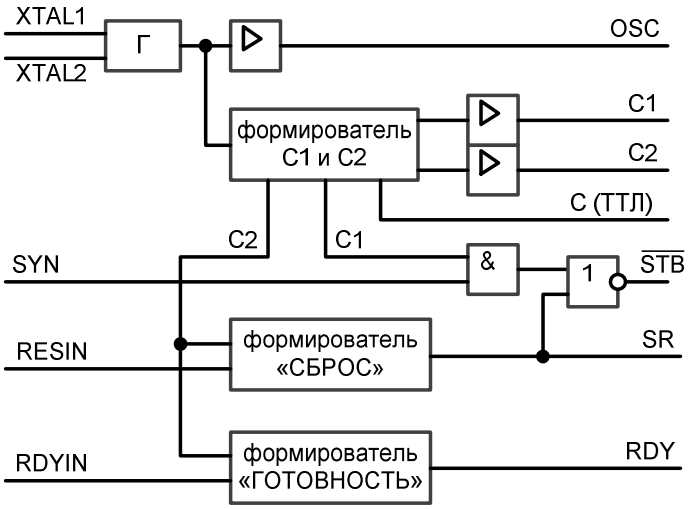


Рис. 22. Функциональная схема генератора тактовых импульсов КР580ГФ24

ИС ГТИ содержит генератор ***Г*** гармонических колебаний, к выводам *XTAL1* и *XTAL2* которого подключается кварцевый резонатор. Гармонические колебания с выхода генератора поступают на выход ИС ГТИ и используются внутри него для управления схемами формирования тактирующих последовательностей *C1* и *C2*, сигнала сброса *SR* и сигнала готовности *RDY*. Наличие гармонических колебаний на выводе *OSC* ГТИ может быть использовано для контроля его работы или во внешних модулях МС. Вывод *C* ГТИ используется для вывода из генератора сигнала высокого уровня, стандартного для ТТЛ-схем, длительностью пять периодов опорной частоты кварцевого резонатора. Этот сигнал может быть использован во внешних устройствах как эталонный импульсный сигнал стабилизированной частоты. Вывод *RESIN* (вход сброса) используется для подключения кнопки, переводящей МП в режим начальной установки. Заметим, что формирование системного стробирующего сигнала *STB* производится ИС ГТИ после подачи на его вывод *SYN* (вход синхросигнала) синхросигнала с соответствующего вывода МП.

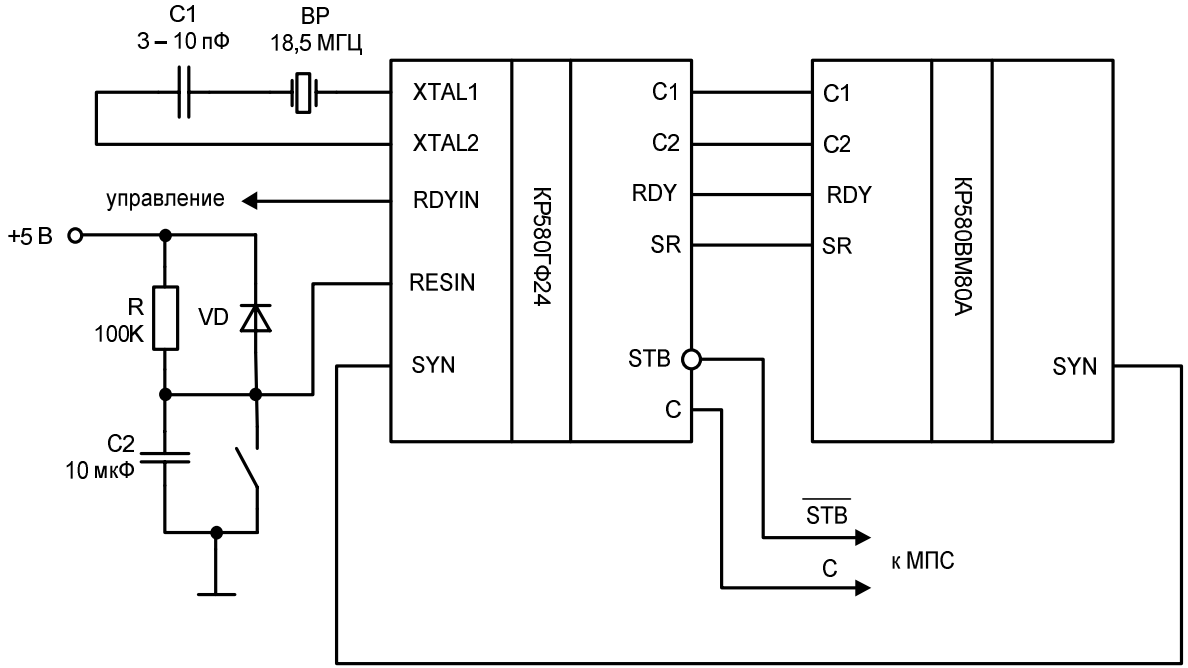


Рис. 23. Схема подключения ИС генератора тактовых импульсов к МП КР580ВМ80А

Временные соотношения тактовых импульсных последовательностей *C1* и *C2* и формирование системного сигнала *STB* показаны на рис. 24.

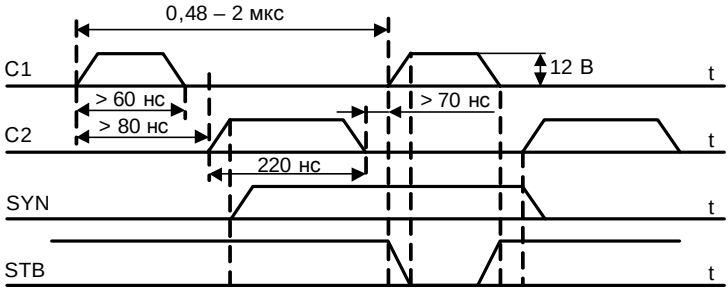


Рис. 24. Временные соотношения сигналов последовательностей C1 и C2, формирование сигнала STB

По переднему фронту тактирующего сигнала *C2* синхросигнал *SYN* подается на вход *SYN* ГТИ и с его помощью на вывод STB ГТИ передается инвертированный сигнал *C1*, который является системным стробом.

Период синхросигналов *C1* и *C2* называется **машинным тактом**. Длительность машинного такта может быть установлена произвольно в диапазоне от 0,5 до 2 мкс.

Условное обозначение ИС генератора тактовых импульсов КР580ГФ24 представлено на рис. 25.

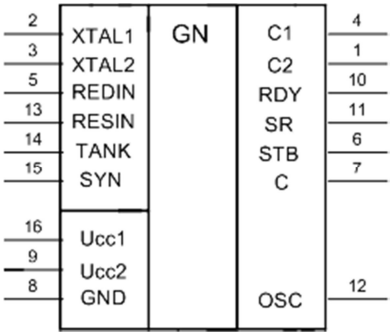


Рис. 25. Условное обозначение ИМС КР580ГФ24

Назначение выводов ИМС КР580ГФ24:

SR – выход установки в исходное состояние микропроцессора и системы.

RESIN – вход установки «0».

RDYIN – вход сигнала «Готовность».

RDY – выход сигнала «Готовность».

SYN – вход сигнала «Синхронизация».

C – выход тактового сигнала, синхронного с фазой *C2*.

STB – выход стробирующего сигнала состояния.

OSC – выход тактовых сигналов опорной частоты.

C1, *C2* – выходы тактовых сигналов фазы *C1* и *C2* соответственно.

TANK – вход для подключения колебательного контура.

XTAL1, *XTAL2* – входы для подключения резонатора.

5.3. **Программная модель микропроцессорной системы на базе МП КР**580**ВМ**80**А**

Знание особенностей структуры микропроцессора, назначения его выводов, электрических и конструктивных параметров необходимо при разработке аппаратной части микропроцессорной системы. В технических описаниях МП КР580ВМ80А приводятся его более подробные структурные схемы, однако для подавляющего большинства пользователей детальное знание особенностей внутренней структуры микропроцессора чаще всего оказывается избыточным, поскольку пользователь в принципе не может изменить его структуру. Число вариантов схем включения МП также невелико, поскольку микропроцессор представляет собой логический автомат с высокой степенью детерминированности связей. В разработанной и изготовленной системе для практического использования МП (составления прикладных программ) достаточно знать его программную модель и систему команд.

Модель микропроцессора содержит только узлы, наиболее важные для понимания процесса его работы. Модель может содержать программно-доступные и программно-недоступные узлы.

На рис. 26 представлены программно-доступные (адресуемые в командах в явной или неявной форме) узлы МП, памяти и устройств ввода-вывода. На рис. 27 указаны программно-недоступные узлы МП, наиболее существенные для процесса выполнения команд.

В состав программной модели включены 8-битные регистры блока ***РОН*** (***B***, ***C***, ***D***, ***E***, ***H***, ***L***, ***A***), 16-битовые регистровые пары (***B***, ***D***, ***H***), указатель стека ***SP***, 256 портов ввода и 256 портов вывода, ячейки памяти общим числом до 64К. При обращении к памяти используются прямая и косвенная адресации. Основным указателем памяти при косвенной адресации является регистровая пара ***HL***. Ячейка памяти, адрес которой определяется содержимым пары ***HL***, обозначается ***M*** (от *MEMORY* – память). Указателями памяти также могут выступать регистровые пары ***BC***, ***DE*** и указатель стека ***SP***.

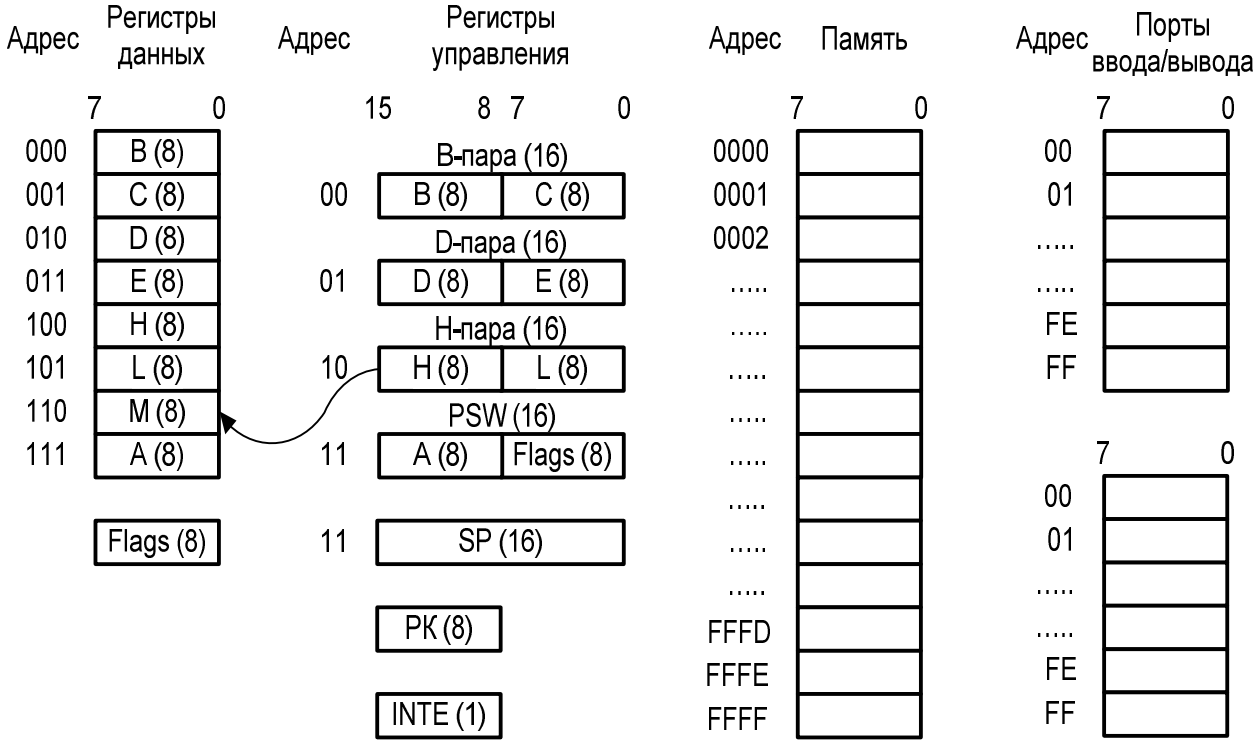


Рис. 26. Программная модель микропроцессорной системы с МП КР580ВМ80А
(программно-доступные узлы МП и МП системы)

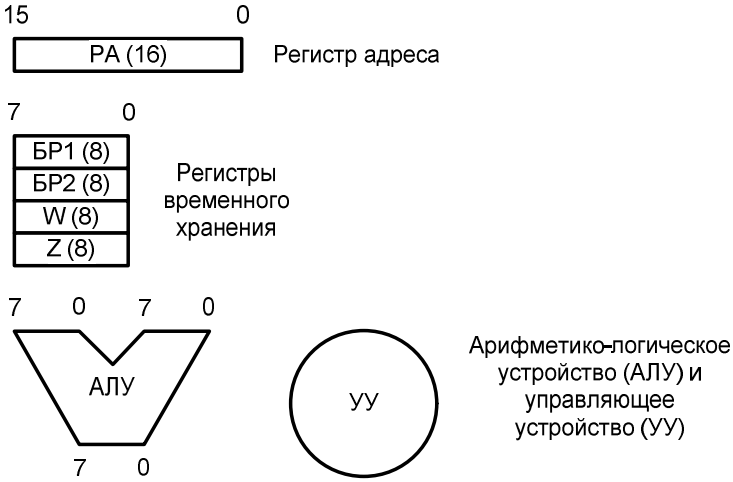


Рис. 27. Программно-недоступные узлы МП КР580ВМ80А

Все регистры блока ***РОН***, в том числе и ячейка ***М***, могут использоваться для создания программно-управляемых счетчиков.

Регистр команд ***РК*** доступен неявно. В этот регистр помещается первый байт команды, выбираемой из памяти при выполнении микропроцессором программы.

INTE − триггер разрешения прерывания («1» − прерывания разрешены, «0» − прерывания запрещены).

Режимы адресации и система команд

Система команд МП КР580ВМ80А состоит из 78 базовых команд и 111 кодов операции. Синтаксис большинства команд ассемблерного языка состоит из мнемонического обозначения функций команды, вслед за которым могут размещаться операнды, указывающие методы адресации и типы данных. Формат команды МП зависит от типа выполняемой операции и может быть одно-, двух- и трехбайтовым. Код операции любой команды размещается в первом байте, а второй и третий байты, если они являются частью команды, могут содержать адрес операнда в памяти или сам операнд. В качестве операндов могут использоваться байты и 16-битные слова, при этом большинство операндов команд являются байтами. Для адресации используют следующие типы адресации: *прямую*, *регистровую*, *непосредственную*, *неявную* и *косвенную*.

Прямая адресация, при которой в коде команды указывается 16-битный адрес, используется в командах пересылок для задания адреса операнда в памяти. Для обозначения прямого адреса

операнда в памяти в мнемониках команд применяется аббревиатура *ADDR*.

Регистровая адресация применяется для адресации регистров блока ***РОН*** (8-разрядных регистров ***B***, ***C***, ***D***, ***E***, ***H***, ***L***, ***A*** и 16-разрядных регистровых пар ***B***, ***D***, ***H***).

Операндом команд с непосредственной адресацией может быть только источник непосредственных данных. Непосредственные данные это 8- или 16-битные константы или прямые адреса, для представления которых используется второй или второй и третий байт команды.

МП КР580ВМ80А выполнен по схеме одноадресного вычислителя. В нем один из операндов размещается в аккумуляторе и результат помещается в аккумулятор. При выполнении команд арифметических и поразрядных логических операций аккумулятор адресуется неявно. Неявная адресация используется и в некоторых других командах, например, в командах работы со стеком и в командах загрузки/запоминания содержимого аккумулятора из памяти/в память.

Косвенную адресацию применяют для обращения к операндам в памяти. Выше отмечено, что указателями адреса при косвенной адресации могут выступать регистровые пары ***BC***, ***DE***, ***HL*** и указатель стека ***SP***.

Систему команд МП КР580ВМ80А по функциональному признаку удобно подразделить на нескольких групп:

– команды пересылок;

– команды арифметических и логических операций;

– команды передачи управления.

Рассмотрим общие закономерности кодирования различных групп команд. Поскольку выделено три группы команд, то для их идентификации достаточно всего двух бит. Пусть команды пересылок имеют код *01*, команды арифметических и логических операций – код *10*, команды передачи управления – код *11*, а для команд других типов, в частности, с непосредственной адресацией зарезервируем код *00*.

Группа команд пересылок (передачи данных). В нее включены команды с мнемониками *MOV* (собственно пересылки), *PUSH*, *POP* (загрузки в стек и извлечения из стека), команды ввода-вывода *IN*, *OUT* и некоторые другие, в том числе команды обмена, загрузки и запоминания содержимого регистровых пар (мнемоника *LXI*). Команды пересылок не модифицируют флаги результата.

Наиболее представительной группой команд являются команды пересылок с мнемоникой *MOV D,S*, которые обеспечивают передачу данных из регистра-источника (*SOURCE*) в регистр-приемник

(*DESTINATION*). В командах *MOV* двухбитный код *01* в принципе полностью идентифицирует код операции. Оставшиеся шесть бит первого байта команды, выделяемого для кода операции, можно использовать для указания адреса источника и приемника данных. Кроме регистров блока *РОН* (***B***, ***C***, ***D***, ***E***, ***H***, ***L***) приемником и источником данных в командах пересылок с этой мнемоникой могут выступать аккумулятор и ячейка памяти ***M***, адресуемая косвенно.

В обобщенном виде команды пересылок МП КР580ВМ80А с мнемоникой *MOV* (рис. 28) можно представить восьмеричным кодом *1DS*, где 1 – код команд пересылок с мнемоникой *MOV*; ***D*** (*DDD*) – код регистра приемника; ***S*** (*SSS*) – код регистра источника. Коды (адреса) регистров ***D*** и ***S*** указаны в поле адреса регистра регистровой модели МП (см. рис. 1). Например, команда *MOV B,L* (переслать содержимое регистра ***L*** в регистр ***B***) имеет восьмеричный код *105*, а команда *MOV A,M* (переслать содержимое ячейки памяти ***M***, адрес которой указан в регистровой паре ***HL***, в регистр ***A***) – восьмеричный код *176*. Пересылки из памяти/в память в МП КР580ВМ80А запрещены. По этой причине из 64 возможных пересылок, определяемых кодом $1DS_8$, разрешенными являются 63 пересылки между любыми регистрами, а также регистрами и памятью (код *166* зарезервирован для команды останова *HLT*).

Рис. 28. Формат команд с мнемоникой *MOV*

Поскольку для кодирования многочисленных команд пересылок двухбитного кода *01* оказывается недостаточно, команды пересылок с мнемоникой, отличной от *MOV*, размещаются в группах команд, идентифицируемых кодами *00* и *11*. Например, двухбайтовые команды непосредственной загрузки регистров с мнемоникой *MVI* R_{D}*,B2* (рис. 29) имеют восьмеричный код *0D6 B2*, где *0x6* – тип команд загрузки 8-битных регистров с непосредственной адресацией; ***D*** (*DDD*) – код регистра приемника; ***B2*** – непосредственный операнд (данные), содержащийся во втором байте команды. Коды (адреса) регистров ***D*** указаны в поле адреса регистра регистровой модели МП (см. рис. 26). Другие команды группы пересылок менее однородны по составу, кодирование этих команд труднее поддается формализации.

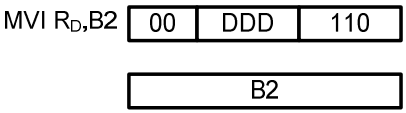


Рис. 29. Формат команд с мнемоникой *MVI*

Двухбайтные команды ввода *IN PORT* и вывода *OUT PORT* (рис. 30) осуществляют внешние обмены байтом данных между аккумулятором и регистром данных периферийного устройства. Адрес внешнего регистра указывается во втором байте команд. Наличие специальных команд ввода/вывода позволяет отделить адресное пространство ввода/вывода от адресного пространства памяти.

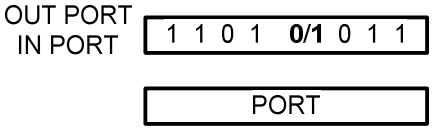


Рис. 30. Формат команд *IN PORT* и *OUT PORT*

Обмен данными между памятью и аккумулятором также может осуществляться с помощью команд загрузки с мнемоникой *LDA* и запоминания с мнемоникой *STA* (рис. 31). Соответствующие команды с прямой адресацией операнда в памяти *LDA ADDR* и *STA ADDR* являются трехбайтными. Они выполняются за четыре машинных цикла. Однобайтные команды загрузки и запоминания с косвенной адресацией операнда в памяти имеют мнемоники *LDAX* и *STAX* (рис. 32). В качестве указателей памяти в этих командах используют пары регистров ***B*** и ***D***. Обобщенный код команд *STAX* и *LDAX 0RP2* (для команды *STAX*) и *0RP12* (для команды *LDAX*), где ***RP*** – код (адрес) регистровой пары, указываемый в соответствующем поле адреса регистровой пары регистровой модели МП (см. рис. 26).

Группа команд арифметических и поразрядных логических операций. МП КР580ВМ80А, как отмечено выше, выполнен по схеме одноадресного вычислителя. При выполнении арифметических и поразрядных логических операций один из операндов команд этой группы всегда размещается в аккумуляторе, и результат операции помещается в аккумулятор.

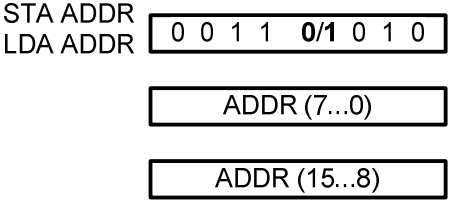


Рис. 31. Формат команд *STA ADDR* и *LDA ADDR*

Рис. 32. Формат команд *STAX RP* и *LDAX RP*

В качестве второго операнда может использоваться содержимое любого регистра блока *РОН* или ячейки памяти ***M***, адресуемой косвенно по адресу в регистровой паре ***HL***. По результату операции модифицируются флаги.

Двухбитовый код *10*, идентифицирующий группу команд арифметических и поразрядных логических операций, не позволяет однозначно определить многообразие операций преобразования данных. Для указания конкретного типа арифметических и поразрядных логических операций в коде команды используется дополнительное поле (рис. 33). Обобщенный код команд указанной группы имеет вид *2XS*, где 2 – код команд арифметических и поразрядных логических операций; ***X*** (*XXX*) – код типа арифметических и поразрядных логических операций (табл. 2); ***S*** (*SSS*) – код регистра источника. Коды (адреса) регистров ***S*** указаны в поле адреса регистра регистровой модели МП (см. рис. 26).

Рис. 33. Формат арифметических и логических команд с регистровой адресацией

Вычислительные возможности микропроцессора ограничены командами сложения *ADD R* и вычитания *SUB R* 8-битных операндов. Операции умножения и деления, а также операции с другими форматами данных, должны выполняться с помощью подпрограмм. Во многих применениях 8-разрядной длины слова микропроцессора для точного представления данных недостаточно. Поэтому данные представляют в виде многобайтных чисел. В памяти такие числа хранятся в смежных ячейках и адресуются по младшему байту. Обработку многобайтных чисел выполняют путем последовательной обработки отдельных байтов, используя для этого специальные команды сложения с переносом *ADC R* и вычитания с заёмом *SBB R*.

В группу арифметических команд входит команда неразрушающего вычитания (арифметического сравнения) *CMP R*, которая производит вычитание из содержимого аккумулятора значение адресуемого операнда, модифицирует по результату все флаги, но не изменяет содержимое аккумулятора. Команду сравнения удобно использовать, например, при упорядочении заданного массива

данных. Базовый адрес массива загружается в регистры ***HL***, первый элемент массива помещается в аккумулятор, а затем с помощью команды *CMP M* последующие элементы массива сравниваются со значением в аккумуляторе, не изменяя его значения. Результат сравнения фиксируется во флагах переноса ***C*** и нулевого результата ***Z***. Перестановки элементов массива (собственно упорядочение) производят в соответствии со значением флагов ***C*** и ***Z***.

Таблица 2

Обобщенный код команд

Арифметические и логические операции (мнемоника команд и выполняемое действие)	Двоичный код	Восьмеричный код
Сложение *ADD*: $\big((A) \leftarrow (A) + (R_S)\big)$	000	0
Сложение с переносом *ADC*: $\big((A) \leftarrow (A) + (R_S) + C\big)$	001	1
Вычитание *SUB*: $\big((A) \leftarrow (A) - (R_S)\big)$	010	2
Вычитание с заёмом *SBB*: $\big((A) \leftarrow (A) - (R_S) - C\big)$	011	3
Логическое И *ANA*: $\big((A) \leftarrow (A) \wedge (R_S)\big)$	100	4
Исключающее ИЛИ *XRA*: $\big((A) \leftarrow (A) \oplus (R_S)\big)$	101	5
Логическое ИЛИ *ORA*: $\big((A) \leftarrow (A) \vee (R_S)\big)$	110	6
Сравнение *CMP*: $\big((A) - (R_S)\big)$	111	7

Поразрядные логические операции выполняются с помощью команд *ANA R*, *ORA R*, *XRA R* независимо для всех разрядов операндов. При выполнении этих команд модифицируются все флаги, кроме флагов ***AC*** и ***C***, который принудительно сбрасывается в 0.

Команду *ANA R*, осуществляющую поразрядную конъюнкцию операндов (см. табл. 2), применяют для проверки значения определенного бита в байте, содержащемся в аккумуляторе, с помощью другого байта-маски. В частности, если требуется проверить состояние бита $\boldsymbol{A_3}$, то необходимо использовать маску *00001000*. О состоянии бита $\boldsymbol{A_3}$ можно судить, проанализировав флаг ***Z***.

$$Z = \overline{A_3}.$$

Команду *ANA R* также применяют для сброса определенных бит слова в аккумуляторе. Используемая для этого маска должна содержать 0 в разрядах сбрасываемых бит и 1 в некорректируемых разрядах.

Команда *ORA R* реализует операцию поразрядной дизъюнкции операндов (см. табл. 2). Эту команду применяют для установки определенных битов байта в аккумуляторе с помощью байта-маски. Другим использованием команды *ORA R* является упаковка байта в аккумуляторе из полей других байт. Например, результатом операции ORA с операндами $00001X_2X_1X_0$ и $Y_7Y_6Y_510000$ будет упакованный байт $Y_7Y_6Y_511X_2X_1X_0$.

Команда *XRA R* (Исключающее ИЛИ) производит операцию поразрядного сложения операндов по *mod 2*. В соответствии с тождеством

$$1 \oplus X = \overline{X},$$

используя байт-маску, эту команду удобно использовать для инвертирования определенных бит содержимого в аккумуляторе. Например, для инвертирования битов A_2 и A_5 маска должна содержать 1 в разрядах 2 и 5 и 0 для неинвертируемых разрядов, т.е. иметь вид *00100100*. Другое применение команды *XRA* связано со сравнением слов на абсолютное равенство. Действительно, после выполнения команды *XRA*, согласно тождеству

$$X \oplus \overline{X} = 0,$$

нулевой результат формируется только при полной идентичности всех разрядов операндов. О равенстве операндов можно судить по значению флага ***Z***.

В дополнение к рассмотренным командам арифметических и поразрядных логических операций, кодируемых выражением $2XS_8$, МП КР580ВМ80А реализует сравнительно большое число других команд этой группы. Прежде всего, это команды арифметических и логических операций с непосредственными данными, команды инкремента/декремента содержимого регистров, команды сдвиговых операций, арифметические команды, оперирующие 16-битными данными, и некоторые другие.

Двухбайтные команды арифметических и поразрядных логических операций с непосредственной адресацией (рис. 34) имеют обобщенный код команд *3X6 B2*, где ***X*** (*XXX*) – код типа арифметических и поразрядных логических операций (см. табл. 2). Операндами этих команд являются содержимое аккумулятора и второй байт команды ***B2***.

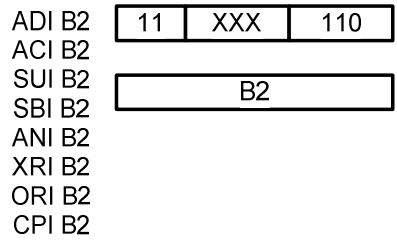


Рис. 34. Формат арифметических и поразрядных логических команд с непосредственной адресацией

Однобайтные команды инкремента *INR R* ($R \leftarrow R + 1$) и декремента *DCR R* ($R \leftarrow R - 1$) (рис. 35) имеют обобщенный код *0D4* (инкремента) и *0D5* (декремента), где ***D*** (*DDD*) – код регистра приемника (см. рис. 26). Они обеспечивают увеличение (инкремент) или уменьшение (декремент) содержимого адресуемого регистра на единицу. Команды инкремента/декремента воздействуют на все флаги, кроме флага переноса ***C***, который не изменяется.

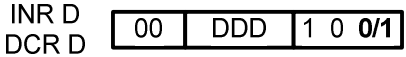


Рис. 35. Формат команд *INR D* и *DCR D*

МП КР580ВМ80А реализует арифметическую операцию сложения операндов в упакованном формате двоично-десятичных чисел (две десятичные цифры *BCD*-формата в байте). Сложение *BCD*-чисел выполняется в два этапа. Сначала выполняется команда *ADD R*. При ее выполнении операнды в упакованном двоично-десятичном формате складываются как обычные двоичные числа. Затем с помощью команды *DAA* производится коррекция в общем случае неправильного результата сложения десятичных чисел. Действие команды *DAA* (рис. 36) следующее. Восьмибитное число результата в аккумуляторе рассматривается как две четырехбитные двоично-десятичные цифры. Коррекция результата суммирования выполняется по правилам:

1. Если значение младших четырех бит больше 9 или признак межтетрадного переноса ***AC*** равен 1, то к содержимому аккумулятора прибавляется число *06h*, образуя правильное *BCD*-число в младшей тетраде.

2. Если после этого значение старших четырех бит больше 9 или установлен флаг переноса ***C***, то к содержимому аккумулятора прибавляется число *60h*, образуя правильное *BCD*-число в старшей тетраде. Перенос ***C***, формируемый при выполнении команды *DAA*, указывает, что сумма двух исходных *BCD*-чисел больше, чем *99*.

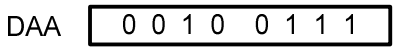


Рис. 36. Формат команды *DAA*

К рассматриваемой группе команд арифметических и логических операций также относятся однобайтные команды циклического сдвига вправо и влево. Операндом этих команд является содержимое аккумулятора, в него же помещается и результат. Действие команд сдвига поясняет рис. 37.

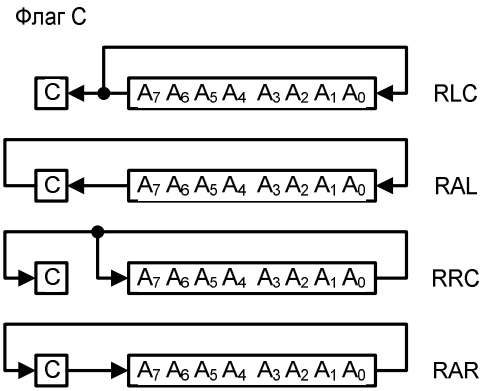


Рис. 37. Команды сдвига

В командах циклического сдвига (команды *RRC* и *RLC*) выдвигающийся бит помещается на место освобождающегося и, кроме того, фиксируется во флаге переноса ***C***. В командах циклического сдвига через перенос (команды *RAR* и *RAL*) выдвигающийся бит помещается в флаг ***C***, а текущее значение флага ***C*** передается в освобождающийся бит. Команды циклического сдвига не изменяют состояние флагов, кроме флага переноса ***C***.

Обобщенный код команд сдвига *0C7*, где ***C*** (*CC*) – тип команды сдвига: 0 – *RLC*, 1 – *RRC*, 2 – *RAL*, 3 – *RAR* (рис. 38).

Однобайтные неявные команды инверсии содержимого аккумулятора *CMA* $\left((\overline{A}) \leftarrow (A)\right)$, инверсии признака переноса ***C*** *CMC* $(\overline{C} \leftarrow C)$ и установки признака переноса ***C*** *STC* $(C \leftarrow 1)$, также входят в рассматриваемую группу команд.

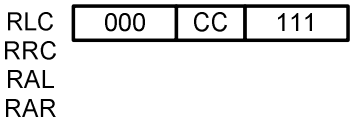


Рис. 38. Формат команд сдвига

Обобщенный код указанных команд 1X7, X (XX) – тип команды: 01 – CMA, 10 – STC, 11 – CMC (рис. 39).

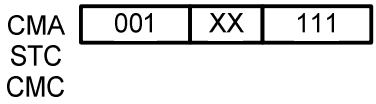


Рис. 39. Формат команд *CMA*, *STC*, *CMC*

Группа команд передачи управления. Команды передачи управления предназначены для изменения естественного порядка выполнения команд программы при реализации разветвляющихся и циклических алгоритмов, вызовов подпрограмм и возврата из них. В системе команд МП КР580ВМ80А содержится сравнительно большое число команд передачи управления, которые подразделяются на безусловные и условные переходы.

Команды безусловного перехода (*JMP*), вызова подпрограмм (*CALL*) и возврата из них (*RET*) передают управление по адресу, указываемому в команде (*JMP*, *CALL*) или по адресу, выбираемому из стека (*RET*).

Трехбайтная команда передачи управления *JMP ADDR* (рис. 40) содержит полный 16-битный адрес перехода. При ее выполнении адрес перехода загружается в счетчик команд ***PC***, а текущее содержимое ***PC*** теряется. Команда *JMP ADDR* выполняется за три машинных цикла.

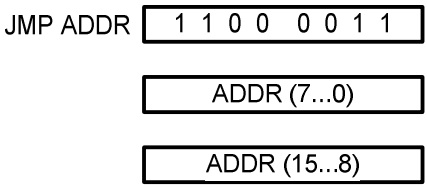


Рис. 40. Формат команды *JMP ADDR*

Трехбайтная команда вызова подпрограмм *CALL ADDR* (рис. 41) имеет формат, аналогичный команде *JMP*, однако при ее выполнении адрес следующей по порядку команды не теряется. Выполнение команд вызова подпрограмм начинается с запоминания адреса возврата (адреса следующей после *CALL* команды) в стеке, и только после этого происходит перезагрузка ***PC*** адресом первой команды подпрограммы. Подпрограмма должна завершаться однобайтной командой возврата *RET* (рис. 42), перезагружающей содержимое ***PC*** адресом возврата. Команда *CALL ADDR* выполняется за пять машинных циклов, а команда *RET* – за три машинных цикла.

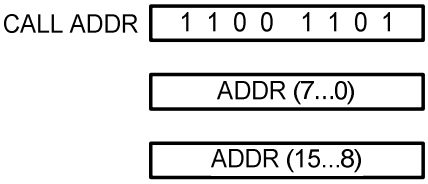


Рис. 41. Формат команды *CALL ADDR*

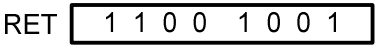


Рис. 42. Формат команды *RET*

Система команд МП КР580ВМ80А содержит еще две команды безусловной передачи управления – команду *RST N* и команду *PCHL*. Однобайтная команда вызова *RST N* предназначена для обработки прерываний и может использоваться для вызова подпрограмм по фиксированным адресам. При поступлении запроса прерывания в последнем такте последнего машинного цикла команды устанавливается внутренний триггер прерывания. Следующим машинным циклом становится машинный цикл прерывание М8. В байте состояния цикла М8 формируется бит подтверждения прерывания *INTA*, с помощью которого периферийное устройство, запросившее прерывание, выдает на системную шину данных однобайтную команду повторного запуска *RST N* с кодом *11NNN111* (рис. 43). Трехбитное поле *NNN* называется вектором прерывания. При выполнении команды *RST N* содержимое счетчика команд ***PC*** (адрес возврата) запоминается в стеке, а в счетчик команд загружается начальный адрес обработчика прерываний *00000000 00NNN000*. Таким образом, в зависимости от значения трехбитного поля *NNN*, формируемого периферийным устройством или указываемым программистом в команде *RST N*, микропроцессор после выполнения данной команды вызывает одну из восьми 8-байтовых подпрограмм, расположенных в первых 64 ячейках памяти по адресам *000 0008*, *000 0108*, *000 0208*, …, *000 0708*. Подпрограмма должна завершаться командой возврата *RET*. Команда вызова *RST N* выполняется за три машинных цикла.

Рис. 43. Формат команды *RST N*

Однобайтная команда пересылки 16-битных операндов *PCHL* (рис. 44) по выполняемым функциям является особой командой безусловного перехода. При ее выполнении содержимое регистро-

вой пары ***HL*** загружается в счетчик команд ***PC***, и микропроцессор продолжает программу с адреса, определяемого содержимым ***HL***, при этом текущее содержимое ***PC*** теряется. Команду *PCHL* иногда называют командой множественного ветвления. Она выполняется за один машинный цикл.

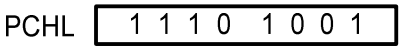


Рис. 44. Формат команды *PCHL*

Наряду с командами безусловной передачи управления в системе команд МП КР580ВМ80А имеется сравнительно многочисленная группа команд условной передачи управления, включающая команды условного перехода, условного вызова подпрограмм и условного возврата из подпрограмм. Передача управления при выполнении команд указанной группы осуществляется только в случае, если выполняется условие, заданное в коде операции. Если условие не выполняется, программа продолжается с команды, следующей за командой условной передачи управления, при этом сама команда условной передачи управления становится эквивалентной холостой команде. Проверяемым условием является текущее значение одного из флагов регистра признаков. Для удобства программирования предусмотрены команды, осуществляющие передачу управления по единичному и нулевому значениям каждого из флагов, кроме флага ***AC***.

Обобщенный код команд условной передачи управления имеет вид *3YZ* (рис. 45), где 3 – код класса команд передачи управления, ***Y*** (*YYY*) – код проверяемого условия (табл. 3), ***Z*** (*ZZZ*) – код одной из трех групп команд условной передачи управления (табл. 4).

Всего имеются 24 команды условной передачи управления, проверяющие единичное и нулевое значения каждого из четырех флагов:

флаг ***Z***: вызовы *CZ*, *CNZ*; переходы *JZ*, *JNZ*; возвраты *RZ*, *RNZ*;
флаг ***S***: вызовы *CM*, *CP*; переходы *JM*, *JP*; возвраты *RM*, *RP*;
флаг ***C***: вызовы *CC*, *CNC*; переходы *JC*, *JNC*; возвраты *RC*, *RNC*;
флаг ***P***: вызовы *CPE*, *CPO*; переходы *JPE*, *JPO*; возвраты *RPE*, *RPO*.

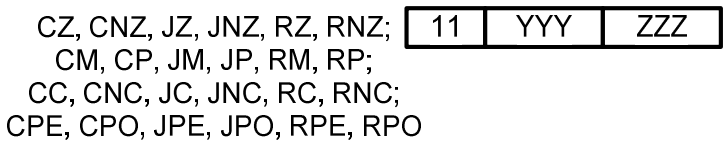


Рис. 45. Формат команд условной передачи управления

Таблица 3

Обобщенный код команд условной передачи управления

Флаг признака результата	Мнемоника в команде	Состояние флага	Двоичный код флага
Ненулевой	NZ	0	000
Нулевой	Z	1	001
Нет переноса	NC	0	010
Перенос	C	1	011
Нечётный	PO	0	100
Чётный	PE	1	101
Положительный	P	0	110
Отрицательный	M	1	111

Таблица 4

Пример кода команд передачи

Название группы команд	Мнемоника в команде	Двоичный код группы команд
Условный возврат	R	000
Условный переход	J	010
Условный вызов	C	100

Отличительной особенностью системы команд 8-разрядного МП КР580ВМ80А является наличие команд пересылок и арифметических операций, операндами которых выступают 16-битные целые числа без знака.

Трехбайтные команды непосредственной загрузки с мнемоникой *LXI RP, B2,B3* (рис. 46) обеспечивают инициализацию регистровых пар ***BC***, ***DE***, ***HL*** и указателя стека ***SP*** исходными 16-битными значениями. Обобщенный код команд непосредственной загрузки регистровых пар *0RP1*, где ***RP*** – код (адрес) регистровой пары, указываемый в соответствующем поле адреса регистровой пары регистровой модели МП (см. рис. 26).

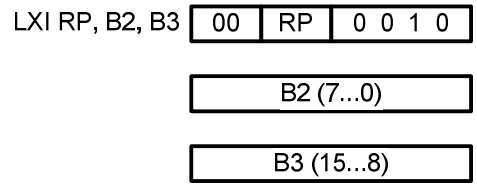


Рис. 46. Формат команд *LXI RP, B2, B3*

Командами 16-битных пересылок являются однобайтные команды засылки содержимого регистровой пары в стек (*PUSH RP*) и

извлечения из стека (*POP RP*). В них код ***RP*** определяет имя регистровой пары ***BC***, ***DE***, ***HL*** и слово состояния программы ***PSW***, включающее содержимое аккумулятора ***A*** и регистра флагов ***РП***.

С помощью команд *PUSH RP* и *POP RP* программно можно «расширять» пространство внутренних регистров блока ***РОН***, если этих регистров оказывается недостаточно для размещения параметров и промежуточных результатов выполняемой программы. Для освобождения внутренних регистров их содержимое командой *PUSH RP* загружается в стек. Восстановление содержимого регистров осуществляется командой *POP RP*, которая возвращает из стека сохраненные значения в соответствующие регистры. При прерываниях команды *PUSH RP* и *POP RP* часто используются для сохранения и восстановления так называемого контекста программы. Прежде чем непосредственно перейти к обработке прерывания, МП должен сохранить содержимое всех внутренних регистров или, по крайней мере, тех из них, которые будут использоваться в программе обработки. По окончанию обработки состояние прерванной программы должно быть восстановлено. В большинстве случаев при контекстном переключении, как минимум, следует временно сохранять и восстанавливать слово состояния программы ***PSW***.

Восьмеричный код команд *PUSH RP* и *POP RP – 3RP5* и *3RP1* соответственно (рис. 47).

При выполнении команды *PUSH RP* в ячейку памяти с адресом (***SP***)–1 записывается содержимое старшего регистра регистровой пары ***RP***, а в ячейку с адресом (***SP***)–2 – содержимое младшего

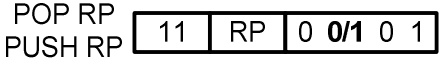


Рис. 47. Формат команд *POP RP* и *PUSH RP*

регистра этой регистровой пары. Содержимое указателя стека ***SP*** уменьшается на 2 (стек растет в область меньших адресов). При выполнении команды *POP RP* (извлечения из стека) данные из вершины стека, адресуемой ***SP***, передаются в младший регистр регистровой пары ***RP***, а в старший регистр этой пары загружается значение из ячейки с адресом (***SP***)+1. После этого содержимое ***SP*** увеличивается на 2 (стек всегда готов к чтению). Для правильной работы стека команды *PUSH* и *POP* должны быть парными.

Трёхбайтные команды загрузки/запоминания содержимого регистровой пары ***HL*** с использованием прямой адресации *LHLD ADDR*, *SHLD ADDR* (рис. 48). Адрес младшего байта при этом указывает-

ся во втором и в третьем байтах команды, а адрес старшего байта вычисляется увеличением на 1 адреса младшего байта.

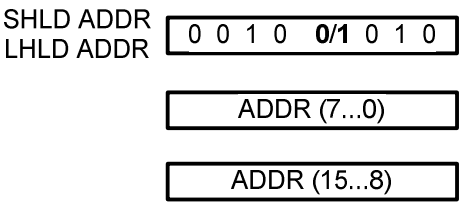


Рис. 48. Формат команд *SHLD ADDR* и *LHLD ADDR*

Команда пересылки двухбайтовых слов из регистровой пары ***HL*** в указатель стека ***SP*** *SPHL* (рис. 49).

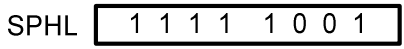


Рис. 49. Формат команды *SPHL*

Команда обмена двухбайтовыми словами из вершины стека и регистровой пары ***HL*** *XTHL* (рис. 50). Содержимое ячейки памяти по адресу (***SP***) записывается в регистр ***L***, а по адресу (***SP***)+1 – в регистр ***H***. При этом содержимое регистровой пары HL сохраняется в указанные ячейки памяти (вершину стека).

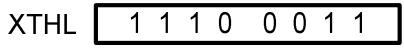


Рис. 50. Формат команды *XTHL*

В число арифметических команд, оперирующих 16-битными числами, входят команды инкремента *INX RP* и декремента *DCX RP* содержимого регистровых пар, а также команды двойного сложения *DAD RP*. Команды *INX RP* и *DCX RP* обеспечивают увеличение или уменьшение содержимого регистровых пар ***BC***, ***DE***, ***HL*** и указателя стека ***SP*** на единицу, и имеют обобщенный восьмеричный код *0RP3* – команда *INX RP* и *0RP13* – команда *DCX RP* (рис. 51), где ***RP*** – код (адрес) регистровой пары, указываемый в соответствующем поле адреса регистровой пары регистровой модели МП (см. рис. 26). Их удобно использовать для модификации указателей памяти при обработке массивов данных с использованием косвенной адресации. Команды двойного сложения *DAD RP* выполняют 16-битное суммирование содержимого ***HL*** с содержимым адресуемой регистровой пары ***BC***, ***DE***, ***HL*** или ***SP***. Восьмеричный код команд *DAD RP – 0RP11* (рис. 52), ***RP*** – код (адрес) регистровой пары, указываемый в соответствующем поле адреса регистро-

вой пары регистровой модели МП (см. рис. 26). Команда *DAD H*, удваивающая значение ***HL***, эквивалентна команде сдвига влево 16-битного операнда. Для сдвига влево 16-битного операнда, размещенного в регистровой паре ***DE,*** удобно использовать последовательность команд, в которой команда *XCHG* (рис. 53) реализует обмен содержимого регистровых пар ***DE*** и ***HL***:

XCHG; DAD H; XCHG

INX RP
DCX RP

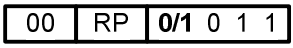


Рис. 51. Формат команд INX RP и DCX RP

DAD RP

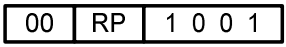


Рис. 52. Формат команд DAD RP

XCHG

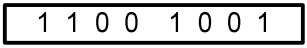


Рис. 53. Формат команды XCHG

Команды управления микропроцессором. В число однобайтных команд этой группы включают команды разрешения *EI* и запрещения *DI* прерывания, команду останова *HLT*, холостую команду *NOP*.

Команды разрешения *EI* и запрещения *DI* прерывания (рис. 54) устанавливают/сбрасывают внутренний триггер разрешения прерываний *INTE*. В состоянии *INTE* = 1 микропроцессор реагирует на внешние запросы прерываний, при *INTE* = 0 – прерывания запрещены.

DI
EI

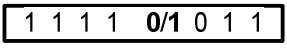


Рис. 54. Формат команд *EI* и *DI*

По команде останова *HLT* (рис. 55) в счетчик команд ***PC*** заносится адрес следующей команды и прекращается выполнение программы, при этом микропроцессор переводится в состояние «Останова». В этом состоянии выходы шин адреса и данных МП устанавливаются в состояние высокого сопротивления и выдается сигнал *WAIT*. Вывести МП из состояния «Останов» можно тремя способами:

– подать на вход *RESET* МП сигнал лог. 1, при этом МП переходит к выполнению команды, записанной по нулевому адресу;

– подать на вход *INT* МП сигнал лог. 1. При разрешенных прерываниях (*INTE* = 1), МП под действием сигнала *INT* переходит в состояние «Прерывание при Останове» и приступает к выполне-

нию программы обработки запроса прерывания. После возврата из прерывающей программы продолжается выполнение программы, начиная с команды, следующей за командой *HLT*;

– подать на вход *HOLD* МП сигнал лог. 1, при этом МП переходит в режим, обеспечивающий прямой доступ в память. После снятия сигнала *HOLD* МП возвращается в состояние «Останов».

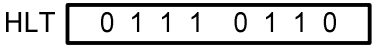


Рис. 55. Формат команды *HLT*

Холостая команда *NOP* (рис. 56) не производит никаких действий, кроме инкремента счетчика команд.

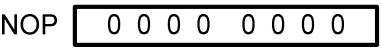


Рис. 56. Формат команды *NOP*

Полный список команд МП КР580ВМ80А приведен в приложении.

5.4. **Выполнение команд в МП КР**580**ВМ**80**А**

При выборке команды код операции, содержащийся в первом байте команды, по шине данных передается в МП и загружается в регистр команд (РК), где хранится в течение всего времени выполнения команды. Дешифратор команд (*ДшК*) перекодирует содержимое регистра команд в управляющее слово, которое передается в блок синхронизации и управления *БСУ*. В *БСУ* также поступают сигналы синхронизации и сигналы управления от внешних устройств (*READY*, *INT*, *HOLD*). Под воздействием названных сигналов *БСУ* генерирует совокупность сигналов управления внутренними операциями в МП и обменом данными между МП и внешними устройствами.

В состав блока *БСУ* входят формирователь машинных тактов, используемый для выработки тактовых импульсов, равных по длительности периоду тактовой частоты, формирователь машинных циклов и формирователь сигнала синхронизации *SYNC*.

Время выполнения команды определяется ее форматом и реализуемыми действиями. На выполнение команд расходуются от одного до пяти машинных циклов. Машинный цикл состоит из 3–5 тактов. Первые три такта всех машинных циклов предназначены для выполнения действий, связанных с внешним обменом между МП и адресуемым в машинном цикле внешним устройством. Такты ***T4*** и ***T5*** в машинном цикле зарезервированы для выполнения операций внутри микропроцессора. К таким операциям относятся дешифрация кода команды, необходимые внутренние передачи и преобразования данных, выполнение сдвиговых, арифметических и логических операций.

В соответствии с предложенным разработчиками принципом синхронизации работы МП легко определить число машинных циклов, затрачиваемых на выполнение любой команды. Минимальное количество машинных циклов исполнения команды определяется форматом команды и равняется числу байтов команды. Если собственно выполнение команды требует дополнительных обращений к внешнему устройству, то командный цикл увеличивается на соответствующее число машинных циклов. Например, однобайтная команда пересылки данных из регистра источника ***RD*** в регистр приемник ***RS*** (*MOV RD,RS*) выполняется за один машинный цикл, поскольку собственно исполнение команды не требует дополнительных внешних обращений. Трехбайтная команда вызова подпрограммы *CALL ADDR* выполняется за пять машинных циклов. При ее выполнении к трем машинным циклам выборки команды добавляются два машинных цикла, затрачиваемых для запоминания в стеке 16-битного адреса возврата из подпрограммы.

При реализации всего списка команд МП КР580ВМ80А используются 10 видов машинных циклов:

1. Извлечение кода команды (***M1***).
2. Чтение данных из памяти.
3. Запись данных в память.
4. Извлечение данных из стека.
5. Запись данных в стек.
6. Ввод данных из внешнего устройства.
7. Запись данных во внешнее устройство.
8. Цикл обслуживания прерывания.
9. Останов.
10. Обслуживание прерываний при работе МП БИС в режиме останов.

Тип выполняемого цикла МП указывает на первом такте каждого машинного цикла с помощью 8-разрядного **слова состояния**, выдаваемого на ШД. Значения разрядов слова состояния используются далее для формирования управляющих сигналов, действующих в течение всего текущего машинного цикла. Для сохранения на время выполнения цикла слово состояния записывается в регистр-защелку – регистр слова состояния МП. На рис. 57 приведена схема записи слова состояния.

В первом такте ***T1*** по переднему фронту сигнала *C2* регистр адреса *РА* выставляет на *ША* адрес ячейки памяти или внешнего устройства. Одновременно с появлением адреса на *ША*, МП вырабатывает синхросигнал *SYN*, а на шину данных *ШД* выдает байт слова состояния МП.

В начале такта ***T2*** положительным фронтом импульса *C1* на выходе ГТИ формируется строб *STB*, которым слово состояния

записывается во внешний регистр слова состояния (*Рг.СС*). Положительным фронтом *С2* заканчивается сигнал *SYN* и МП формирует управляющие сигналы текущего машинного цикла.

Назначение отдельных разрядов байта слова состояния МП приведено в табл. 5.

Таблица 5

Назначение отдельных размеров байта слова состояния

Разряды	Обозначение	Назначение сигналов
D0	INTA	Подтверждение прерывания
D1	WO	Указывает, что операция в текущем цикле является операцией записи (при WO = 0) или чтения (при WO = 1)
D2	STACK	Указывает на наличие на шине адреса содержимого стека
D3	HLTA	Подтверждение останова
D4	OUT	Указывает, что в данном цикле на адресной шине установлен номер внешнего устройства и осуществляется вывод содержимого аккумулятора на внешнее устройство
D5	M1	Указывает, что в данном цикле микропроцессор принимает первый байт команды
D6	INP	Указывает, что в данном цикле осуществляется ввод информации из устройства ввода в аккумулятор
D7	MEMR	Указывает, что в данном цикле производится чтение (прием информации из памяти в микропроцессор)

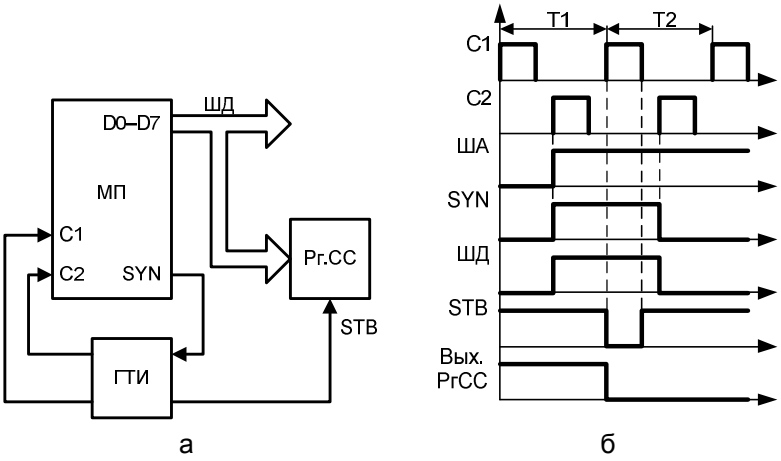


Рис. 57. Фиксация слова состояния микропроцессора

В табл. 6 приведено соответствие сигналов состояния отдельным видам циклов.

Таблица 6

Соответствие сигналов состояния отдельным видам циклов

Вид цикла	Слово состояние МП							
	D7	D6	D5	D4	D3	D2	D1	D0
M1 – Выборка первого байта команды	1	0	1	0	0	0	1	0
M2 – Чтение памяти	1	0	0	0	0	0	1	0
M3 – Запись в память	0	0	0	0	0	0	0	0
M4 – Чтение стека	1	0	0	0	0	1	1	0
M5 – Запись в стек	0	0	0	0	0	1	0	0
M6 – Ввод в УВВ	0	1	0	0	0	0	1	0
M7 – Вывод из УВВ	0	0	0	1	0	0	0	0
M8 – Подтверждение прерывания	0	0	1	0	0	0	1	1
M9 – Подтверждение останова	1	0	0	1	1	0	1	0
M10 – Подтверждение прерывания в режиме останов	0	0	1	1	1	0	1	1

5.5. **Управление микропроцессором и микропроцессорной системой**

Управляющее устройство МП состоит из двух независимых частей:

1. Первичного автомата, управляющего процессами внутри МП (ПУА).

2. Схемы, обрабатывающей осведомительные сигналы и генерирующей управляющие сигналы МПС.

Примерная схема алгоритма функционирования управляющего автомата в течение рабочего цикла выполнения команды приведена на рис. 58. Выполнение рабочего цикла команды начинается с опроса триггера прерывания. Если запрос прерывания поступил и прерывания разрешены (командой *EI*), то автомат формирует машинный цикл обработки прерывания, в котором управление передается подпрограмме обработки прерывания и она выполняется. При отсутствии прерывания управляющий автомат создает цикл выборки команды из памяти и формирует адрес следующей команды.

Далее управляющий автомат дешифрирует код операции в команде и генерирует соответствующую коду операции серию управляющих сигналов, обеспечивающую выполнение в МП заданной операции.

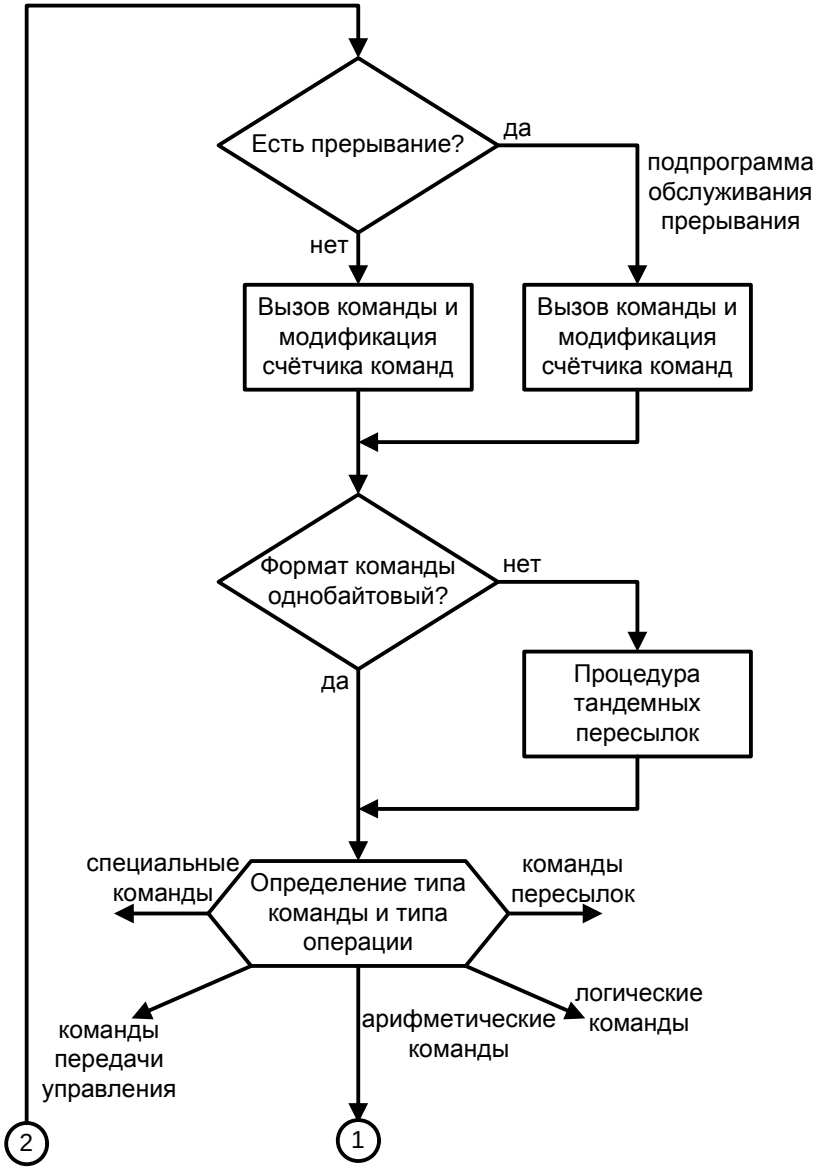


Рис. 58. Рабочий цикл выполнения команды МП КР580ВМ80А

Рис. 58. Окончание (см. также с. 83)

Цикл команды состоит из двух фаз: выборки команды и ее исполнения. Фаза выборки команды одинакова для всех команд микропроцессора. Последняя операция фазы исполнения команды, а именно размещение результата в аккумуляторе или в одном из регистров, также одинакова для целого ряда команд. Классификация команд по типам отражается в самом коде команды, таким образом, существенно упрощается схема декодирования команды.

Алгоритм работы управляющего автомата содержит условный оператор ожидания готовности операнда. Наличие такого оператора в алгоритме позволяет МП приспосабливаться для работы с различными видами внешней памяти, имеющей разное время доступа, а также с медленно действующими устройствами ввода-вывода (УВВ). Наличие в схеме алгоритма устройства управления, оператора ожидания готовности операнда, механизма анализа за-

просов на прерывание и запросов на захват шин, позволяет МП формировать последовательность управляющих сигналов не только на основе команды, но и под воздействием внешних управляющих сигналов *READY*, *INT*, *HOLD*.

Устройство управления МП в зависимости от кода текущей команды, состояния своего управляющего автомата, а также в зависимости от значений сигналов оповещения с шины управления МПС, вырабатывает последовательности сигналов, реализующие процедуры системного обмена информацией.

В МП управляющий автомат в зависимости от сложности команды реализует цикл команды за несколько внутренних машинных циклов. В простейшем случае цикл команды реализуется за 1–5 машинных циклов. Один машинный цикл требуется МП для одного обращения к памяти или устройству ввода/вывода (УВВ). Выборка байта команды или каждого байта адреса или данных (а также их условного представления) требует одного машинного цикла. Аналогичность операций, выполняемых в этих циклах, несмотря на то, что они расположены в различных фрагментах блок-схемы алгоритма работы устройства управления, позволяет реализовывать их в течение цикла команды на одном и том же оборудовании первичного автомата. Эффективность работы управляющего автомата достигается за счет того, что машинные циклы могут быть переменной длины. В МП КР580ВМ80А каждый цикл может состоять из 3–5 тактов. Каждому такту ***T*** соответствует отдельное состояние первичного автомата управляющего устройства МП. На рис. 59 представлена блок-схема алгоритма работы первичного управляющего автомата МП КР580ВМ80А.

Все такты ***T1***–***T5*** имеют одинаковую длительность. Существуют три исключения из этого положения:

1. Состояние *WAIT* (***Tw***), в котором МП находится в ожидании операнда.

2. Состояние *HOLD* (***Twh***), в которое МП переходит под воздействием внешних сигналов управления МП системой.

3. Состояние *HALT*, в которое МП может быть введен командой останова.

Названные состояния МП не связаны с тактовой частотой сигналов синхронизации *C1* и *C2* и их продолжительность не определена, так как зависит от внешних, по отношению к МП, событий. Эти состояния длятся целое число тактов и выход из них МП тактируется. Таким образом, каждый период тактирования МП соответствует особому состоянию первичного автомата. В стандартном машинном цикле может быть от трех до пяти состояний автомата (***T1***–***T3***, ***T1***–***T4***, ***T1***–***T5***). Цикл команды содержит от одного до пяти машинных циклов. В зависимости от сложности операций, определяемых командой, цикл команды может быть реализован первич-

ным автоматом с числом переходов по внутренним состояниям от четырех до восемнадцати.

На предложенной выше блок-схеме первичного автомата (рис. 59) можно отметить следующее:

1. Переход в состояние «захват» выполняется в промежутках между машинными циклами.

2. Микропроцессор позволяет приступить к анализу запросов на прерывание только после окончания выполнения текущей команды.

3. Из состояния останова *HALT* МП может быть выведен двумя способами: поступившим внешним сигналом прерывания и соответствующим разрешением на него или сигналом системной установки в исходное состояние *RESET*, который переводит первичный автомат в состояние ***T1***.

При выполнении стандартного машинного цикла МП всегда безоговорочно выполняет такт ***T1***. Такт ***T2*** – временная задержка. Кроме того, на этом такте осуществляется проверка трёх состояний: *READY* (*ГОТОВНОСТЬ*), *HOLD* (*ЗАПРОС ЗАХВАТА*), *HLTA* (*ПОДТВЕРЖДЕНИЕ ОСТАНОВА*). Если эти сигналы не активны, то МП переходит к такту ***T3***. Такт ***T3*** выполняется безоговорочно.

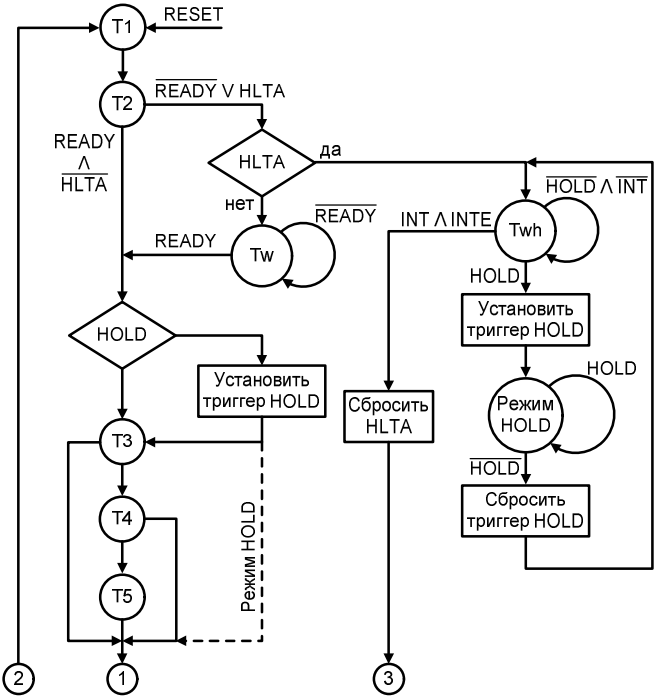


Рис. 59. Алгоритм работы первичного управляющего автомата

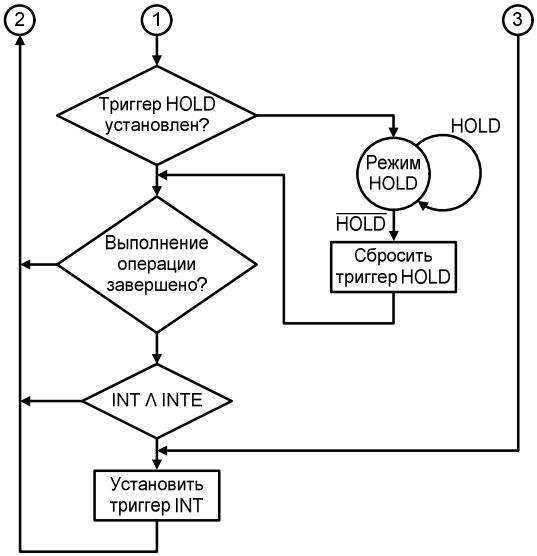


Рис. 59. Окончание (см. также с. 86)

После выполнения такта ***T3*** происходит проверка необходимости выполнения такта ***T4*** – если в данном машинном цикле такт ***T4*** необходим – он выполняется. Если нет, то происходит переход к концу машинного цикла. Аналогично с тактом ***T5***. После окончания машинного цикла (после такта ***T5***) производится проверка состояния внутреннего триггера *HOLD*. В случае отсутствия запроса на захват МП переходит к анализу окончания выполнения команды. Если это не последний машинный цикл текущей команды, то МП переходит к такту ***T1*** следующего машинного цикла этой команды. Если это последний машинный цикл текущей команды, то МП выполняет проверку состояния сигнала *INT* (*ЗАПРОС ПРЕРЫВАНИЯ*). Если прерывание программным путём разрешено, то МП переводится в цикл чтения кода операции команды прерывания. Если выполнение прерывания запрещено, то МП игнорирует сигнал *INT* и переходит к такту ***T1*** цикла ***M1*** следующей команды.

Таким образом, в общем случае распределение машинных тактов можно представить в виде табл. 7.

Пример выполнения некоторых команд иллюстрируется на рис. 60. В арифметической команде сложения содержимого регистра с содержимым аккумулятора *ADD S* (рис. 60, а) в единственном цикле *ВЫБОРКА* вначале осуществляется выборка байта кода операции, для чего в такте ***T1*** на адресную шину выдается содержимое программного счетчика ***PC***, которое затем увеличивается на 1 в так-

те ***T2*** для адресации следующей команды. Команда извлекается в такте ***T3*** и по ней в такте ***T4*** осуществляется подготовка операндов к выполнению операции сложения – операнд из регистра (***S***) пересылается по внутренней шине в регистр (*БР1*), а операнд из аккумулятора пересылается в регистр (*БР2*). В пятом такте, который выполняется в течение действия такта ***T2*** следующей команды, осуществляется сложение операндов с сохранением результата в аккумуляторе. Совмещение тактов позволяет повысить производительность процессора.

Таблица 7.

Распределение действий по тактам процессора КР580ВМ80А

Такт	Действие
T1	Адрес памяти или устройства ввода/вывода выдается на адресную шину (A15...A0), информация о состоянии выдается на шину данных (D7...D0)
T2	Осуществляется анализ сигналов на входах *READY* и *HOLD*, а также контроль команды останова *HLT*
Tw	Если сигнал на входе *READY* имеет низкий уровень или на входе *HOLD* высокий уровень или выполнится команда *HLT*, процессор переходит в одно из состояний (*ОЖИДАНИЕ*, *ЗАХВАТ* или *ОСТАНОВ*)
T3	С шины данных в процессор вводится байт команды (в цикле *ВЫБОРКА*), байт данных (в циклах *ЧТЕНИЕ ПАМЯТИ*, *ЧТЕНИЕ СТЕКА*, *ВВОД*) или команда прерывания *RST N* (цикл *ПРЕРЫВАНИЕ*), или из процессора выводится байт данных (цикл *ЗАПИСЬ В ПАМЯТЬ*, *ЗАПИСЬ В СТЕК*, *ВЫВОД*) на шину данных
T4	Выполняется операция. Такты ***T4***, ***T5*** оба вместе или один из них (***T5***) могут не использоваться при выполнении команды
T5	

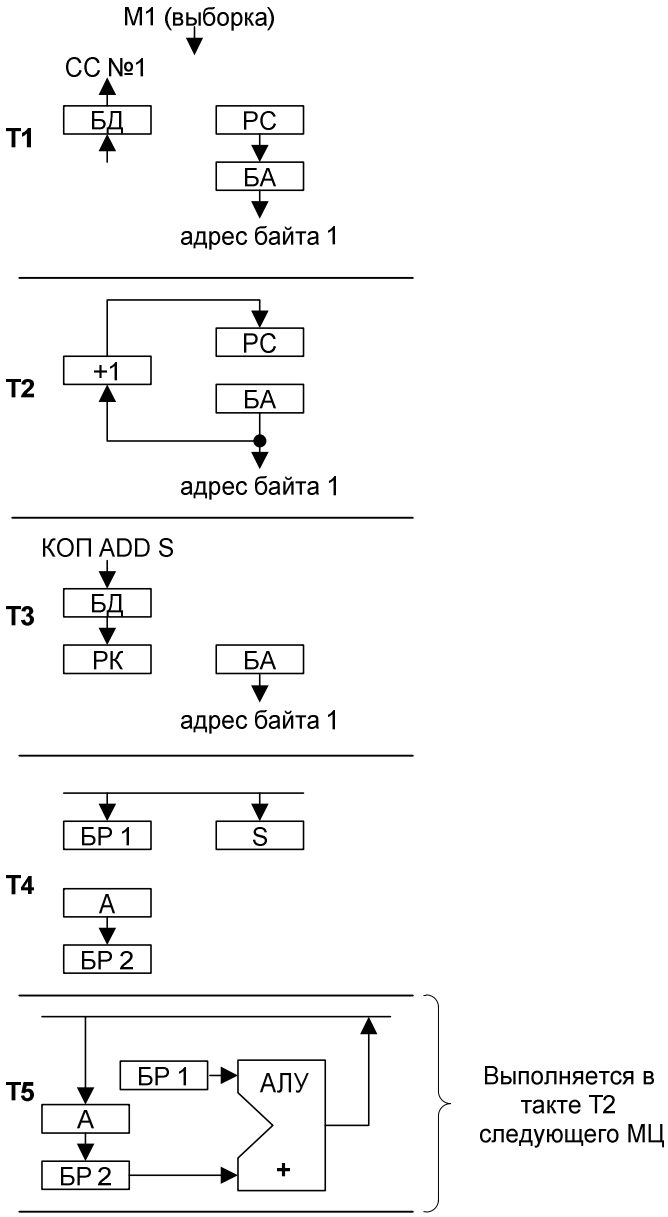
В команде сложения содержимого ячейки памяти с содержимым аккумулятора *ADD M* (рис. 60,б) цикл *ВЫБОРКА*, состоящий из тактов ***T1*** – ***T4***, выполняется почти аналогично, а цикл ***M2*** используется для выполнения операции сложения. Для выборки операнда из памяти в такте ***T1*** на адресную шину выдается его адрес из пары (***HL***), по которому в такте ***T3*** операнд выбирается в регистр (*БР1*). В четвертом такте, которой также выполняется в течение такта ***T2*** следующей команды, осуществляется сложение операндов.

В командах ввода/вывода (*IN*, *OUT*) (рис. 60,в) для выборки используются два цикла, в каждом из которых на адресную шину выдается содержимое программного счетчика, увеличиваемого в ***T2*** на единицу. Первый байт выбирается в регистр команд, второй – в регистры (***W***), (***Z***). В третьих циклах команд – циклах ввода/вывода – на адресную шину выдается адрес порта (параллельно на младшие и старшие разряды шины), по которому затем осуществляется обмен с адресуемым портом.

В команде обращения к подпрограмме *CALL ADR* (рис. 60,г) команда выбирается в течение трех машинных циклов по адресам, формируемым в программном счетчике: в первом цикле в регистр (*PK*) выбирается байт кода операции, во втором – младшие разряды адреса (второй байт команды) в регистр (***Z***), в третьем – старшие разряды адреса (третий байт команды) в регистр (***W***). Таким образом, адрес подпрограммы хранится в паре (***WZ***). В тактах ***T4***, ***T5*** цикла ***M1*** осуществляется подготовка адреса стека – уменьшение содержимого (***SP***) на единицу. По этому адресу в цикле ***M4*** в ячейку стека, адресуемую содержимым (***SP***), записывается старший байт адреса возврата, и адрес стека уменьшается на 1. Затем в цикле ***M5*** в ячейку стека записывается младший байт адреса возврата. Адрес перехода на подпрограмму заносится в счётчик команд (***PC***) в цикле ***M1*** следующей команды с последующим увеличением его на 1.

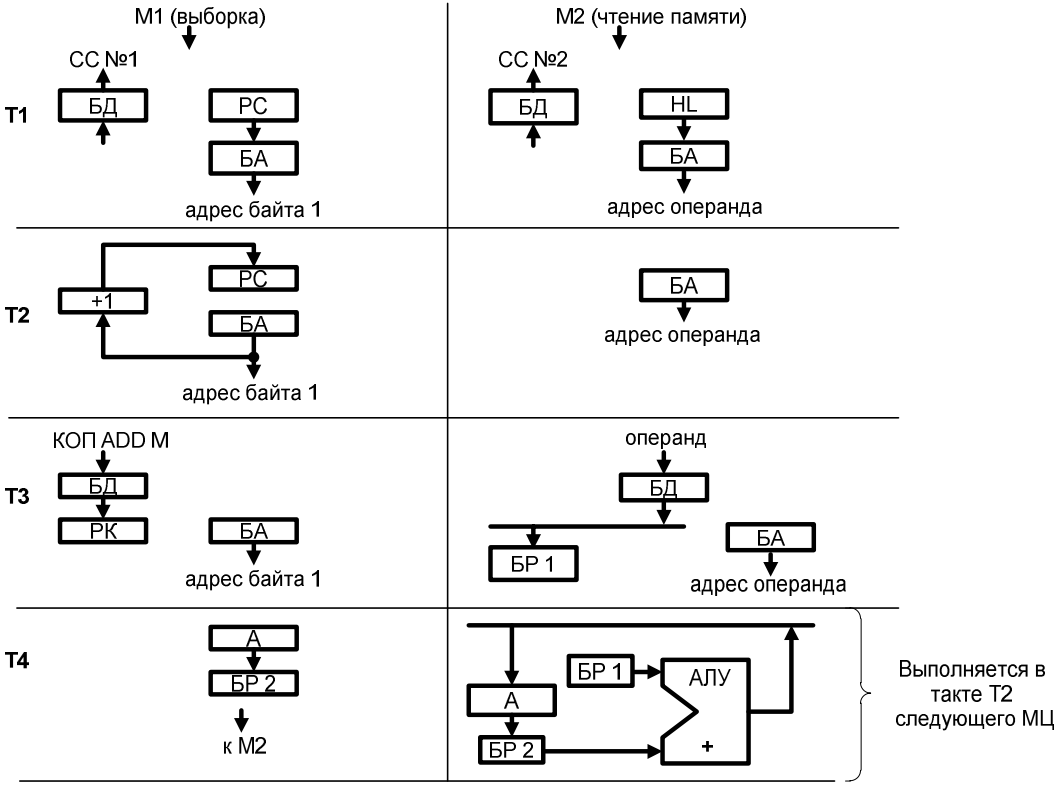
Временная диаграмма типичного машинного цикла показана на рис. 61. Длительность такта в МП КР580ВМ80А задается периодом синхроимпульсов фазы *C1*, внутренние микрооперации инициируются импульсами из последовательности *C2*. Начало каждого машинного цикла (такт ***T1***) отмечается появлением сигнала на выходе *SYN* (начало связано с фронтом импульса *C2* с некоторой задержкой, а конец – с фронтом следующего импульса *C2*). В течение этого интервала выдается информация о состоянии процессора (текущее слово состояния МП).

Фронт импульса *C2* в такте ***T1*** также вызывает загрузку адреса на адресную шину. Адрес сохраняется до появления фронта импульса *C2* после такта ***T3***. Выдав адрес на адресную шину, процессор в такте ***T2*** осуществляет проверку сигналов подтверждения состояний *ОЖИДАНИЕ*, *ЗАХВАТ*, *ОСТАНОВ*. Если, например, память не готова к обмену, на линии *RDY* выдается сигнал низкого уровня. При этом процессор переходит в состояние *ОЖИДАНИЕ*, выдает сигнал высокого уровня на линию *WI*, подтверждающий запрос памяти (переключение сигнала на линии *WI* осуществляется фронтом импульса *C1*).



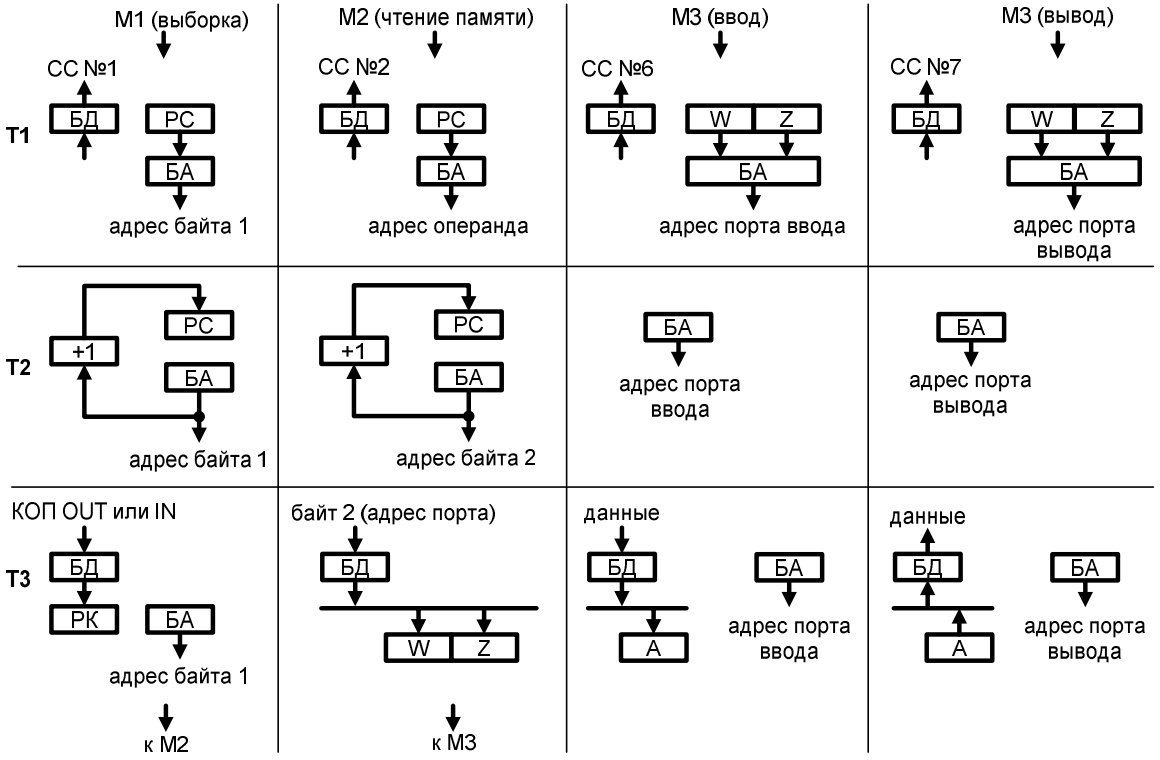
а

Рис. 60 а. Схема выполнения команд *ADD S*



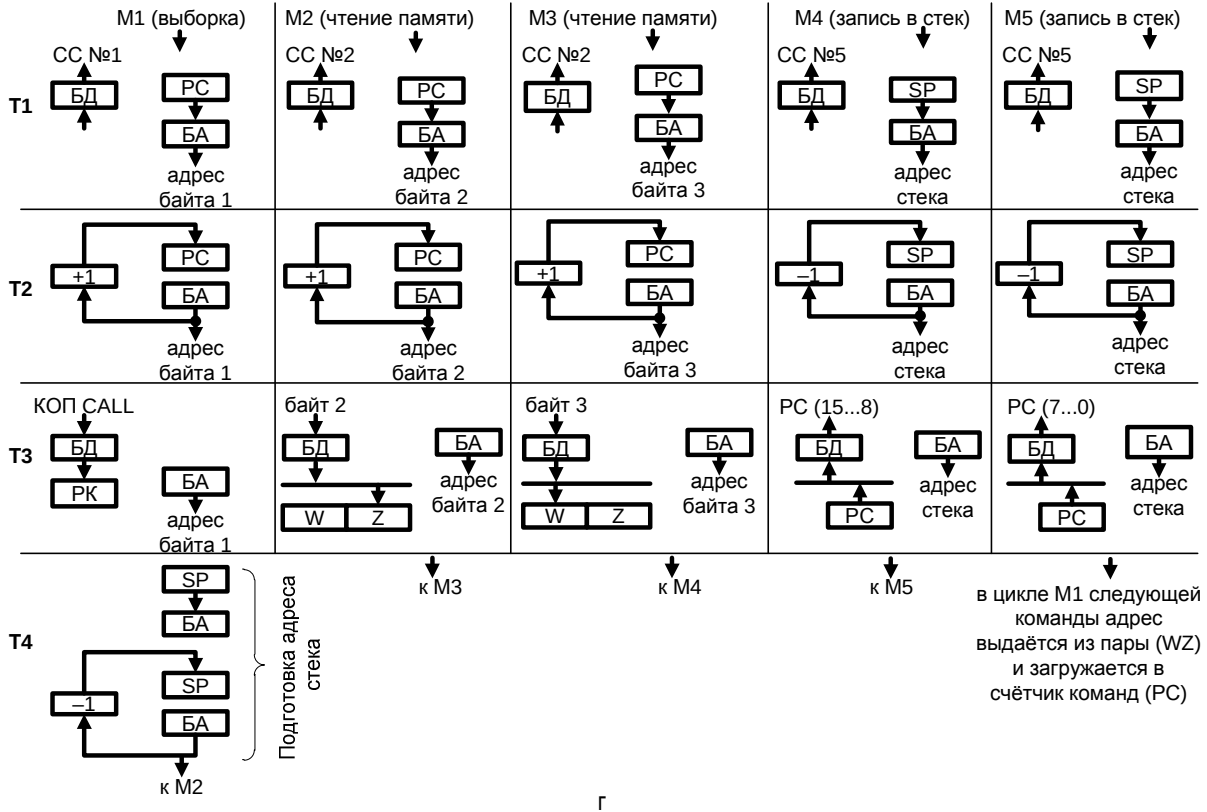
б

Рис. 60 б. Схема выполнения команд *ADD M*



в

Рис. 60 в. Схема выполнения команд *IN* / *OUT*



г

Рис. 60 г. Схема выполнения команд *CALL*

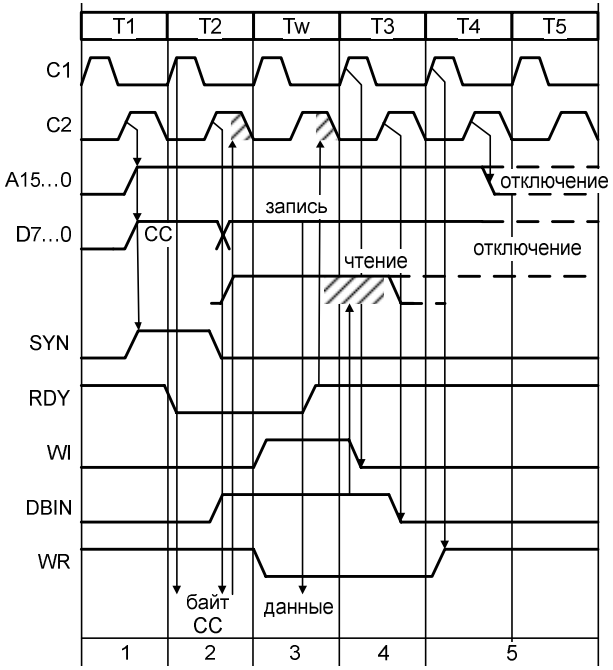


Рис. 61. Временные диаграммы типового машинного цикла МП КР580ВМ80А:
1 – A15–0 адрес памяти или номер ВУ D7–0 байт слова состояния МП;
2 – анализ состояний: «готовность», «захват», «останов.»; 3 – останов. или синхронизация с памятью; 4 – выборка данных команд или запись данных; 5 – выполнение команды

Процессор остается в состоянии *ОЖИДАНИЕ* до тех пор, пока сигнал на линии *RDY* не переходит в состояние высокого уровня. Если информация о готовности предшествует спаду импульса *C2*, то следующий импульс *C1* переводит линию *WI* в состояние низкого потенциала и процессор переходит в состояние ***T3***. В этом состоянии осуществляется обмен информацией с памятью или внешним устройством через шину данных.

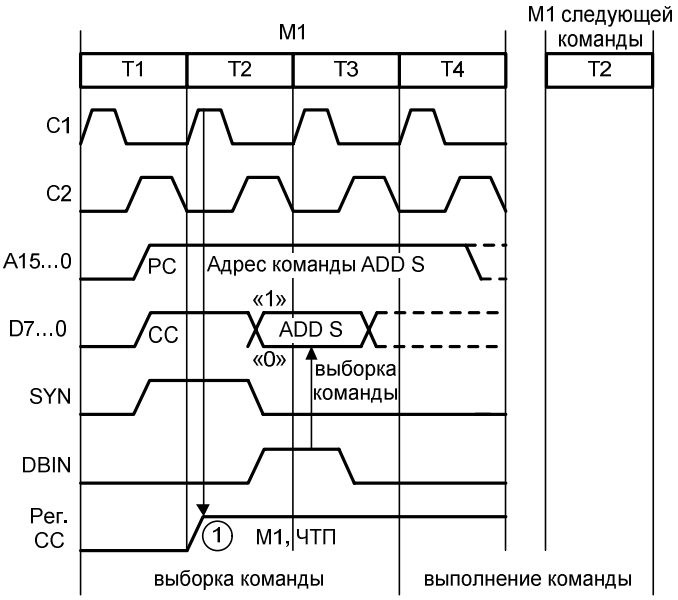
При вводе данных в процессор по фронту импульса *C2* в такте ***T2*** снимается информация о состоянии и шина готова к приему данных. Данные, вводимые в процессор, должны быть стабильными в течение интервала времени, начало которого предшествует спаду импульса *C1* или фронту импульса *C2* в такте ***T3***, а конец следует за фронтом импульса *C2*. При этом процессор формирует сигнал высокого уровня на линии *DBIN* (сигнал формируется в пределах действия фронтов импульсов *C2* в тактах ***T2*** и ***T3***). Длительность сигнала зависит от длительности такта ожидания ***Tw***, находящегося между тактами ***T2***, ***T3***.

При выводе данные появляются на шине по фронту импульса *C2* (одновременно информация о состоянии снимается) и выдаются на шину до появления новой информации о состоянии. При этом процессор формирует сигнал низкого уровня на линии *WR*. Сигнал появляется по фронту импульса *C1*, следующего за тактом ***T2*** (это могут быть такты ***T3*** или ***Tw***), а оканчивается по фронту *C1* в такте, следующем за ***T3***. Таким образом, его длительность также зависит от длительности такта ожидания ***Tw***.

Все машинные циклы содержат такты ***T1***, ***T2***, ***T3***. Такты ***T4*** и ***T5*** используются для выполнения команды. Завершив цикл ***M1***, процессор переходит к циклу ***M2*** выполняемой команды или к циклу ***M1*** следующей команды, если команда одноцикловая.

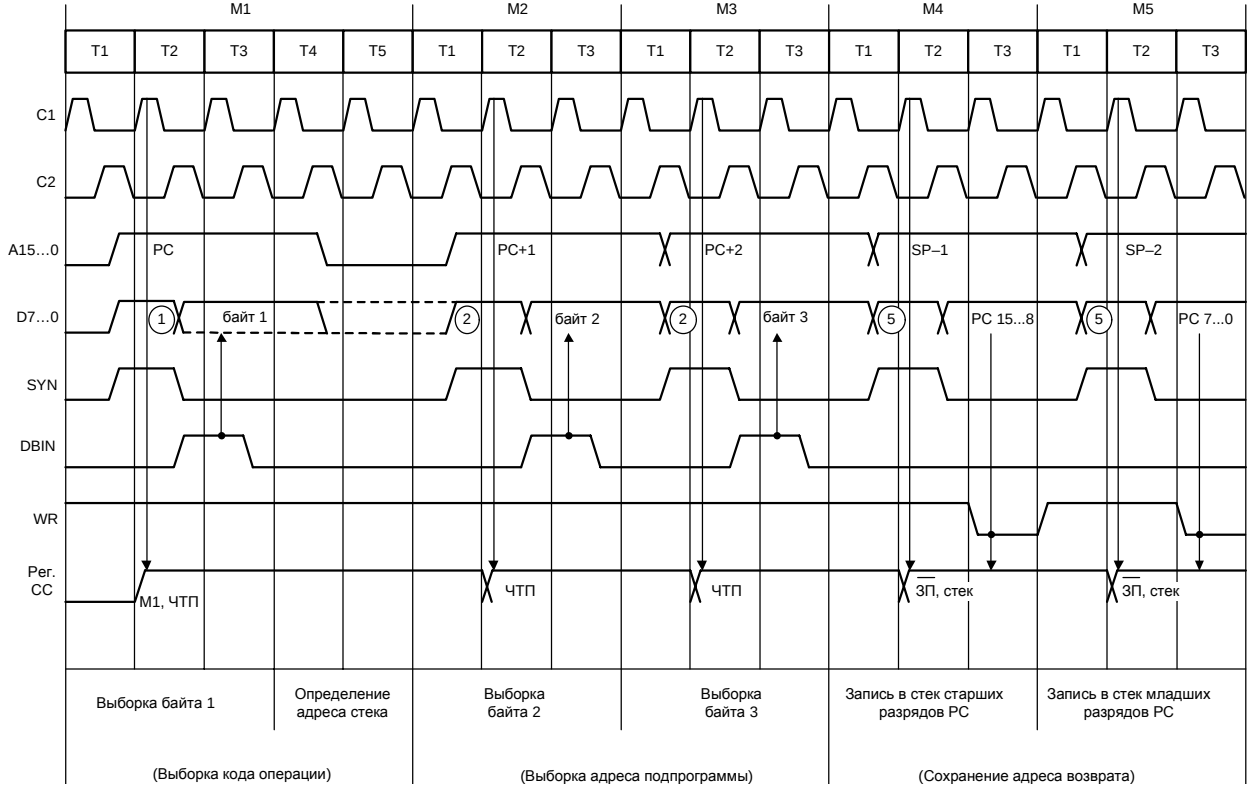
В последнем такте последнего машинного цикла проверяется наличие запроса прерывания и, если он имеется, ЦП переходит к выполнению цикла ***M1*** специального типа. В течение этого цикла содержимое программного счетчика не увеличивается на 1, а выдается сигнал подтверждения прерывания. Из устройства, вызвавшего прерывание, должна быть послана команда *RST*, в которой указывается вектор прерывания – адрес подпрограммы, обслуживающей прерывание.

Примеры временных диаграмм выполнения некоторых команд показаны на рис. 62.



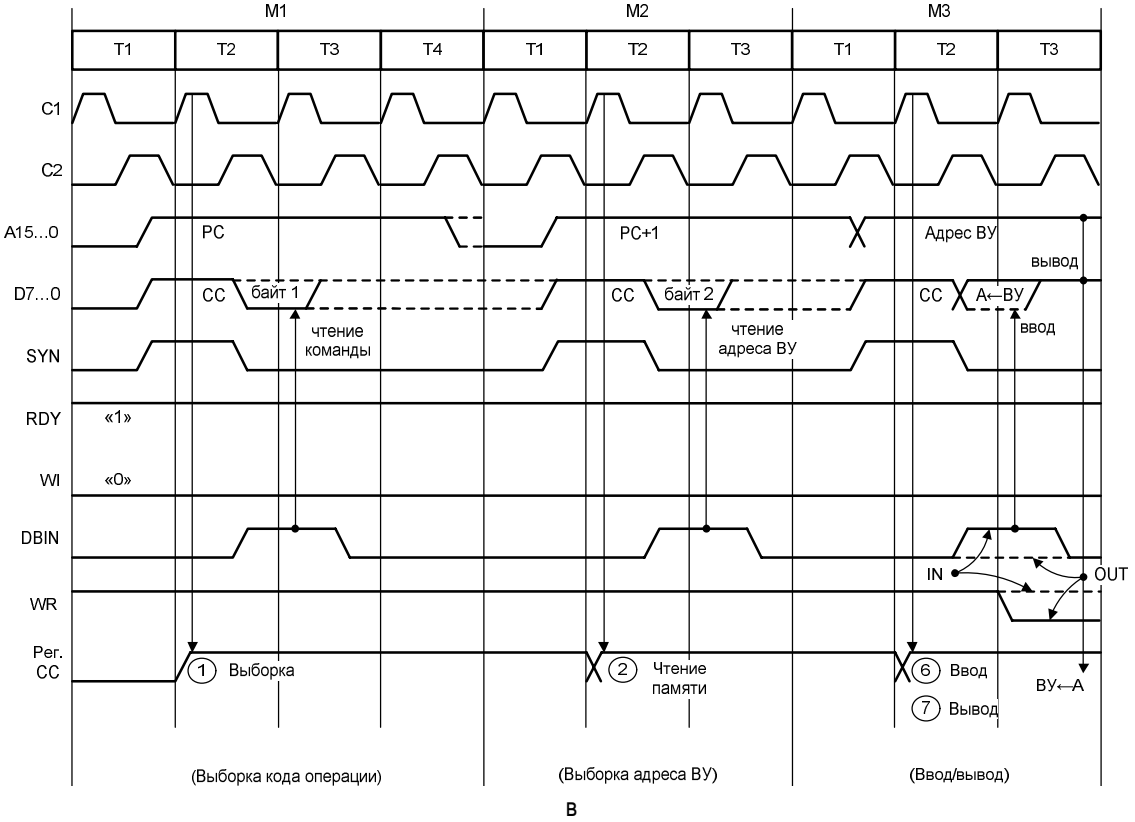
а

Рис. 62 а. Временные диаграммы выполнения команды *ADD S*



б

Рис. 62 б. Временные диаграммы выполнения команды *CALL*



в

Рис. 62 в. Временные диаграммы выполнения команд *IN* / *OUT*

5.6. **Управление режимами работы МП КР**580**ВМ**80**А**

Микропроцессор КР580ВМ80А может работать в режимах прерывания, ожидания, захвата шин при прямом доступе к памяти внешних устройств (ВУ), инициируемых внешними сигналами управления, и в режиме останова, переход к которому осуществляется по программе. Кроме указанных существует ещё один специализированный режим начальной установки МП по сигналу *RESET*.

Режим ожидания

Переход в режим *ОЖИДАНИЕ* происходит при пассивном состоянии сигнала готовности *READY (RDY* = 0) при выполнении любого из циклов МП (кроме ***M8*** – подтверждение прерывания, ***M9*** – подтверждение останова, ***M10*** – подтверждение прерывания во время останова).В этом случае МП приступает к выполнению целого числа тактов ожидания. В конце каждого такта ожидания МП проверяет состояние сигнала *READY*. Если *RDY* = 1, то МП возвращается в основную цепочку алгоритма: проверяет состояние сигнала *HOLD* и приступает к выполнению третьего такта (см. рис. 59).

Режим ожидания присущ всем МП. В ряде случаев быстродействие МП оказывается несогласованным с быстродействием внешнего устройства (ВУ). Например, быстродействие интегральных схем (ИС) памяти или устройства цифропечати существенно ниже быстродействия МП. В этих случаях необходимо приостановить действие МП, т.е. «растянуть» машинный цикл на целое количество тактов.

Аналогичная ситуация может возникнуть при реализации поциклового и покомандного режимов работы МП. Необходимость работы МП в этих режимах возникает при отладке программ, когда после выполнения каждого машинного цикла или после выполнения команды требуется переводить МП в режим ожидания.

Режим ожидания МП реализуется путем подачи на вход *RDY* микропроцессора сигнала низкого уровня. Чтобы МП отреагировал на низкий уровень сигнала *RDY* в текущем машинном цикле, этот уровень должен стабилизироваться минимум за 180 нс до спадающего фронта сигнала *C2* и сохраняться до его окончания. При соблюдении этого условия МП не переходит к такту ***T3***, а входит в состояние ожидания, которое может сохраняться как угодно долго. В состоянии ожидания на шине адреса сохраняется выданный в такте ***T1*** адрес ячейки памяти (или любого регистра, ВУ) и сигнал *DBIN* – готовность к приёму данных, если текущий цикл связан с чтением данных в МП. Перейдя в режим ожидания, МП информирует ВУ о своем состоянии сигналом *WI* – ожидание, который формируется по нарастающему фронту сигнала *C1* (рис. 63).

Продолжительность режима ожидания определяется моментом подачи на вход *RDY* сигнала высокого уровня от ВУ системы. Чтобы МП отреагировал на готовность ВУ, высокий уровень сигнала *RDY* должен стабилизироваться не менее чем за 180 нс до спадающего фронта *C2*. После этого МП переходит к такту ***T3*** и по нарастающему фронту снимает сигнал *WI*.

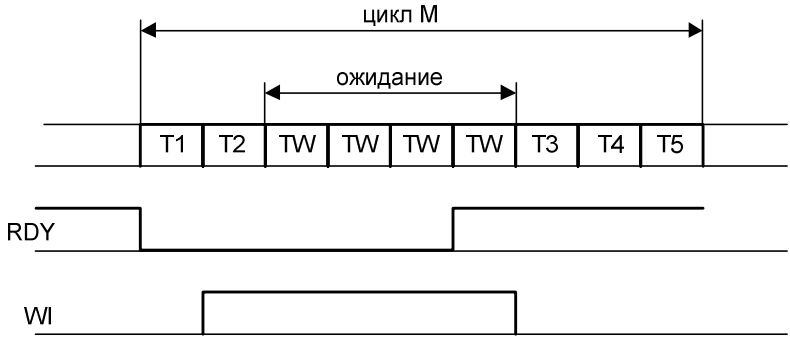


Рис. 63. Формирование режима ожидания МП КР580ВМ80А

Использование режима ожидания снижает производительность системы, но обеспечивает надежное согласование работы МП и ВУ с различным быстродействием.

В структуре МП КР580ВМ80А не предусмотрено ограничение по длительности тактов ожидания, т.е. при аварийной установке *RDY* = 0 происходит зависание МП. В современных МП число тактов ожидания лимитировано и если этот предел превышается, то МП переходит к внутреннему прерыванию по ошибке обращения к несуществующему устройству.

Режим начальной установки

Режим начальной установки используется при включении микропроцессора (МП). В режиме начальной установки счетчик команд ***PC*** микропроцессора обнуляется, а на шине адреса *ША*, следовательно, выставляется шестнадцатеричный адрес ячейки памяти *0000h*. Дальнейшие действия МП, как известно, сводятся к реализации цикла ***M1*** (чтение кода операции). Следовательно, в «нулевой» ячейке памяти должна быть записана любая команда, с которой начинается выполнение программы. Чаще всего в эту ячейку записывают команду безусловного перехода к области памяти, где располагается управляющая программа системы. Таким образом, режим начальной установки обеспечивает запуск МП.

Практически режим начальной установки реализуется подачей на вход *SR* (*RESET*) МП сигнала низкого уровня, который мо-

жет быть сформирован либо в момент включения питания, либо нажатием кнопки сброса, подключенной ко входу RESIN тактового генератора (см. рис. 23). Длительность этого сигнала должна быть не менее трех периодов синхроимпульсов. В момент включения источника питания +5 В конденсатор *C* разряжается, напряжение на входе *RESIN* ГТИ равно нулю, что позволяет ГТИ сформировать на своем входе сигнал *SR* низкого уровня. После заряда конденсатора *C* током через резистор *R* сигнал *SR* становится равным 1, и МП начинает выполнять цикл ***M1*** первой команды программы.

При нажатии на кнопку конденсатор *C* разряжается и на выходе ГТИ возникает сигнал сброса *SR*. При размыкании кнопки процедура зарядки конденсатора повторится, что приведет к снятию сигнала *SR*. По сигналу *SR* в МП сбрасывается триггер прерывания, а для его установки необходимо выполнить команду *EI*. В противном случае запросы на прерывания игнорируются.

Если запуск программы в системе осуществляется по определенному стартовому сигналу, в первых двух ячейках необходимо размещать команды перевода процессора в состояние останова (*EI* и *HLT*), а для запуска использовать ручное или автоматическое прерывание.

Содержимое рабочих регистров, в частности, регистра флагов, при этом остается неопределенным до тех пор, пока программа не установит его. Поэтому необходимо принять меры по исключению возможных ошибок.

Способы обмена информацией в микропроцессорной системе

В микропроцессорной системе (МПС) применяются три режима ввода/вывода (ВВ): программно-управляемый ВВ (называемый также программным или неформированным ВВ), ВВ по прерываниям (форсированный ВВ) и прямой доступ к памяти. Первый из них характеризуется тем, что инициирование и управление ВВ осуществляется программой, выполняемой процессором, а внешние устройства играют сравнительно пассивную роль и сигнализируют только о своем состоянии, в частности, о готовности к операциям ввода/вывода. Во втором режиме ВВ 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 ВВ управляет процессор.

Наконец, в режиме прямого доступа к памяти, который используется, когда пропускной способности процессора недостаточно,

действия процессора приостанавливаются, он отключается от системной шины и не участвует в передачах данных между основной памятью и быстродействующим ВУ. Заметим, что во всех вышеуказанных случаях основные действия, выполняемые на системной магистрали МПС, подчиняются двум основным принципам.

1. В процессе взаимодействия любых двух устройств ЭВМ одно из них обязательно выполняет активную, управляющую роль и является задатчиком, второе оказывается управляемым, исполнителем. Чаще всего задатчиком является процессор.

2. Другим важным принципом, заложенным в структуру интерфейса, является принцип квитирования (запроса – ответа): каждый управляющий сигнал, посланный задатчиком, подтверждается сигналом исполнителя. При отсутствии ответного сигнала исполнителя в течение заданного интервала времени формируется так называемый тайм-аут, задатчик фиксирует ошибку обмена и прекращает данную операцию.

Программно-управляемый ввод/вывод

Данный режим характеризуется тем, что все действия по вводу/выводу реализуются командами прикладной программы. Наиболее простыми эти действия оказываются для «всегда готовых» внешних устройств, например, индикатора на светодиодах. При необходимости ВВ в соответствующем месте программы используются команды *IN* или *OUT*. Такая передача данных называется синхронным или безусловным ВВ.

Однако для большинства ВУ до выполнения операций ВВ надо убедиться в их готовности к обмену, т.е. ВВ является асинхронным. Общее состояние устройства характеризуется флагом готовности *READY*, называемым также флагом готовности/занятости (*READY/BUSY*). Иногда состояния готовности и занятости идентифицируются отдельными флагами *READY* и *BUSY*, входящими в слово состояния устройства.

Процессор проверяет флаг готовности с помощью одной или нескольких команд. Если флаг установлен, то инициируются собственно ввод или вывод одного или нескольких слов данных. 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 и реализуется в различных процессорах по-разному.

Основной недостаток программного ВВ связан с непроизводительными потерями времени процессора в циклах ожидания. К до-

стоинствам следует отнести простоту его реализации, не требующей дополнительных аппаратных средств.

Режим прерывание

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. 64).

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 микропроцессорной системы яв-

ляются уменьшение числа команд, обеспечивающих сохранение информации о прерванной программе, и реализация этих функций аппаратными средствами.

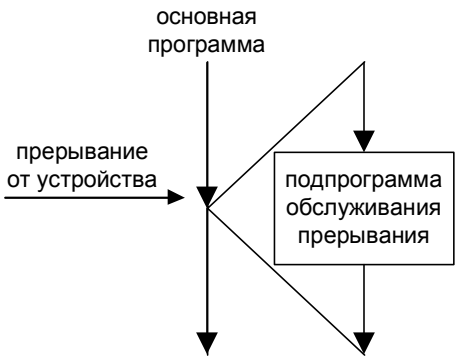


Рис. 64. Реакция на прерывание основной программы

Прерывать можно не только фоновую программу, но и подпрограмму обслуживания прерывания (многоуровневые или вложенные прерывания). Каждая прерванная программа, после завершения прервавшей ее программы, продолжает работу так, как будто прерывания не было (рис. 65).

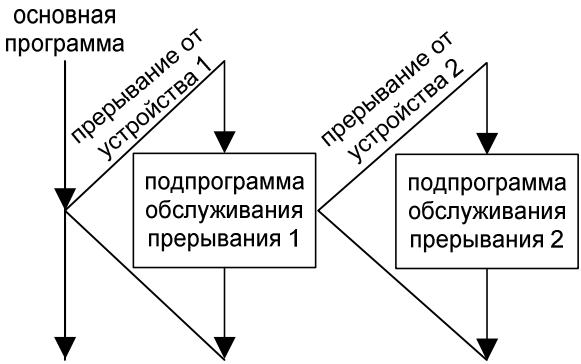


Рис. 65. Многоуровневые (вложенные прерывания)

В микропроцессоре КР580ВМ80А имеются средства обработки векторных запросов прерываний восьми уровней. Источник прерываний должен сформировать запрос прерывания, сопровождаемый вектором в форме команды *RST*. Последовательность перехода в режим прерывания представлена на рис. 66.

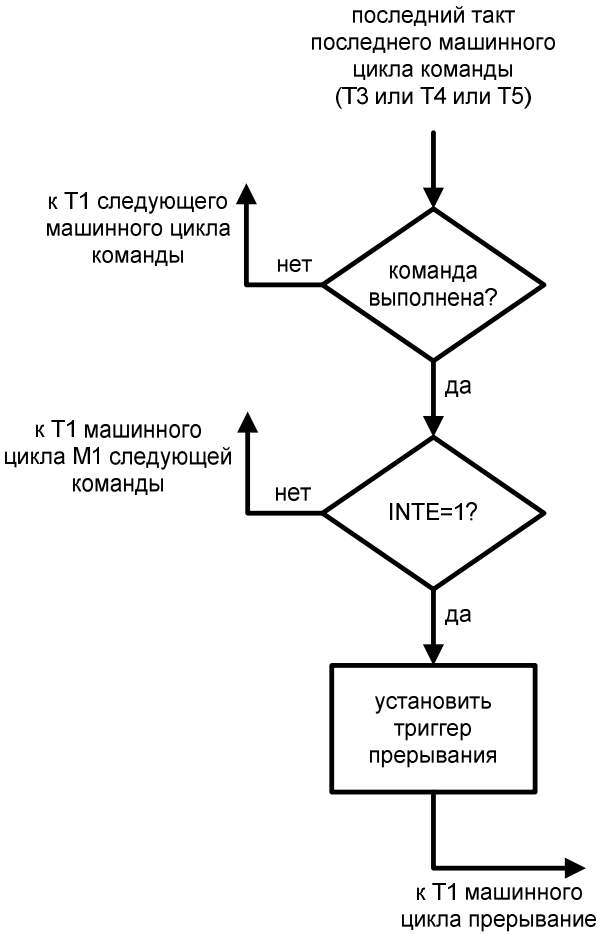


Рис. 66. Последовательность перехода в режим прерывание

Последовательность действий микропроцессора в режиме прерываний следующая:

– прием запроса прерывания и блокировка входа запроса прерывания;

– прием команды *RST*;

– сохранение адреса возврата (содержимого программного счетчика) в стеке и формирование адреса подпрограммы обслуживания источника запроса.

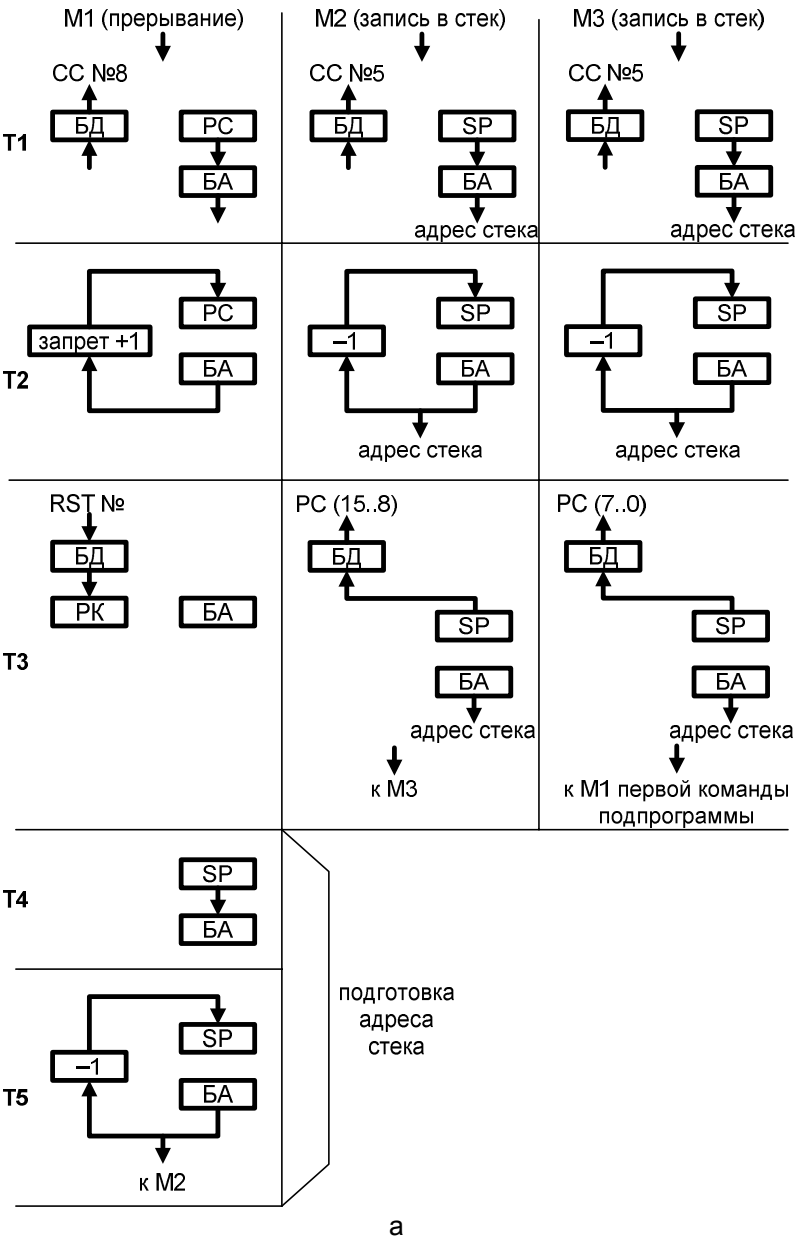
Запросы прерывания микропроцессора КР580ВМ80А принимаются с входа *INT* триггером прерываний, который управляется триггером разрешения прерываний.

Внешнее устройство, запрашивающее прерывание, устанавливает на линии *INT* сигнал высокого уровня. Сигнал этот может возникнуть в произвольный момент выполнения программной команды, однако прием его синхронизируется внутренними цепями процессора.

При наличии сигнала разрешения прерывания (триггер разрешения прерываний установлен в единичное состояние и на выходе *INTE* имеется сигнал высокого уровня) триггер прерывания устанавливается в последнем такте последнего машинного цикла команды, во время выполнения которой возник запрос. Это дает возможность процессору завершить выполнение команды, перед тем как начнется обработка прерывания.

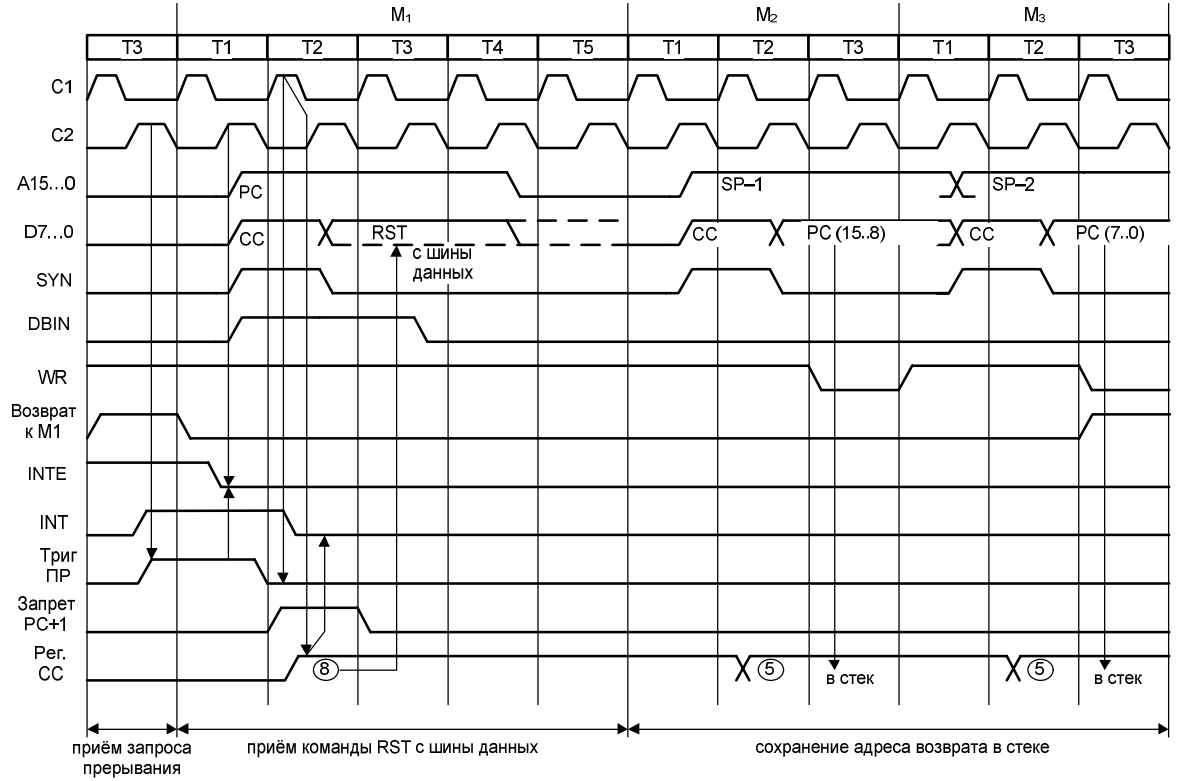
После восприятия сигнала запроса прерывания триггер разрешения прерываний сбрасывается, а процессор переходит к выполнению последовательности, состоящей из трех машинных циклов (рис. 67), первый из которых (цикл *ПРЕРЫВАНИЕ*) предназначен для приема команды *RST*, а два других – для сохранения адреса возврата в стек (циклы *ЗАПИСЬ В СТЕК*). В цикле ПРЕРЫВАНИЕ, который во многом подобен циклу *ВЫБОРКА* (так как в байте состояния установлен бит ***D5***), в первом такте ***T1*** в байте состояния дополнительно формируется сигнал подтверждения прерывания (бит слова состояния ***D0***), который используется для управления чтением команды *RST*, а бит слова состояния ***D7*** (считывание из памяти) – сбрасывается. Содержимое программного счетчика при этом хотя и выдается на адресную шину, но не используется для адресации и при этом не увеличивается на 1 с тем, чтобы сохранить его в дальнейшем в стеке. В том же такте (***T1***) по нарастающему фронту сигнала *C2* с максимальной задержкой 200 нс формируется низкий уровень на выводе *INTE*. Следовательно, МП будет игнорировать последующие запросы на прерывание до тех пор, пока триггер разрешения прерываний не будет установлен командой *EI*.

В такте ***T2*** генерируется сигнал считывания *DBIN*, который в обычном цикле ***M1*** вводит код операции из программной памяти в регистр команд. Но в цикле ***M8*** обращение к программной памяти запрещено (бит ***D7*** равен 0), поэтому код операции должен быть сформирован подсистемой прерываний. Кроме того, в такте ***T2*** сбрасывается внутренний триггер прерывания. В такте ***T3*** процессор принимает по шине данных байт команды *RST*, формируемый внешним устройством, запросившим прерывание. В тактах ***T4***, ***T5*** цикла *ПРЕРЫВАНИЕ* осуществляется формирование адреса первой ячейки стека, отведенной для сохранения адреса возврата.



а

Рис. 67. Схема выполнения (а) и временные диаграммы (б) последовательности перехода в режим прерывания



б

Рис. 67. Окончание (см. также с 106)

В циклах ***M2*** и ***M3*** происходит загрузка адреса возврата (содержимого программного счетчика) в стек, как и при выполнении команды вызова подпрограммы *CALL ADR*. В следующем цикле извлекается первая команда подпрограммы обслуживания прерывания по адресу, указываемому вектором прерывания в команде *RST*.

Последующее управление микропроцессором возлагается на спецпрограмму. В частности, с помощью подпрограммы осуществляются сохранение содержимого основных рабочих регистров процессора, управление триггером разрешения прерывания (маскирование прерываний), восстановление содержимого основных рабочих регистров и возврат к основной программе (восстановление содержимого программного счетчика).

В микропроцессорном комплекте КР580 механизм управления прерываниями реализован в специальной БИС КР580ВН59 (рис. 68), 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 (незамаскированы), формирует сигнал подтверждения прерывания *INTA* (*INTERRUPT ACKNOWLEDGEMENT*).

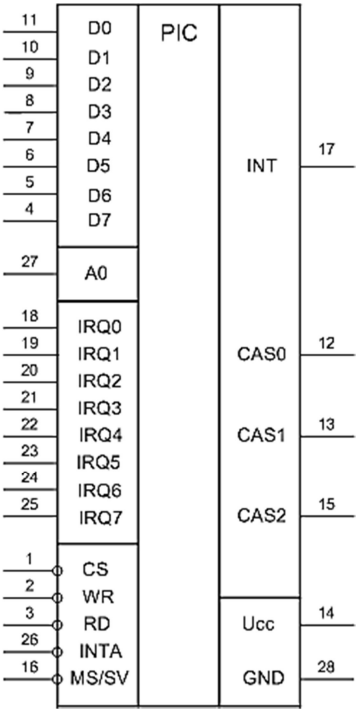
3. Запоминается содержимое счетчика команд ***PC*** и некоторых других внутренних регистров в стеке (в КР580 автоматически сохраняется только адрес возврата).

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

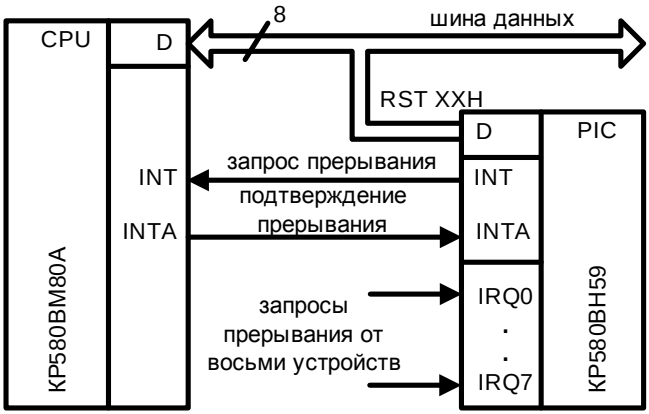
5. Выполняется короткая (30–50 байт) подпрограмма обслуживания прерывания, в которой запрограммированы действия по передаче данных, модификации указателей, проверке окончания операций ввода/вывода.

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.



а



б

Рис. 68. ИМС контроллера прерываний КР580ВН59 (а) и организация ввода/вывода в режиме пребывания (б)

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 микропроцессорной системе 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 *11100111*), которой соответствует стартовый адрес *0020h* (*0000 0000 00NN N000*). Число *NNN* также определяет максимальное количество внешних устройств, которое для КР580ВН59 равно восьми.

Последней командой обработки прерываний должна быть команда возврата *RET*, которая возвращает в ***PC*** адрес возврата в основную программу. Каждая подпрограмма обслуживания прерывания должна содержать команду *EI*, чтобы, выйдя из подпрограммы прерывания, МП был готов к восприятию следующего запроса на прерывание.

Назначение выводов ИМС КР580ВН59 (см. рис. 68, а):

CS – вход выбора микросхемы.

WR – вход записи данных.

RD – вход чтения данных.

D7–D0 – входы/выходы шины данных.
CAS2–CAS0 – входы/выходы шины каскадирования.
GND – общий.
MS/SV – вход выбора ведущей/ведомой микросхемы.
INT – выход запроса прерывания.
IRQ7–IRQ0 – входы запросов прерываний от внешних устройств.
INTA – вход подтверждения прерывания.
A0 – 0 разряд шины адреса.
Ucc – напряжение питания.

Режим прямого доступа к памяти (режим захват)

Многие микропроцессорные системы имеют в своем составе ВУ с высокой скоростью передачи больших массивов информации. В этом случае обмен данными с ВУ организуется в режиме прямого доступа к памяти (ПДП). Суть режима заключается в том, чтобы осуществить обмен информацией между ВУ и памятью МПС, минуя микропроцессор. 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.

Для реализации режима прямого доступа к памяти необходимо обеспечить непосредственную связь контроллера ПДП и памяти МПС. Для этой цели можно было бы использовать специально выделенные шины адреса и данных, связывающие контроллер ПДП с памятью. Но такое решение нельзя признать оптимальным, так как это приведет к значительному усложнению МПС в целом, особенно при подключении нескольких ВУ. В целях сокращения количества линий в шинах МПС контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса процессором и контроллером ПДП. Можно выделить два основных способа ее решения: реализация обмена в режиме «прозрачного ПДП» и в режиме «ПДП с приостановкой процессора».

В режиме «прозрачного ПДП» передача данных выполняется без информирования процессора (наиболее простой способ организации ПДП), для чего для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью, а выполняет внутренние преобразования данных. В такие циклы контроллер ПДП может обмениваться данными с памятью, не мешая работе процессора. Такими интервалами в процессоре КР580ВМ80А являются такты ***T4*** и ***T5*** (части цикла, используемые для внутренних операций). Однако возникает необходимость выделения таких циклов, чтобы не произошло временного перекры-

тия обмена ПДП с операциями обмена, инициируемыми процессором. В некоторых процессорах (например КР580ВМ80А) формируется специальный управляющий сигнал, указывающий циклы, в которых процессор не обращается к системному интерфейсу. При использовании других процессоров для выделения таких циклов необходимо применение в контроллерах ПДП специальных селектирующих схем, что усложняет их конструкцию. Применение рассмотренного способа организации ПДП не снижает производительности МПС, но при этом обмен в режиме ПДП возможен только в случайные моменты времени одиночными байтами или словами.

Более распространенным является ПДП с принудительным отключением процессора от шин системного интерфейса. Для реализации такого режима ПДП системный интерфейс МПС дополняется двумя линиями для передачи управляющих сигналов *ЗАПРОС ЗАХВАТА* (*HLD* для МП КР580ВМ80А) и *ПОДТВЕРЖДЕНИЕ ЗАХВАТА* (*HLDA* для МП КР580ВМ80А).

Управляющий сигнал *ЗАПРОС ЗАХВАТА* формируется контроллером прямого доступа к памяти. Процессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает на системный интерфейс управляющий сигнал *ПОДТВЕРЖДЕНИЕ ЗАХВАТА* и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Контроллер ПДП, используя шины системного интерфейса, осуществляет обмен одним байтом или словом данных с памятью МПС и затем, сняв сигнал *ЗАПРОС ЗАХВАТА*, возвращает управление системным интерфейсом процессору. Как только контроллер ПДП будет готов к обмену следующим байтом, он вновь «захватывает» цикл процессора и т.д. В промежутках между сигналами *ЗАПРОС ЗАХВАТА* процессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний.

Применение в МПС обмена данными с ВУ в режиме ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т.е. количество записываемых в память или читаемых из памяти байт (слов) информации. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байт (слов).

В МПС на базе МП КР580ВМ80А режим *ЗАХВАТ* (режим прямого доступа к памяти) инициируется внешним активным устройством (устройством, в составе которого есть МП) подачей 1 на вход *HLD* МП. Для организации режима ПДП в МП предусмотрен диалоговый обмен управляющими сигналами между ВУ и МП.

Когда ВУ инициирует запрос на ПДП, микропроцессор приостанавливает выполнение основной программы и переводит буферы шин адреса и данных в высокоимпедансное состояние, практически означающее отключение шин адреса (*ША*) и данных (*ШД*) от микропроцессора. Шинами начинает управлять контроллер ПДП, организуя обмен данными между ВУ и памятью МС (рис. 69).

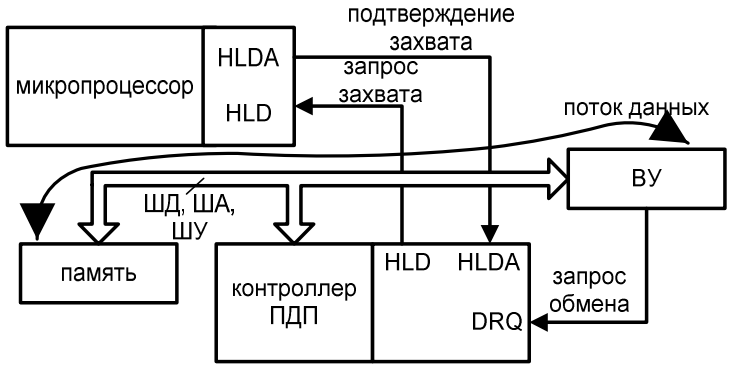


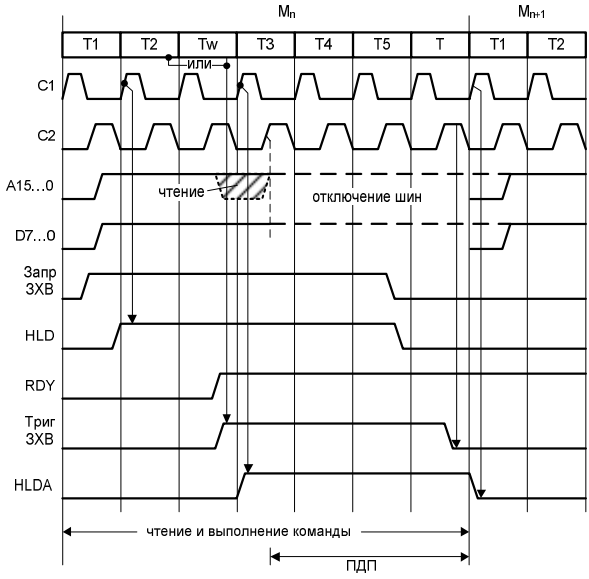
Рис. 69. Организация обмена в режиме ПДП

Инициирование ПДП осуществляет контроллер установкой на выводе *HLD* микропроцессора сигнала высокого уровня. В наиболее простом варианте обмена данными высокий уровень сигнала *HLD* сохраняется до окончания передачи массива данных. Все это время текущий машинный цикл после прохождения всех тактов находится в режиме ожидания окончания ПДП. Об окончании обмена контроллер ПДП сообщает снятием сигнала *HLD*, после чего начинается выполнение следующего машинного цикла. МП реагирует на высокий уровень сигнала *HLD* в текущем машинном цикле, если этот уровень стабилизировался за 180 нс до нарастающего фронта *C2*.

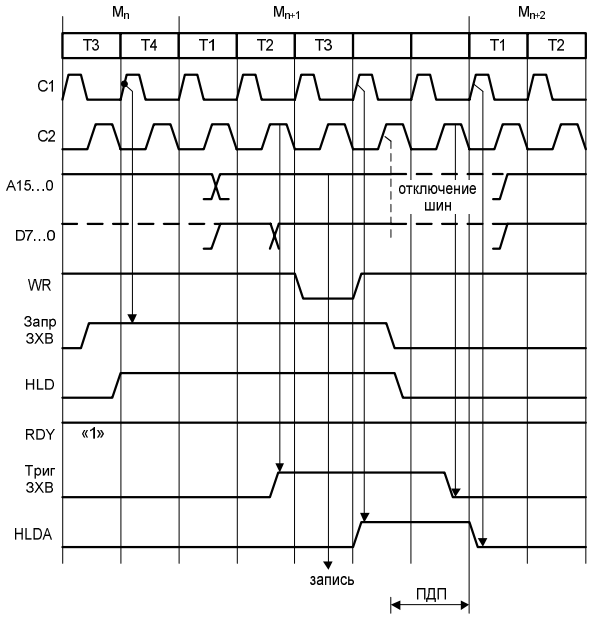
Переход микропроцессора в состояние *ЗАХВАТ* осуществляется по-разному в зависимости от типа машинного цикла.

Если запрос на захват появился во время выполнения операции ввода или чтения, то по переднему фронту *C1* в третьем такте на выходе *HLDA* появляется высокий уровень, подтверждающий состояние *ЗАХВАТ*. Выводы ***D0*** – ***D7*** микропроцессора в третьем такте переходят в высокоимпедансное состояние, а выводы ***A0*** – ***A15*** – в такте, следующем за третьим.

Если выполнялся цикл извлечения команды, который состоит из четырех или пяти тактов, то микропроцессор завершает выполнение этого цикла с «отключенными» выводами ***D0*** – ***D7***, ***A0*** – ***A15*** (рис. 70,а).



а



б

Рис. 70. Временные диаграммы работы МП КР580ВМ80А в режиме *ЗАХВАТ* в цикле чтения (а) и в цикле записи (б)

Если выполнялась операция записи или вывода, сигнал на выходе *HLDA* появляется в такте, следующем за третьим. Это дает возможность микропроцессору завершить процесс записи данных перед переходом в состояние *ЗАХВАТ*. Выводы ***D0*** – ***D7*** и ***A0*** – ***A15*** также переходят в высокоимпедансное состояние в такте, следующем за третьим (см. рис. 70, б).

Сигнал воспринимается процессором на интервале «установка захвата», который предшествует фронту импульса *С2* (см. рис. 70). Под действием сигнала *ЗАХВАТ* при наличии разрешающего сигнала на входе *RDY*, указывающего готовность внешнего устройства, по импульсу *С2* устанавливается внутренний триггер захвата, благодаря чему фронт следующего импульса *С1* переключает в единицу выход *HLDA*. При выполнении циклов чтения или ввода процессор подтверждает захват в начале такта ***Т3*** (по окончании чтения). В других случаях (в циклах записи и вывода) это происходит в такте, следующем за ***Т3*** (по окончании записи). В обоих случаях шины процессора отключаются по фронту импульса *С2*, следующего за импульсом *С1*, по которому выполнялось переключение выхода *HLDA*.

Если процессор подтвердил запрос ПДП в такте ***Т3*** цикла с большим числом тактов, он продолжает выполнение цикла до полного его окончания. Это позволяет осуществить частичное перекрытие процессов обработки информации в процессоре с передачей по каналу ПДП, что улучшает производительность системы.

Процессор выходит из состояния *ЗАХВАТ* примерно в той же последовательности, что и входит. После окончания асинхронного сигнала запроса захвата на входе *HLD* импульсом *С2* сбрасывается триггер захвата, благодаря чему сигнал подтверждения захвата на выходе *HLDA* переключается в состояние низкого потенциала по переднему фронту импульса *С1*. Процессор переходит к выполнению следующего машинного цикла.

В качестве контроллера, организующего обмен данными в режиме ПДП, в МПК К580 имеется контроллер ПДП КР580ВТ57 (рис. 71). Микросхема КР580ВТ57 – четырехканальный программируемый контроллер прямого доступа к памяти (ПДП), предназначенный для высокоскоростного обмена данными между памятью системы и периферийными устройствами путем генерации массива последовательных адресов памяти по требованию периферийного устройства.

Микросхема осуществляет двунаправленный обмен данными между памятью и периферийными устройствами путем формирования в адресном канале микропроцессорной системы параметров

заданного массива адресов ячеек памяти и управляющих сигналов. Массив адресов, по которым происходит обмен данными между периферией и памятью, характеризуется начальным адресом, т.е. первым адресом начала обмена и числом циклов обращений к памяти. После предоставления системной шины со стороны процессора микросхема может осуществить обмен массивом данных между памятью и периферийными устройствами без дальнейшего вмешательства процессора.

Каждый из четырех каналов микросхемы обеспечивает адресацию (путем инкрементирования выработанного адреса) внешней памяти объемом до 16 Кбайт с возможностью задания любого до 64К начальных адресов.

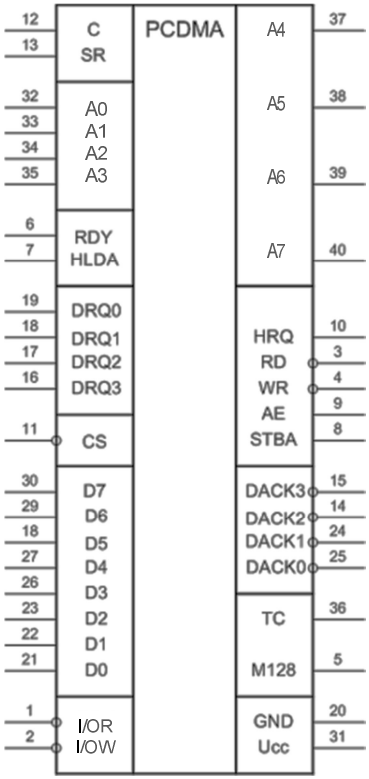


Рис. 71. ИМС контроллера ПДП КР580ВТ57

Назначение выводов ИМС контроллера ПДП КР580ВТ57:
I/OR – вход/выход чтения устройства ввода/вывода.
I/OW – вход/выход записи устройства ввода/вывода.

RD – выход чтения памяти.
WR – выход записи в память.
M128 – выход модуля 128.
RDY – вход сигнала готовности.
HLDA – вход сигнала подтверждения захвата.
STBA – выход стробирующего сигнала адреса.
AE – выход разрешения адреса.
HRQ – выход запроса захвата.
CS – вход выбора микросхемы.
C – вход тактового сигнала.
SR – вход сигнала начальной установки.
DACK3–DACK0 – выходы подтверждения захвата каналов 3–0.
DRQ3–DRQ0 – входы запросов захвата каналов 3–0.
GND – общий.
D7–D0 – входы/выходы шины данных.
Ucc – напряжение питания.
A3–A0 – входы/выходы разрядов шины адреса.
TC – вход сигнала конца счёта.
A7–A4 – выходы разрядов шины адреса.

Младший байт адреса памяти выдается по линиям ***A3*** – ***A0*** и ***A7*** – ***A4***, а старший байт – через шину ***D7*** – ***D0***, поэтому ИМС обычно подключается вместе с буферным регистром.

Режим останов

Как указывалось, МП работает в циклическом режиме, когда после выполнения команды МП инициирует цикл ***M1*** – чтение кода операции очередной команды. Остановить естественный процесс функционирования возможно специальной командой останова *HLT*, которая выполняется за шесть тактов следующим образом (рис. 72).

В машинном цикле ***M1***, состоящем из четырех тактов ***T1***–***T4***, производятся выборка и дешифрирование команды, а в цикле ***M2*** осуществляется собственно выполнение команды. В такте ***T1***, цикла ***M2*** микропроцессор выдает на шину адреса содержимое счетчика команд ***PC***, а на шину данных – байт состояния с установленным битом ***D3*** подтверждения останова. В такте ***T2*** по нарастающему фронту сигнала *C2* с максимальной выдержкой 200 нс внутренние буферы шин адреса и данных переводятся в высокоимпедансное состояние, а по нарастающему фронту сигнала *C1* в следующем такте формируется высокий уровень сигнала ожидания *WI*. Выполнение программы прекращается и в состоянии останова микропроцессор может находиться как угодно долго.

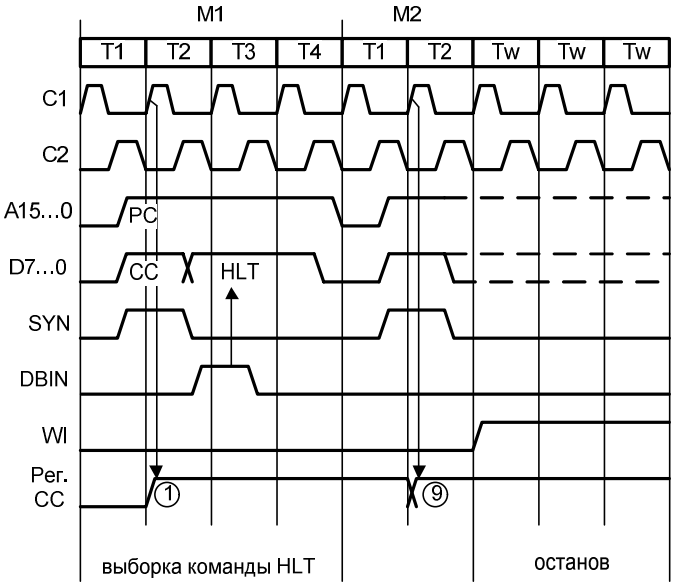


Рис. 72. Временные диаграммы выполнения команды *HLT*

Из состояния останова МП выводится следующими способами:

– путем подачи сигнала высокого уровня на вход сброса *SR* с продолжительностью не менее трех периодов синхронизации. Когда на линии *SR* после этого устанавливается низкий уровень, то по нарастающему фронту сигнала *C1* генерируется внутренний сигнал сброса. Он загружает в счетчик ***PC*** нули и заставляет устройство управления сформировать следующий такт ***T1*** машинного цикла ***M1*** выборки кода операции. Следовательно, МП обращается к ячейке памяти, которая обычно является начальным адресом подпрограммы;

– путем подачи сигнала высокого уровня на вход запроса прерывания *INT*.

Микропроцессор реагирует на этот сигнал только в том случае, если установлен внутренний триггер прерываний (*INTE* = 1). Следовательно, при необходимости нового запуска микропроцессора из состояния останова сигналом *INT* до команды *HLT* необходимо разрешить прерывание командой *EI*. Реагируя на сигнал *INT*, микропроцессор вводит цикл ***M1*** выборки команды *RST*. Если же микропроцессор остановлен и *INTE* = 0, то единственным средством запуска оказывается сигнал *SR*.

Особенностью режима останова является возможность входа в режим ПДП по наличию высокого уровня сигнала *HLD*. Запрос на

ПДП не будет удовлетворяться только в том случае, если был уже подан сигнал *INT* (запрос на прерывание), но не было еще подтверждения прерывания сигналом *INTE*.

Взаимная связь между запросами захвата и прерывания показана на рис. 73. Если выполняется последовательность захвата в режиме ПДП, то запросы прерывания не воспринимаются процессором до окончания обмена. При обработке прерываний запрос захвата не воспринимается процессором до окончания выполнения только первого цикла – цикла чтения команды *RST*. Последовательность прерывания завершается по окончании запрошенного режима ПДП.

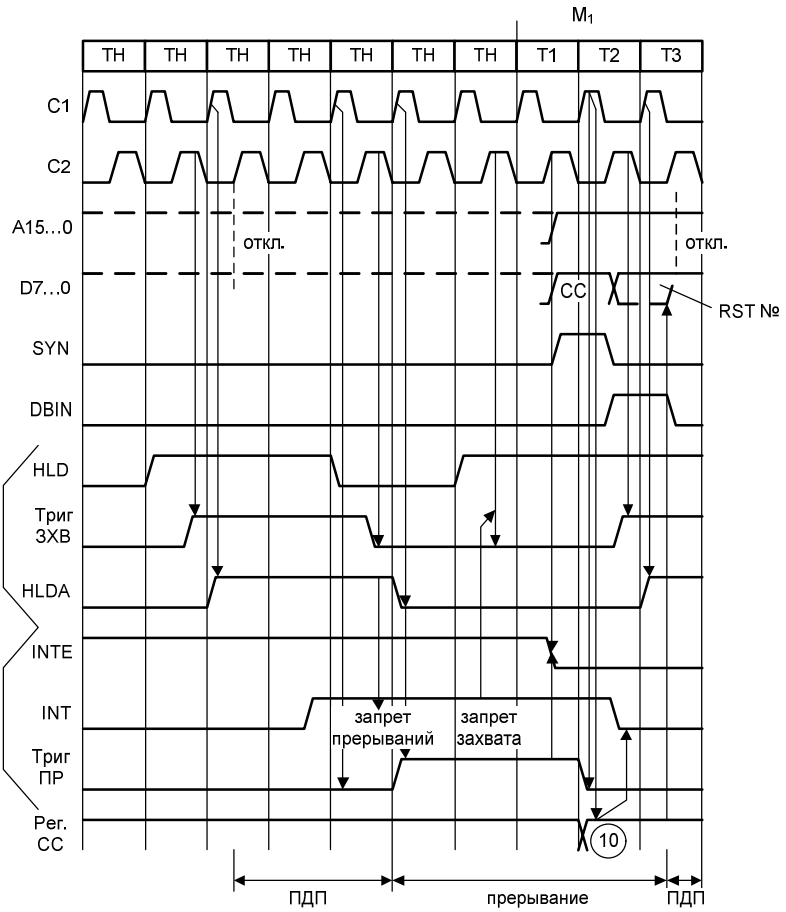


Рис. 73. Временные диаграммы обработки запросов захвата и прерывания в режиме останов

6. **МП КР**580**ВМ**80**А В СТРУКТУРЕ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ**

Для микропроцессорного набора КР580 используется системный интерфейс КР580 Microbus, в котором реализуются различные способы обмена данными с ЗУ и ВУ (с использованием команд ввода-вывода и по аналогии с обращением к памяти).

Номенклатура линий системного интерфейса Microbus определяется выводами БИС МП КР580ИК80. Всего в интерфейсе используются 36 линий, которые разделены на три шины:

16-разрядная шина адреса ***A0***…***A15***;

8-разрядная шина данных ***D0***…***D7***;

12-разрядная шина управления.

Шина адреса предназначена, во-первых, для пересылки 16-разрядов адреса ячейки памяти, что обеспечивает прямую адресацию к памяти емкостью 64 Кбайт, и, во-вторых, для передачи по восьми младшим линиям ***A0***...***A7*** адреса ВУ, что обеспечивает обращение к 256 регистрам контроллеров ВУ при выполнении команды *ВЫВОД* (*OUT*) и к 256 регистрам контроллеров ВУ при выполнении команды *ВВОД* (*IN*).

Шина данных интерфейса Microbus используется в трех различных режимах:

1) для обмена байтом информации между процессором и памятью, т.е. при чтении из памяти команды или данных и при записи в память данных;

2) для обмена данными между процессором и контроллером ВУ при выполнении команд *ВВОД* (*IN*) и *ВЫВОД* (*OUT*);

3) для передачи 8-разрядного слова состояния МП (управляющего слова) во внешние схемы управления.

Необходимость передачи части управляющих сигналов по шине данных возникла из-за малого числа выводов БИС МП, не позволяющего выделить для шины управления более 12 выводов.

Шина управления предназначена для передачи шести входных и выходных сигналов. Вместе со словом состояния МП они обеспечивают как управление работой процессора, так и порядок использования шин адреса и данных при взаимодействии процессора с памятью и ВУ. Использование передаваемого по шине данных управляющего слова позволяет организовать эффективное управление работой системы ввода-вывода МПС при ограниченном количестве выводов БИС микропроцессора.

Центральный микропроцессорный элемент КР580ВМ80А изготавливается в корпусе, имеющем 40 выводов. Этого недостаточно для того, чтобы предоставить каждому сигналу управления отдельный вывод. Поэтому часть сигналов управления передается по выводам внутренней шины данных в режиме разделения вре-

мени. Для этого буферы шины управления соединяются как с управляющим устройством, так и с шиной данных.

Кроме того, МП имеет малую нагрузочную способность выводов

$$I^0_{\text{вых}} \le 2 \text{ мА};\ I^1_{\text{вых}} \le 50 \text{ мкА}.$$

Нагрузочная способность выводов МП позволяет работать только на одну ТТЛ нагрузку. Необходимо усилить выводы МП по току. Для этого используют буферные элементы (или усилители). Для шин такие усилители называются шинными формирователями (рис. 74).

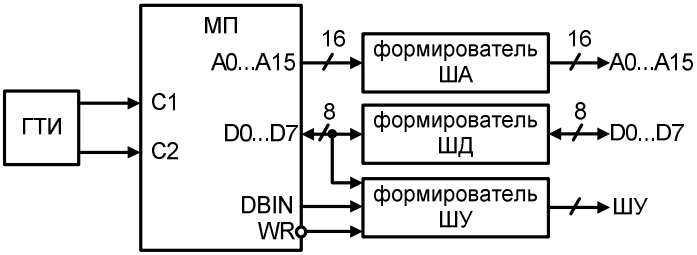


Рис. 74. Типовая структура микропроцессорного модуля на базе МП КР580ВМ80А

В качестве формирователей шины адреса микропроцессорной системы, для увеличения нагрузочной способности выводов микропроцессора применяют ИМС КР580ВА86 и КР580ВА87 (рис. 75). Они представляют собой 8-разрядные двунаправленные приемопередатчики с третьим состоянием, обеспечивающие ток нагрузки до 32 мА. Микросхема ВА87 отличается от ВА86 инвертированными выходами и имеет двунаправленный канал *A*, подключаемый к шине процессора, и двунаправленный канал *B*, 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.

Формирование основных команд, используемых в микропроцессорной системе, вынесено в отдельную микросхему КР580ВК28 или КР580ВК38 (рис. 76). Эта микросхема содержит коммутатор направления передачи данных с буферными усилителями (двунаправленный шинный формирователь), регистр для хранения слова состояния микропроцессора и логическую схему формирования системных сигналов управления. Микросхемы КР580ВК28 и КР580ВК38 отличаются лишь длительностью двух формируемых сигналов:*MEMW* и *I/OW*. ИМС КР580ВК28 (ВК38) применяется в качестве формирователя шин данных и управления микропроцессорной системы на базе МП КР580ВМ80А.

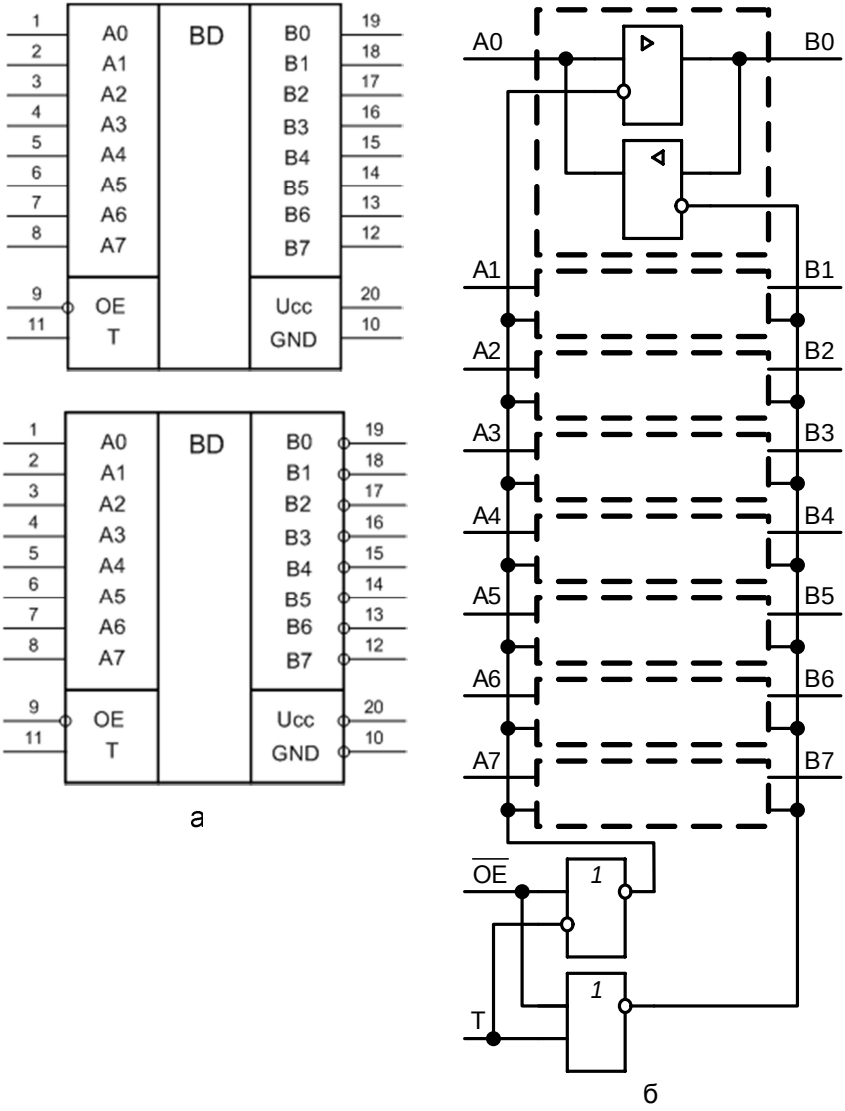
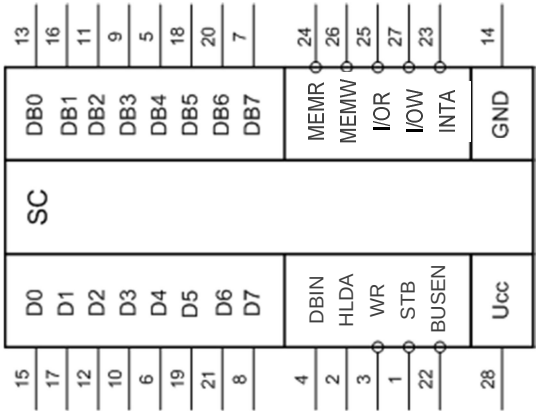
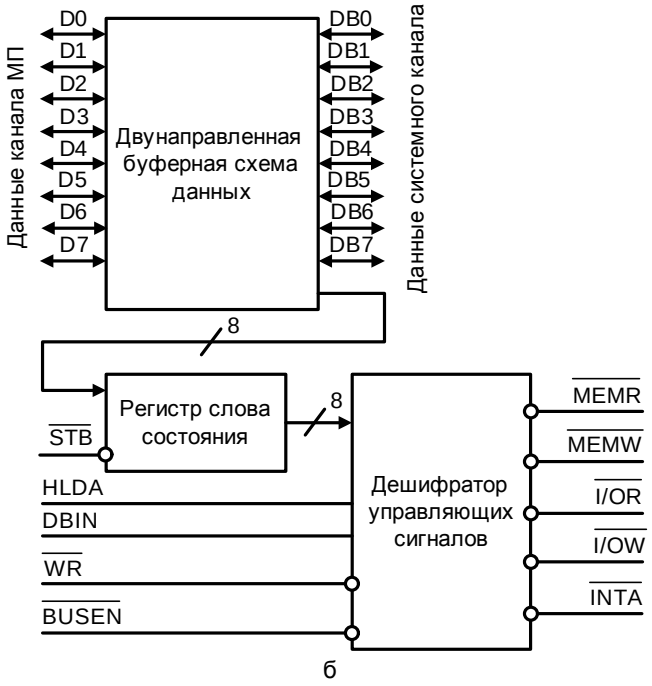


Рис. 75. ИМС шинных формирователей КР580ВА86 и КР580ВА87 (а)
и их структурная схема (б)



а



б

Рис. 76. ИМС системного контроллера КР580ВК28 и КР580ВК38 (а) и их структурная схема (б)

Структурная схема микропроцессорного модуля системы на базе МП КР580ВМ80А представлена на рис. 77. На схеме показа-

ны микросхемы КР580ГФ24 (***G***) – генератор тактовых импульсов, КР580ВМ80 (***CPU***) – центральный микропроцессорный элемент, КР580ВК28 (***SC***) – формирователь системных сигналов управления, КР580ВН59 (***PIC***) – программируемый контроллер прерываний и КР580ВТ57 (***PCDMA***) – программируемый контроллер прямого доступа к памяти. На выходы шины адреса подключаются два 8-разрядных буферных регистра КР580ВА87 или КР580ВА86 (***F***).

Сигналы управления вырабатываются формирователем ***SC*** в начале каждого цикла работы МП по стробу состояния *STB*. Кроме того, три сигнала управления: *HLDA* (подтверждение перехода в режим ПДП), *DBIN* (переключение шины данных на ввод) и *WR* (прием информации) – подаются на ***SC*** из ЦПЭ непосредственно по отдельным проводам. На основе принятых сигналов вырабатываются управляющие системные сигналы *INTA* (подтверждение прерывания), *MEMR* – чтение из ячейки памяти (*Чт.ЗУ*), *I/OR* – чтение из устройства ввода-вывода (*Чт.ВУ*), *MEMW* – запись в ячейку памяти (*Зп.ЗУ*), *I/OW* – запись в устройство ввода-вывода (*Зп.ВУ*).

Как показано на рис. 77, в шину управления входят также линии, подающие сигналы от указанных устройств на процессор. Это сигнал запроса прерывания *INT*, сигнал запроса на захват шин от контроллера прямого доступа к памяти *HOLD*, сигнал готовности устройства к выполнению операции *RDYIN*.

Процессор вырабатывает еще два сигнала управления, которые могут быть использованы при построении микропроцессорной системы. Сюда относятся сигнал *WAIT*, свидетельствующий, что процессор находится в режиме ожидания сигнала готовности *RDY*, и сигнал разрешения прерывания *INTE*, информирующий о том, что процессор готов реагировать на сигнал запроса прерывания *INT*.

Формирование системных управляющих сигналов

В начале каждого цикла работы микропроцессор выдает одновременно сигнал *СИНХРОНИЗАЦИЯ* (*SYN*) на соответствующий выход и информацию о состоянии на шину данных.

Информация о состоянии микропроцессора предназначена для формирования управляющих сигналов.

С помощью, получаемой на выходе регистра слова состояния информации о состоянии микропроцессора и управляющих сигналов *ПРИЕМ* (*DBIN*) и *ВЫДАЧА* (*WR*), выдаваемых микропроцессором, формируются управляющие сигналы:ъ

$$\overline{\text{MEMR (Чт. ЗУ)}} = \overline{\text{D7} \times \text{DBIN}}; \quad \overline{\text{MEMW (Зп. ЗУ)}} = \overline{\overline{\text{D4}} \times \overline{\text{WR}}};$$

$$\overline{\text{I/OR (Чт. ВУ)}} = \overline{\text{D6} \times \text{DBIN}}; \quad \overline{\text{I/OW (ЗП. ВУ)}} = \overline{\text{D4} \times \overline{\text{WR}}}.$$

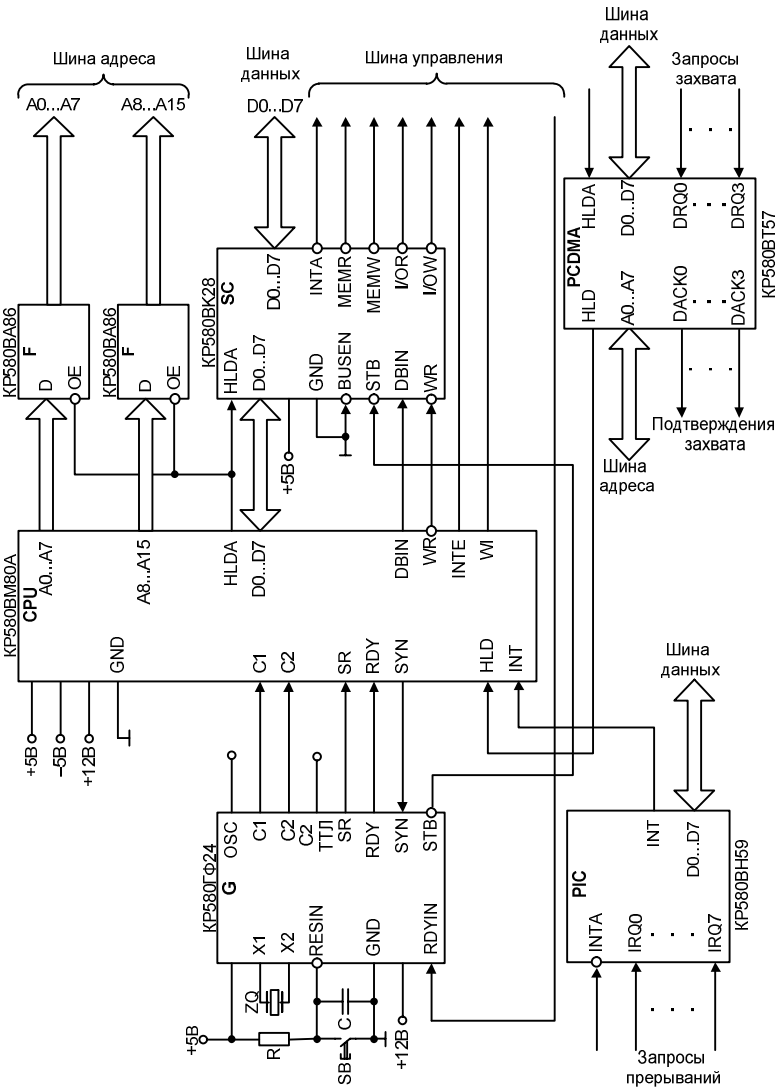


Рис. 77. Микропроцессорный модуль системы на базе МП КР580ВМ80А

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Павловский Е.Г., Жвариков В.А., Кузьмин А.А.. Основы организации ЭВМ и микропроцессоров**:** учеб. пособие и метод. указания к лаб. практикуму. – СПб.: СПб ГПУ, 2011. – 118 с., ил.
2. Мельников А.А., Мельников А.А. (ст.), Мельников А.А. (мл.). Микропроцессоры, микроконтроллеры и однокристальные микропрограммируемые устройства: хрестоматия устройств с микропрограммным управлением. Ч. II. / под ред. А.А. Мельникова. – М.: МГТУ «МАМИ», 2008. – 235 с.
3. Мокрецов В.П.. Микропроцессоры и МПС. Ч. 1. Архитектура микропроцессора К580ВМ80. Организация МП-систем. – Екатеринбург: Изд-во УГТУ-УПИ, 2007. – 143 с.
4. Могнонов П.Б. Организация микропроцессорных систем: учеб. пособие. – Улан-Удэ: Изд-во ВСГТУ, 2003. – 355 с.
5. Ливенцов С.Н., Вильнин А.Д., Горюнов А.Г.. Основы микропроцессорной техники: учеб. пособие. – Томск: Изд-во ТПУ, 2007. – 118 с.
6. Новиков Ю.В., Скоробогатов П.К.. Основы микропроцессорной техники: учеб. пособие. – М.: Интернет-университет информ. технологий; БИНОМ. Лаборатория знаний, 2009. – 357 с., ил.

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА КР580ВМ80А

Условные обозначения:

A – аккумулятор, ***B***, ***C***, ***D***, ***E***, ***H***, ***L*** – регистры общего назначения, обозначаемые в зависимости от выполняемой функции R_S (регистр источник, ***S*** – *SOURCE*) или R_D (регистр приемник, ***D*** – *DESTINATION*).

M – ячейка памяти (от *MEMORY*). В нотации команд МП КР580ВМ80А ячейка памяти ***M*** трактуется как своеобразный регистр, обращение к которому производится только с использованием косвенной адресации. Адрес ***M*** размещается в регистровой паре ***HL***.

BC, ***DE***, ***HL*** – регистровые пары (*RP*), обозначаемые в командах по именам старших регистров пары: ***B*** – имя регистровой пары ***BC***, ***D*** – имя регистровой пары ***DE***, ***H*** – имя регистровой пары ***HL***.

Кодирование регистров и регистровых пар представлено в табл. 1.

Таблица 1

Имя регистра	Код регистра	Имя регистровой пары RP	Код RP
B	000	B-пара	000
C	001		
D	010	D-пара	010
E	011		
H	100	H-пара	100
L	101		
M	110	PSW	110
A	111	SP	110

ADDR – 16-битный адрес памяти; *PORT* – 8-битный адрес периферийного устройства;

data8, *data16* – 8-битные, 16-битные данные; *B2*, *B3* – второй, третий байты команды;

Z*/*NZ, ***C*/*NC***, ***PE*/*PO***, ***P*/*M*** (знак ***S***), ***AC*** – признаки результата;

() – содержимое ячейки памяти или регистра, символическое имя которых заключено в скобки;

[ ] – адрес ячейки памяти.

Список команд микропроцессора КР580ВМ80А приведен в табл. 2, 3. В табл. 2 представлены команды, не воздействующие на. Это – команды пересылок, команды загрузки в стек и извлечения из стека (за исключением команды *POP PSW*, которая изменяет флаги), команды передачи управления и команды управления микропроцессором. При выполнении команд этой группы признаки результата не изменяют своего значения. В табл. 3 представлены команды, при выполнении которых признаки результата (флаги) модифицируются в соответствии со значением результата. Таблицы 2 и 3 отличаются друг от друга наличием (табл. 3) или отсутствием колонки «признаки результата» (табл. 2).

Машинные коды команд МП КР580ВМ80А приведены в табл. 4. Пересечение соответствующих строки и столбца дает машинный код любой команды микропроцессора.

Таблица 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Команды пересылки данных						
Пересылки 8-битных данных	MOV R_D,R_S	1DS	1	1	5	$(R_S) \rightarrow (R_D)$
	MOV M,R_S	16S	1	2	7	$(R_S) \rightarrow (M)$
	MOV R_D,M	1D6	1	2	7	$(M) \rightarrow (R_D)$
Загрузка регистра (непосредственная)	MVI R_D,data8	0D6 B2	2	2	7	$B2 \rightarrow (R_D)$
	MVI M,data8	066 B2	2	3	10	$B2 \rightarrow (M)$
Загрузка пары регистров (непосредственная)	LXI B,data16	001 B2 B3	3	3	10	$B2 \rightarrow (C)$ $B3 \rightarrow (B)$
	LXI D,data16	021 B2 B3	3	3	10	$B2 \rightarrow (E)$ $B3 \rightarrow (D)$
	LXI H,data16	041 B2 B3	3	3	10	$B2 \rightarrow (L)$ $B3 \rightarrow (H)$
	LXI SP,data16	061 B2 B3	3	3	10	$B3B2 \rightarrow (SP)$
Команды пересылок данных в аккумулятор и из аккумулятора						
Загрузка A (прямая)	LDA ADDR	072 B2 B3	3	4	13	$[(B2)(B3)] \rightarrow (A)$
Загрузка A (косвенная)	LDAX B	012	1	2	7	$[(B)(C)] \rightarrow (A)$
	LDAX D	032	1	2	7	$[(D)(E)] \rightarrow (A)$
Запоминание A (прямое)	STA ADDR	062 B2 B3	3	4	13	$(A) \rightarrow [(B2)(B3)]$
Запоминание A (косвенное)	STAX B	002	1	2	7	$(A) \rightarrow [(B)(C)]$
	STAX D	022	1	2	7	$(A) \rightarrow [(D)(E)]$
Ввод	IN PORT	333 B2	2	3	10	$[PORT] \rightarrow (A)$
Вывод	OUT PORT	323 B2	2	3	10	$(A) \rightarrow [PORT]$

Продолжение табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Команды обмена, загрузки и запоминания содержимого регистровой пары HL						
Обмен	XCHG	353	1	1	4	$(H) \leftrightarrow (D)$ $(L) \leftrightarrow (E)$
	XTHL	343	1	5	18	$(L) \leftrightarrow ([SP])$ $(H) \leftrightarrow ([SP + 1])$
Загрузка (прямая)	LHLD ADDR	052 B2 B3	3	5	16	$[(B3)(B2)] \rightarrow (L)$ $[(B3)(B2 + 1)] \rightarrow (H)$
Запоминание (прямое)	SHLD ADDR	042 B2 B3	3	5	16	$(L) \rightarrow [(B3)(B2)]$ $(H) \rightarrow [(B3)(B2 + 1)]$
Загрузка указателя стека	SPHL	371	1	1	5	$(H)(L) \rightarrow (SP)$
Загрузка счётчика команд	PCHL	351	1	1	5	$(H)(L) \rightarrow (PC)$
Команды загрузки в стек и извлечения из стека						
Загрузка в стек	PUSH B	305	1	3	11	$(B) \rightarrow [(SP - 1)]$ $(C) \rightarrow [(SP - 2)]$ $(SP) = (SP - 2)$
	PUSH D	325	1	3	11	$(D) \rightarrow [(SP - 1)]$ $(E) \rightarrow [(SP - 2)]$ $(SP) = (SP - 2)$
	PUSH H	345	1	3	11	$(H) \rightarrow [(SP - 1)]$ $(L) \rightarrow [(SP - 2)]$ $(SP) = (SP - 2)$

Продолжение табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Загрузка в стек	PUSH PSW	365	1	3	11	$(A) \rightarrow [(SP-1)]$ $(F) \rightarrow [(SP-2)]$ $(SP) = (SP-2)$
Извлечение из стека	POP B	301	1	3	10	$[(SP)] \rightarrow (C)$ $[(SP+1)] \rightarrow (B)$ $(SP) = (SP+2)$
	POP D	321	1	3	10	$[(SP)] \rightarrow (E)$ $[(SP+1)] \rightarrow (D)$ $(SP) = (SP+2)$
	POP H	341	1	3	10	$[(SP)] \rightarrow (L)$ $[(SP+1)] \rightarrow (H)$ $(SP) = (SP+2)$
	POP PSW	361	1	3	10	$[(SP)] \rightarrow (F)$ $[(SP+1)] \rightarrow (A)$ $(SP) = (SP+2)$
Команды безусловной передачи управления						
Безусловный переход	JMP ADDR	303 B2 B3	3	3	10	$B3B2 \rightarrow (PC)$
Безусловный вызов подпрограммы	CALL ADDR	315 B2 B3	3	5	17	$(PC) \rightarrow [(SP-1)(SP-2)]$ $(SP) = (SP-2)$ $B3B2 \rightarrow (PC)$

Продолжение табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Безусловный возврат из подпрограммы	RET	311	1	3	10	$[(SP)(SP+1)] \rightarrow (PC)$ $(SP) = (SP+2)$
Загрузка счётчика команд (множественное ветвление)	PCHL	351	1	1	5	$(H)(L) \rightarrow (PC)$
Вызов прерывающей программы	RST N	3N7	1	3	11	$(PC) \rightarrow [(SP-1)(SP-2)]$ $(SP) = (SP-2)$; $(PC) = 000\,0N0$
Команды условного перехода						
Переход по нулевому значению флага Z (ненулевому результату)	JNZ ADDR	302 B2 B3	3	3	10	Переход по условию, заданному в команде. Если условие выполняется, то осуществляется переход к команде, размещенной по адресу B2 B3: $B3\,B2 \rightarrow (PC)$. Если условие не выполняется, то осуществляется переход к следующей по порядку команде: $(PC) = (PC) + 3$.
Переход по единичному значению флага Z (нулевому результату)	JZ ADDR	312 B2 B3	3	3	10	
Переход по нулевому значению флага C (при отсутствии переноса)	JNC ADDR	322 B2 B3	3	3	10	
Переход по единичному значению флага C (при наличии переноса)	JC ADDR	332 B2 B3	3	3	10	
Переход по нулевому значению флага P (при отсутствии четности)	JPO ADDR	342 B2 B3	3	3	10	

Продолжение табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Переход по единичному значению флага P (при наличии четности)	JPE ADDR	352 B2 B3	3	3	10	
Переход по нулевому значению флага S (положительное значение)	JP ADDR	362 B2 B3	3	3	10	
Переход по единичному значению флага S (отрицательное значение)	JM ADDR	372 B2 B3	3	3	10	
Команды условного вызова подпрограмм						
Вызов подпрограммы при Z = 0	CNZ ADDR	304 B2 B3	3	3/5	11/17	Вызов подпрограммы при выполнении условия, заданного в команде. Если условие выполняется, то осуществляется переход к подпрограмме, т.е. к команде, размещенной по адресу B2 B3: $B3\,B2 \rightarrow (PC)$. Если условие не выполняется, то осуществляется переход к следующей по порядку команде: $(PC) = (PC) + 3$.
Вызов подпрограммы при Z = 1	CZ ADDR	314 B2 B3	3	3/5	11/17	
Вызов подпрограммы при C = 0	CNC ADDR	324 B2 B3	3	3/5	11/17	
Вызов подпрограммы при C = 1	CC ADDR	334 B2 B3	3	3/5	11/17	
Вызов подпрограммы при P = 0	CPO ADDR	344 B2 B3	3	3/5	11/17	
Вызов подпрограммы при P = 1	CPE ADDR	354 B2 B3	3	3/5	11/17	

Продолжение табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Вызов подпрограммы при S = 0	CP ADDR	364 B2 B3	3	3/5	11/17	
Вызов подпрограммы при S = 1	CM ADDR	374 B2 B3	3	3/5	11/17	
Команды условного возврата из подпрограмм						
Возврат из подпрограммы при ненулевом результате (Z = 0)	RNZ	300	1	1/3	5/11	Возврат из подпрограммы при выполнении условия, заданного в команде. Если условие выполняется, то осуществляется переход к команде, размещенной по адресу возврата из стека: $[(SP)(SP) + 1] \rightarrow (PC)$, $(SP) = (SP) + 2$. Если условие не выполняется, то осуществляется переход к следующей по порядку команде: $(PC) = (PC) + 3$.
Возврат из подпрограммы при нулевом результате (Z = 1)	RZ	310	1	1/3	5/11	
Возврат из подпрограммы при отсутствии переноса (C = 0)	RNC	320	1	1/3	5/11	
Возврат из подпрограммы при наличии переноса (C=1)	RC	330	1	1/3	5/11	
Возврат из подпрограммы при отсутствии четности (P = 0)	RPO	340	1	1/3	5/11	
Возврат из подпрограммы при наличии четности (P=1)	RPE	350	1	1/3	5/11	

Окончание табл. 2

Название команды	Мнемоника команды	Код операции	Параметры команды			Описание команды
			б	ц	т	
Возврат из подпрограммы при положительном результате (S = 0)	RP	360	1	1/3	5/11	
Возврат из подпрограммы при отрицательном результате (S = 1)	RM	370	1	1/3	5/11	
Команды управления микропроцессором						
Разрешение прерываний	EI	373	1	1	4	Формирование сигнала INTE = 1
Запрет прерываний	DI	363	1	1	4	Формирование сигнала INTE = 0
Холостая команда	NOP	000	1	1	4	Переход к следующей команде без операции
Команда останова	HLT	166	1	1	4	Останов. Возможно продолжение программы по запросу прерывания

Таблица 3

Название команды	Мнемоника команды	Код операции	Признаки (флаги) результата					Параметры команды			Описание команды
			S	Z	AC	P	C	б	ц	т	
Команды арифметических операций											
Сложение	ADD R_S	20S	+	+	+	+	+	1	1	4	$(A) + (R_S) \rightarrow (A)$
	ADD M	206	+	+	+	+	+	1	2	7	$(A) + (M) \rightarrow (A)$
	ADI data8	306 B2	+	+	+	+	+	2	2	7	$(A) + B2 \rightarrow (A)$
Сложение с переносом	ADC R_S	21S	+	+	+	+	+	1	1	4	$(A) + (R_S) + C \rightarrow (A)$
	ADC M	216	+	+	+	+	+	1	2	7	$(A) + (M) + C \rightarrow (A)$
	ACI data8	316 B2	+	+	+	+	+	2	2	7	$(A) + B2 + C \rightarrow (A)$
Вычитание	SUB R_S	22S	+	+	+	+	+	1	1	4	$(A) - (R_S) \rightarrow (A)$
	SUB M	226	+	+	+	+	+	1	2	7	$(A) - (M) \rightarrow (A)$
	SUI data8	326 B2	+	+	+	+	+	2	2	7	$(A) - B2 \rightarrow (A)$
Вычитание с заёмом	SBB R_S	23S	+	+	+	+	+	1	1	4	$(A) - (R_S) - C \rightarrow (A)$
	SBB M	236	+	+	+	+	+	1	2	7	$(A) - (M) - C \rightarrow (A)$
	SBI data8	336 B2	+	+	+	+	+	2	2	7	$(A) - B2 - C \rightarrow (A)$
Сравнение (неразрушающее вычитание)	CMP R_S	27S	+	+	+	+	+	1	1	4	$(A) - (R_S)$
	CMP M	276	+	+	+	+	+	1	2	7	$(A) - (M)$
	CPI data8	376 B2	+	+	+	+	+	2	2	7	$(A) - B2$
Инкремент (увеличение содержимого адресуемого источника на 1)	INR R_D	0D4	+	+	+	+	–	1	1	5	$(R_D) + 1 \rightarrow (R_D)$
	INR M	064	+	+	+	+	–	1	3	10	$(M) + 1 \rightarrow (M)$
	INX B	003	–	–	–	–	–	1	1	5	$(B)(C) + 1 \rightarrow (B)(C)$
	INX D	023	–	–	–	–	–	1	1	5	$(D)(E) + 1 \rightarrow (D)(E)$
	INX H	043	–	–	–	–	–	1	1	5	$(H)(L) + 1 \rightarrow (H)(L)$

Продолжение табл. 3

Название команды	Мнемоника команды	Код операции	Признаки (флаги) результата					Параметры команды			Описание команды
			S	Z	AC	P	C	б	ц	т	
	INX SP	063	–	–	–	–	–	1	1	5	$(SP) + 1 \rightarrow (SP)$
Декремент (уменьшение содержимого адресуемого источника на 1)	DCR R_D	0D5	+	+	+	+	–	1	1	5	$(R_D) - 1 \rightarrow (R_D)$
	DCR M	065	+	+	+	+	–	1	3	10	$(M) - 1 \rightarrow (M)$
	DCX B	013	–	–	–	–	–	1	1	5	$(B)(C) - 1 \rightarrow (B)(C)$
	DCX D	033	–	–	–	–	–	1	1	5	$(D)(E) - 1 \rightarrow (D)(E)$
	DCX H	053	–	–	–	–	–	1	1	5	$(H)(L) - 1 \rightarrow (H)(L)$
	DCX SP	073	–	–	–	–	–	1	1	5	$(SP) - 1 \rightarrow (SP)$
Двойное сложение	DAD B	011	–	–	–	–	+	1	3	10	$(HL) + (BC) \rightarrow (HL)$
	DAD D	031	–	–	–	–	+	1	3	10	$(HL) + (DE) \rightarrow (HL)$
	DAD H	051	–	–	–	–	+	1	3	10	$(HL) + (HL) \rightarrow (HL)$
	DAD SP	071	–	–	–	–	+	1	3	10	$(HL) + (SP) \rightarrow (HL)$
Десятичная коррекция	DAA	047	+	+	+	+	+	1	1	4	
Команды логических операций											
Логическое умножение	ANA R_S	24S	+	+	–	+	0	1	1	4	$(A) \wedge (R_S) \rightarrow (A)$
	ANA M	246	+	+	–	+	0	1	2	7	$(A) \wedge (M) \rightarrow (A)$
	ANI data8	346 B2	+	+	–	+	0	2	2	7	$(A) \wedge B2 \rightarrow (A)$
Исключающее ИЛИ (сложение по mod 2)	XRA R_S	25S	+	+	–	+	0	1	1	4	$(A) \oplus (R_S) \rightarrow (A)$
	XRA M	256	+	+	–	+	0	1	2	7	$(A) \oplus (M) \rightarrow (A)$
	XRI data8	356 B2	+	+	–	+	0	2	2	7	$(A) \oplus B2 \rightarrow (A)$

Окончание табл. 3

Название команды	Мнемоника команды	Код операции	Признаки (флаги) результата					Параметры команды			Описание команды
			S	Z	AC	P	C	б	ц	т	
Логическое сложение	ORA R_S	26S	+	+	–	+	0	1	1	4	$(A) \vee (R_S) \rightarrow (A)$
	ORA M	266	+	+	–	+	0	1	2	7	$(A) \vee (M) \rightarrow (A)$
	ORI data8	366 B2	+	+	–	+	0	2	2	7	$(A) \vee B2 \rightarrow (A)$
Инверсия А	CMA	057	–	–	–	–	–	1	1	4	$(\overline{A}) \rightarrow (A)$
Команды циклических сдвигов											
Сдвиг влево	RLC	007	–	–	–	–	+	1	1	4	
Сдвиг вправо	RRC	017	–	–	–	–	+	1	1	4	
Циклический сдвиг влево через перенос	RAL	027	–	–	–	–	+	1	1	4	
Циклический сдвиг вправо через перенос	RAR	037	–	–	–	–	+	1	1	4	
Команды управления флагом С											
Установка переноса	STC	067	–	–	–	–	+	1	1	4	$1 \rightarrow C$
Инверсия переноса	CMC	077	–	–	–	–	+	1	1	4	$\overline{C} \rightarrow C$

Окончание приложения

Таблица 4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B,16	STAX B	INX B	INR B	DCR B	MVI B,08	RLC	—	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C,08	RRC
1	—	LXI D,16	STAX D	INX D	INR D	DCR D	MVI D,08	RAL	—	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E,08	RAR
2	—	LXI H,16	SHLD ADR	INX H	INR H	DCR H	MVI H,08	DAA	—	DAD H	LHLD ADR	DCX H	INR L	DCR L	MVI L,08	CMA
3	—	LXI SP,16	STA ADR	INX SP	INR M	DCR M	MVI M,08	STC	—	DAD SP	LDA ADR	DCX SP	INR A	DCR A	MVI A,08	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	MOV M,M	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ ADR	JMP ADR	CNZ ADR	PUSH B	ADI 08	RST 0	RZ	RET	JZ ADR	—	CZ ADR	CALL ADR	ACI 08	RST 1
D	RNC	POP D	JNC ADR	OUT 08	CNC ADR	PUSH D	SUI 08	RST 2	RC	—	JC ADR	IN 08	CC ADR	—	SBI 08	RST 3
E	RPO	POP H	JPO ADR	XTHL	CPO ADR	PUSH H	ANI 08	RST 4	RPE	PCHL	JPE ADR	XCHG	CPE ADR	—	XRI 08	RST 5
F	RP	POP PSW	JP ADR	DI	CP ADR	PUSH PSW	ORI 08	RST 6	RM	SPHL	JM ADR	EI	CMP ADR	—	CPI 08	RST 7

Учебное издание

ЛУКЬЯНОВ Сергей Иванович
ШВИДЧЕНКО Дмитрий Владимирович
СУСПИЦЫН Евгений Сергеевич
и др.

Основы микропроцессорной техники

Учебное пособие

Редактор Т.А. Колесникова
Компьютерная верстка Г.Н. Лапиной

Подписано в печать 24.01.2013. Рег. № 31-13. Формат 60x84/16. Бумага тип. № 1.
Плоская печать. Усл.печ.л. 8,75. Тираж 100 экз. Заказ №50

Издательский центр ФГБОУ ВПО «МГТУ»
455000, Магнитогорск, пр. Ленина, 38
Полиграфический участок ФГБОУ ВПО «МГТУ»

Основы микропроцессорной техники

Магнитогорск
2013